Botnet Detection Engine

Version 1.0

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Namespace Documentation

3.1 detection_engine_modules.cmd_line_args Namespace Reference

Functions

- def **get_cmd_line_args** ()

3.1.1 Detailed Description

```
The cmd_line_args Module.
```

3.1.2 Function Documentation

3.1.2.1 get_cmd_line_args()

```
def detection_engine_modules.cmd_line_args.get_cmd_line_args ( )


Parses the command line arguments.

Returns:
    args (dict): A populated dictionary, containing the parsed arguments from sys.argv.
```

3.2 detection_engine_modules.Detector Namespace Reference

Classes

- class **Detector**

3.2.1 Detailed Description

The Detector Module.

3.3 detection_engine_modules.Logger Namespace Reference

Classes

- class **Logger**

3.3.1 Detailed Description

The Logger Module.

3.4 detection_engine_modules.Model Namespace Reference

Classes

- class **Model**

3.4.1 Detailed Description

The Model Module.

3.5 detection_engine_modules.Sniffer Namespace Reference

Classes

- class **Sniffer**

3.5.1 Detailed Description

The Sniffer Module.

3.6 detection_engine_modules.Websocket_Client Namespace Reference

Classes

- class **Websocket_Client**

3.6.1 Detailed Description

The Websocket_Client Module.

Chapter 4

Class Documentation

4.1 detection_engine_modules.Detector.Detector Class Reference

Public Member Functions

- def **__init__** (self, args)
- def **__del__** (self)
- def **run** (self)
- def **load_models** (self)
- def **process_flow** (self, flow_string)
- def **predict** (self, flow)
- def **generate_alert** (self, flow, prediction, predicted_model_data)
- def **get_model_data** (self, data_file_path)
- def **flow_feature_exclusion** (self, flow)

Public Attributes

- **gui**
- **logging**
- **debug**
- **read**
- **dataset_feature_columns**
- **models**
- **model_directory**
- **socket_addr**
- **sniffer**
- **ws_client**
- **logger**

4.1.1 Detailed Description

```
Detector class handles the sniffing and prediction of network flows via the multiple Model objects instantiate
Also allows handling of labelled network flows and generates alerts in the event of positive predicitons, whic
to file and sent to a GUI instance via a websocket client.
```

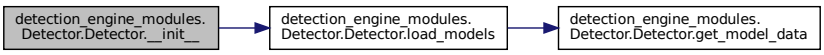
4.1.2 Constructor & Destructor Documentation

4.1.2.1 __init__()

```
def detection_engine_modules.Detector.Detector.__init__ (
              self,
              args )
```

```
Constructor for the Detector.

Takes in parsed command line argument data and sets up the Detector's mode accordingly.

These modes include sniffing from the network, processing a local .pcap file or reading
from a pre-processed network flow file.

Also loads the relevent models into an object array to facilitate model prediciton.

Args:
    args (dict): Parsed command line arguments to specify the Detector's mode of operation.

Attributes:
    gui (bool): Specifies the GUI mode required.
    logging (bool): Specifies the logging mode required.
    debug (bool): Specifies the debugging mode required.
    read (None, string): Specifies the file read mode required.
    dataset_feature_columns (:obj:'list' of :obj:'str'): The required dataset feature column headings used in
    models (:obj:'list' of :obj:'Model'): List to store successfully instantiated Model objects.
    sniffer (:obj':'Sniffer'): Sniffer object storage.
    ws_client (:obj':'Websocket_Client'): Websocket Client object storage.
    logger (:obj:'Logger'): Logger object storage.
```

Here is the call graph for this function:



4.1.2.2 __del__()

```
def detection_engine_modules.Detector.Detector.__del__ (
              self )
```

```
Detector destructor.
```

