

---

# **AntiNex Network Pipeline Documentation**

***Release 1.0.0***

**Jay Johnson**

**Nov 02, 2018**



---

## Contents

---

<b>1</b>	<b>AntiNex Stack Status</b>	<b>1</b>
<b>2</b>	<b>Table of Contents</b>	<b>3</b>
2.1	Source Code . . . . .	3
<b>3</b>	<b>Indices and tables</b>	<b>13</b>
	<b>Python Module Index</b>	<b>15</b>



# CHAPTER 1

---

## AntiNex Stack Status

---

AntiNex Network Pipeline is part of the AntiNex stack:

Component	Build	Docs Link	Docs Build
<a href="#">REST API</a>		<a href="#">Docs</a>	
<a href="#">Core Worker</a>		<a href="#">Docs</a>	
<a href="#">Network Pipeline</a>		<a href="#">Docs</a>	
<a href="#">AI Utils</a>		<a href="#">Docs</a>	
<a href="#">Client</a>		<a href="#">Docs</a>	



These are the docs for the AntiNex Network Pipeline repository.

## 2.1 Source Code

### 2.1.1 Handle Packets from a Network Interface

This is the default handler for processing network packets received from the network interface with `eth0` or `eth1`. In production, this is the starting point for making live predictions with the AntiNex REST API.

Here is the workflow for processing a network packet from a monitored interface:

1. Get Available Layers in the Packet
2. Convert the Packet to a JSON dictionary
3. Publish the Message using Kombu with environment values setting the routing decision for the message in the aggregation message broker: `FORWARD_EXCHANGE`, `FORWARD_ROUTING_KEY`, `FORWARD_QUEUE`.

`network_pipeline.handle_packets.handle_packets(pk)`

**Parameters** `pk` – data packet that kamene sends in

### 2.1.2 Process Consumed Messages from the Queue

This is the default handler for processing messages consumed from the aggregation message broker. At the conceptual level, all network interface capture tools forward JSON dictionaries to this class.

**class** `network_pipeline.record_packets_to_csv.RecordPacketsToCSV`

`build_all_keys_dict()`

`build_flat_msg(id=None, msg=None)`

#### Parameters

- **id** – unique id for this message
- **msg** – message dictionary to flatten

**convert\_to\_df** ()

**create\_json\_archive** ()

**flatten\_all** ()

**handle\_msg** (*body, org\_message*)

#### Parameters

- **body** – dictionary contents from the message body
- **org\_message** – message object can ack, requeue or reject

**process\_arp\_frame** (*id=None, msg=None*)

Convert a complex nested json dictionary to a flattened dictionary and capture all unique keys for table construction

#### Parameters

- **id** – key for this msg
- **msg** – arp frame for packet

**process\_dns\_frame** (*id=None, msg=None*)

Convert a complex nested json dictionary to a flattened dictionary and capture all unique keys for table construction

#### Parameters

- **id** – key for this msg
- **msg** – dns frame for packet

**process\_ether\_frame** (*id=None, msg=None*)

Convert a complex nested json dictionary to a flattened dictionary and capture all unique keys for table construction

#### Parameters

- **id** – key for this msg
- **msg** – ether frame for packet

**process\_icmp\_frame** (*id=None, msg=None*)

Convert a complex nested json dictionary to a flattened dictionary and capture all unique keys for table construction

#### Parameters

- **id** – key for this msg
- **msg** – icmp frame for packet

**process\_ip\_frame** (*id=None, msg=None*)

Convert a complex nested json dictionary to a flattened dictionary and capture all unique keys for table construction

#### Parameters

- **id** – key for this msg



- **msg** – ip frame for packet

**process\_ipvsix\_frame** (*id=None, msg=None*)

Convert a complex nested json dictionary to a flattened dictionary and capture all unique keys for table construction

**Parameters**

- **id** – key for this msg
- **msg** – ipv6 frame for packet

**process\_pad\_frame** (*id=None, msg=None*)

Convert a complex nested json dictionary to a flattened dictionary and capture all unique keys for table construction

**Parameters**

- **id** – key for this msg
- **msg** – pad frame for packet

**process\_raw\_frame** (*id=None, msg=None*)

Convert a complex nested json dictionary to a flattened dictionary and capture all unique keys for table construction

**Parameters**

- **id** – key for this msg
- **msg** – raw frame for packet

**process\_tcp\_frame** (*id=None, msg=None*)

Convert a complex nested json dictionary to a flattened dictionary and capture all unique keys for table construction

**Parameters**

- **id** – key for this msg
- **msg** – tcp frame for packet

**process\_udp\_frame** (*id=None, msg=None*)

Convert a complex nested json dictionary to a flattened dictionary and capture all unique keys for table construction

**Parameters**

- **id** – key for this msg
- **msg** – udp frame for packet

**publish\_predictions\_to\_core** ()

**save\_data** ()

**save\_df\_as\_csv** ()

**write\_to\_file** (*data\_dict, output\_file\_path*)

**Parameters**

- **data\_dict** –
- **output\_file\_path** –

### 2.1.3 Network Pipeline Internal Modules

`network_pipeline.build_packet_key.build_packet_key()`

`network_pipeline.connect_forwarder.connect_forwarder` (*forward\_host=None,*  
*forward\_port=None,*  
*max\_retries=-1,*  
*sleep\_interval=1.0*)

**Parameters**

- **forward\_host** – host for receiving forwarded packets
- **forward\_port** – port for the forwarded packets
- **max\_retries** – retries, -1 = infinite
- **sleep\_interval** – how often to retry in this loop

`network_pipeline.convert_pkt_to_json.convert_pkt_to_json` (*pkg*)

Inspired by: [https://gist.github.com/cr0hn/1b0c2e672cd0721d3a07/raw/9144676ceb12dbd545e6dce366822bbedde8de2c/pkg\\_to\\_json.py](https://gist.github.com/cr0hn/1b0c2e672cd0721d3a07/raw/9144676ceb12dbd545e6dce366822bbedde8de2c/pkg_to_json.py) This function convert a Scapy packet to JSON

**Parameters** *pkg* (*objects*) – A kamene package

**Returns** A JSON data

**Return type** dict()

`network_pipeline.create_layer_2_socket.create_layer_2_socket()`

`network_pipeline.parse_network_data.eth_addr` (*f*)

**Parameters** *f* – eth frame

`network_pipeline.parse_network_data.unshift_flags` (*tcp\_flags*)

De-shift the TCP flags to a string repr

`network_pipeline.parse_network_data.build_key()`

`network_pipeline.parse_network_data.parse_network_data` (*data\_packet=None,* *in-*  
*clude\_filter\_key=None,*  
*filter\_keys=[],*  
*record\_tcp=True,*  
*record\_udp=True,*  
*record\_arp=True,*  
*record\_icmp=True*)

`build_node`

**Parameters**

- **data\_packet** – raw recvfrom data
- **filter\_keys** – list of strings to filter and remove baby-birding packets to yourself
- **record\_tcp** – want to record TCP frames?
- **record\_udp** – want to record UDP frames?
- **record\_arp** – want to record ARP frames?
- **record\_icmp** – want to record ICMP frames?

`network_pipeline.publisher.get_publisher()`

`network_pipeline.utils.rnow` (*f='%Y-%m-%d %H:%M:%S'*)

**Parameters** *f* – format for the string

`network_pipeline.utils.ppj` (*json\_data*)

**Parameters** *json\_