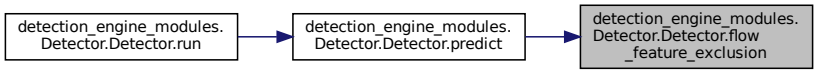
4.1.3 Member Function Documentation

4.1.3.1 flow_feature_exclusion()

```
def detection_engine_modules.Detector.Detector.flow_feature_exclusion (
              self,
              flow )
```

```
Excludes flow DataFrame features that are not used in the prediction of the data.

Args:
    flow (:obj:'DataFrame'): The flow's DataFrame that is to be processed via the exclusion process.

Returns:
    feature_excluded_flow (:obj:'DataFrame'): The processed flow DataFrame, with the relevant columns maintain
```

Here is the caller graph for this function:

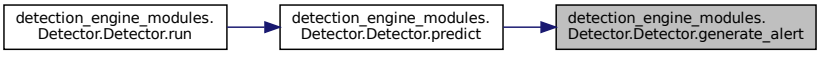


4.1.3.2 generate_alert()

```
def detection_engine_modules.Detector.Detector.generate_alert (
              self,
              flow,
              prediction,
              predicted_model_data )
```

```
Generates an alert json data structure from the specific data of each positive prediction model.

Args:
    flow (:obj:'DataFrame'): The flow DataFrame that is responsible for the alert's generation.
    prediction (string): The prediction of the flow.
    predicted_model_data (json-formatted string): The metadata of the models that made a prediction that cause

Returns:
    alert (json-formatted string): The generated alert data in json format.
```

Here is the caller graph for this function:

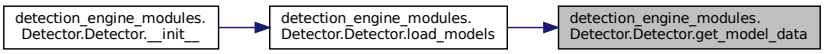


4.1.3.3 get_model_data()

```
def detection_engine_modules.Detector.Detector.get_model_data (
              self,
              data_file_path )
```

```
Opens and retrieves all of the metadata stored for the models from the metadata file.

If the file cannot be read, the system exits (cannot operate without loaded model metadata).

Args:
    data_file_path (string): The relative filepath and name of the model metadata file.

Returns:
    model_data (json-formatted string): The metadata retrieved from the file.
```

Here is the caller graph for this function:

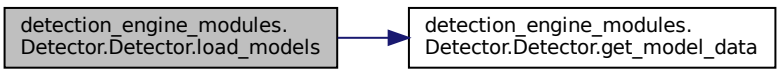


4.1.3.4 load_models()

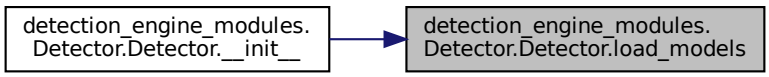
```
def detection_engine_modules.Detector.Detector.load_models (
              self )
```

```
De-serialisation of trained Machine Learning models that are contained in the
'Models' directory.

Gets the model metadata, from 'model_data.json', and attempts to load the expected models with
and passes the relevant metadata to the instantiated Model objects.

Successfully loaded Model objects get stored in the 'models' list, and returned.

Note:
    If no valid models are found in the valid directory, and thus aren't loaded, the process exits.

Returns:
    models (:obj:'list' of :obj:'Model'): Successfully loaded Model instances.
```

Here is the call graph for this function:



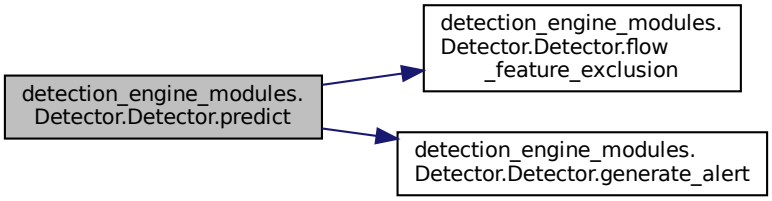
Here is the caller graph for this function:



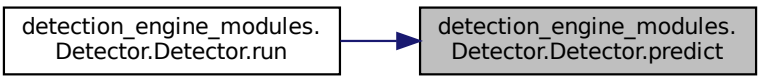
4.1.3.5 predict()

```
def detection_engine_modules.Detector.Detector.predict (
              self,
              flow )

Predicts the label of the given network flow data.
If the flow is predicted as botnet, we return the data about the models that made the prediction.

Args:
    flow (:obj:'DataFrame'): The processed flow DataFrame object that is to be predicted by the models.

Returns:
    prediction (string): The prediction verdict ('Normal' or 'Botnet')
    alert (json-formatted string): The json model data, in the form of a generated alert, from the positive mo
```

Here is the call graph for this function:



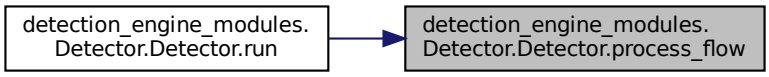
Here is the caller graph for this function:



4.1.3.6 process_flow()

```
def detection_engine_modules.Detector.Detector.process_flow (
              self,
              flow_string )

Processes flow_string into the valid DataFrame, with the column headers included.

Args:
    flow_string (string): A string, containing the comma-seperated value network flow data.

Returns:
    processed_flow The flow DataFrame; or False in the case of the flow_string being that of a the network flo
```

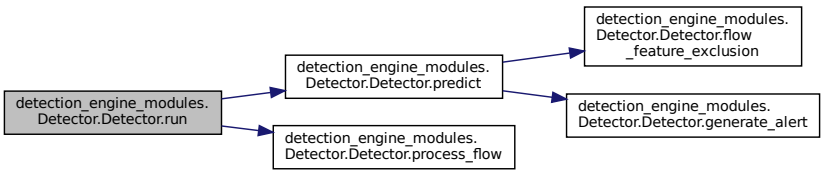
Here is the caller graph for this function:



4.1.3.7 run()

```
def detection_engine_modules.Detector.Detector.run (
              self )

Main running loop for the detector.

Gets flows from the network flow sniffer, predicts the behaviour via the models
and outputs the labelled network flows and any alert data if encountered.

If GUI is enabled, sends labelled network flows and alerts via the Websocket_Client
to the GUI.

If logging is enabled, sends the labelled network flows and alerts to the Logger to handle
writing to their respective files.

Returns:
    0 once main running loop is broken.
```

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- /home/asmarus/Desktop/botnetdetectionengine/detection_engine_modules/Detector.py

4.2 detection_engine_modules.Logger.Logger Class Reference

Public Member Functions

- def **__init__** (self)
- def **__del__** (self)
- def **open_flow_file** (self)
- def **write_flow_to_file** (self, flow_string)
- def **open_alert_file** (self)
- def **write_alert_to_file** (self, alert)

Public Attributes

- **alert_no**
- **flow_file**
- **alert_file**
- **flow_file_name**
- **alert_file_name**

4.2.1 Detailed Description

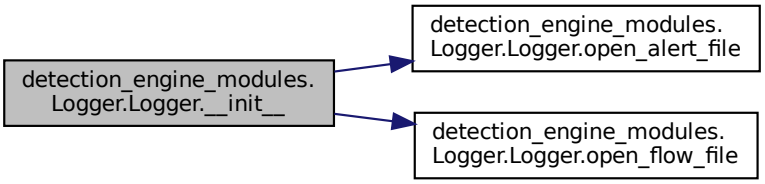
```
The Logger class intends to control the flow log and alert log file handles.

Allows the opening and writing of the relevant data to their specific files.
```

4.2.2 Constructor & Destructor Documentation

4.2.2.1 __init__()

```
def detection_engine_modules.Logger.Logger.__init__ (
              self )

The constructor of the Logger object.

Attributes:
    alert_no (int): The incremental counter for the alert number.
    flow_file (None or 'obj':FILE): The flow file object storage.
    alert_file (None or 'obj':FILE): The alert file object storage.
    flow_file_name (string): The file name for the flow file.
    alert_file_name (string): The file name for the alert file.
```

Here is the call graph for this function:



4.2.2.2 __del__()

```
def detection_engine_modules.Logger.Logger.__del__ (
              self )
```

The Logger object's destructor.

Explicitally closes the file handle objects.

4.2.3 Member Function Documentation

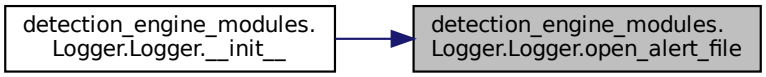
4.2.3.1 open_alert_file()

```
def detection_engine_modules.Logger.Logger.open_alert_file (
              self )
```

Creates a file handle for the alert log file, with the ability to write to the file.

```
Returns:
    0: If successful.
```

Here is the caller graph for this function:



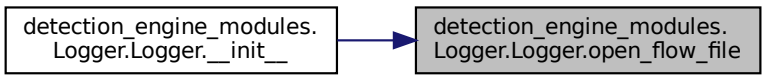
4.2.3.2 open_flow_file()

```
def detection_engine_modules.Logger.Logger.open_flow_file (
              self )
```

Creates a file handle for the flow log file, with the ability to write to the file.

```
Returns:
    0: If successful.
```

Here is the caller graph for this function:



4.2.3.3 write_alert_to_file()

```
def detection_engine_modules.Logger.Logger.write_alert_to_file (
              self,
              alert )
```

```
Formats and writes the json alert data to the file.

Args:
    alert (json-formatted string): The json-formatted alert data is to be written to the file.

Returns:
    0: If successful.
```

4.2.3.4 write_flow_to_file()

```
def detection_engine_modules.Logger.Logger.write_flow_to_file (
              self,
              flow_string )
```

```
Writes the flow data to the file.

Args:
    flow_string (string): The flow string data that is to be written to the file.

Returns:
    0: If successful.
```

The documentation for this class was generated from the following file:

- /home/asmarus/Desktop/botnetdetectionengine/detection_engine_modules/Logger.py

4.3 detection_engine_modules.Model.Model Class Reference

Public Member Functions

- def **__init__** (self, model_directory, model_file_name, model_metadata, debug)
- def **__del__** (self)
- def **load_model** (self)
- def **predict** (self, flow)

Public Attributes

- **model_object**
- **model_metadata**
- **model_file_name**
- **rel_model_file_path**
- **debug**
- **loading_status**

4.3.1 Detailed Description

```
The Model class handles the deserialised model objects.

The Model objects must be instantated with the directory that the models are stored in, the model file name an

Predictions on the status of the DataFrame can then be made.
```

4.3.2 Constructor & Destructor Documentation

4.3.2.1 __init__()

```
def detection_engine_modules.Model.Model.__init__ (
              self,
              model_directory,
              model_file_name,
              model_metadata,
              debug )


Constructor for the Model objects.

Takes in a directory name, model file, name, and the model's data. This is retrieved and inputted from the Det

Args:
    model_directory (string): The name of the directory that the models are stored in.
    model_file_name (string): The name of the model file that is to be loaded.
    model_metadata (json-formatted string): The model's related metadata.

Attributes:
    model_object (None or :obj:'RandomForestClassifier object'): The variable in which the deserialised model
    model_metadata (json-formatted string): The model's related metadata.
    model_file_name (string): The name of the model file that is to be loaded.
    rel_model_file_path (string): The full relative path for the model file.
```

Here is the call graph for this function:

4.3.2.2 __del__()

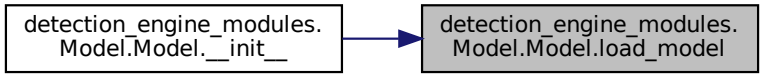
```
def detection_engine_modules.Model.Model.__del__ (
              self )


The Model object's destructor.
```

4.3.3 Member Function Documentation

4.3.3.1 load_model()

```
def detection_engine_modules.Model.Model.load_model (
              self )

Opens the model file and de-serialises the pickeled object data into the model object's storage variable.

Returns:
    loading_status (bool): The status of the model's deserialisation.
```

Here is the caller graph for this function:



4.3.3.2 predict()

```
def detection_engine_modules.Model.Model.predict (
              self,
              flow )

Makes a prediction, against the deserialised loaded model in memory, on the flow data to determine the data's

Args:
    flow (:obj:'DataFrame'): The processed network flow DataFrame which it to be predicted on.

Returns:
    prediction (string): The prediction, either 'Normal' or 'Botnet', depending on the model's prediction resu
```

The documentation for this class was generated from the following file:

- /home/asmarus/Desktop/botnetdetectionengine/detection_engine_modules/Model.py

4.4 detection_engine_modules.Sniffer.Sniffer Class Reference

Public Member Functions

- def **__init__** (self, read_from_file=False)
- def **__del__** (self)
- def **start** (self)
- def **get_flow** (self)

Public Attributes

- **tcpdump**
- **argus**
- **ra**
- **read_from_file**
- **file**
- **argus_command**
- **tcpdump_command**
- **ra_command**

4.4.1 Detailed Description

```
Sniffs raw data from the Network Interface Card (NIC), generally connected to a
SPAN'd switchport.

Performs netflow feature extraction for sniffed data, or inputted '.pcap' files.

Network flow sniffing mimics bash shell behaviour of:
$ TCPDUMP (interface) | ARGUS | RA CLIENT (CSV Formatted Network Flow Exporter).

Alternative behaviour is to run offline ('.pcap' or network flow ('binetflow'
or '.csv') files).
```

4.4.2 Constructor & Destructor Documentation

4.4.2.1 __init__()

```
def detection_engine_modules.Sniffer.Sniffer.__init__ (
              self,
              read_from_file = False )


The constructor of the Sniffer object.

Attributes:
    tcpdump (None or :obj:'Subprocess'): The subprocess handle for the tcpdump process.
    argus (None or :obj:'Subprocess'): The subprocess handle for the argus process.
    ra (None or :obj:'Subprocess'): The subprocess handle for the ra process.
    read_from_file (False or string): The file that is to be read from, or False if sniffing from the network
    file (None or :obj:'FILE'): The file handle object, if there is a file to be opened (i.e. not in network s
    argus_command (string): The argus command to run in the subprocess.
    tcpdump_command (string): The tcpdump command to run in the subprocess.
    ra_command (string): The ra command to run in the subprocess.
```

4.4.2.2 __del__()

```
def detection_engine_modules.Sniffer.Sniffer.__del__ (
              self )


The Sniffer object's destructor.

Explicitally kills the subprocesses via their parent shell PIDs (their PGIDs).
```

4.4.3 Member Function Documentation

4.4.3.1 get_flow()

```
def detection_engine_modules.Sniffer.Sniffer.get_flow (
              self )
```

```
Reads and returns stdout data from the 'ra' subprocess (for .pcap files or raw tcpdump
network data), or by reading the lines in the pre-processed file handle.

Also sanitises the flow that it receivies (such as filling in empty csv fields).

Returns:
    sniffed_flow (string): A sniffed network flow string; from the stdout of the 'ra' subprocess, or a line fr
```

4.4.3.2 start()

```
def detection_engine_modules.Sniffer.Sniffer.start (
              self )
```

```
Starts the required subprocesses for the sniffer.

Operation depends on the sniffer object's attribute values (i.e. the mode that it is to run in).

Three main modes:
    - Network sniffer and process mode
    - .pcap file read and process mode
    - Pre-processed network flow file read mode

Returns:
    started (bool): Status of the sniffer's starting operation.
```

The documentation for this class was generated from the following file:

- /home/asmarus/Desktop/botnetdetectionengine/detection_engine_modules/Sniffer.py

4.5 detection_engine_modules.Websocket_Client.Websocket_Client Class Reference

Public Member Functions

- def **__init__** (self)
- def **__del__** (self)
- def **connect** (self, socket_addr)
- def **send** (self, labelled_flow, alert=None)
- def **attempt_reconnect** (self)

Public Attributes

- **socket_addr**
- **socket**

4.5.1 Detailed Description

The Websocket Client class controls the connection and data transfer to the Websocket Server.

This class will attempt to reconnect the instance to the server in the case of a failed data send attempt.

If reconnect fails, an exception is raised.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 __init__()

def detection_engine_modules.Websocket_Client.Websocket_Client.__init__ (
self)

The constructor of the Websocket_Client object.

Attributes:
socket_addr (string): The address of the websocket server.
socket (False or :obj:'WebSocket'): The websocket's connection socket storage variable.

4.5.2.2 __del__()

def detection_engine_modules.Websocket_Client.Websocket_Client.__del__ (
self)

The Websocket_Client object's destructor.

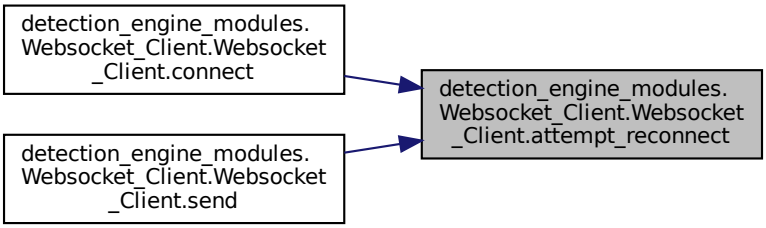
Explicitally closes a running socket.

4.5.3 Member Function Documentation

4.5.3.1 attempt_reconnect()

```
def detection_engine_modules.Websocket_Client.Websocket_Client.attempt_reconnect (
              self )

Attempts to re-establish the Websocket connection to the server.

Makes a maximum of 5 reconnect attempts. If reconnect is successful, the function returns a new socket connnec
else, an exception is raised.

Returns:
    socket (:obj:'WebSocket'): The WebSocket's connection object, in the case of a successful reconnect attemp
```

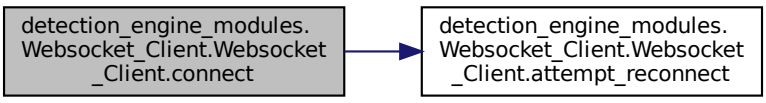
Here is the caller graph for this function:



4.5.3.2 connect()

```
def detection_engine_modules.Websocket_Client.Websocket_Client.connect (
              self,
              socket_addr )

This function attempt to create a conneciton to a websocket server.

If a connection cannot be created, it also attempts the reconnect functionality.

Args:
    socket_addr (string): An address of the websocket server that it intends to attempt a connection with.

Returns:
    (bool): True if it successfully creates a connection.
```

Here is the call graph for this function:

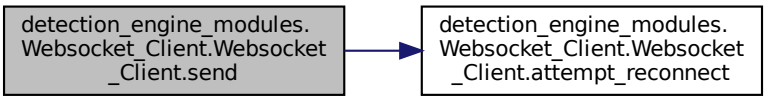


4.5.3.3 send()

```
def detection_engine_modules.Websocket_Client.Websocket_Client.send (
              self,
              labelled_flow,
              alert = None )
```

```
Attempts to send the labelled_flow and alert_data as a JSON data structure through the active websocket connec

Attempts to reconnect if the data send via the socket fails.

Args:
    labelled_flow (string): The labelled flow data string.
    alert (None or json-formatted string): The alert data, if required to be sent.

Returns:
    (bool): True, if the data is sent successfully.
```

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- /home/asmarus/Desktop/botnetdetectionengine/detection_engine_modules/Websocket_Client.py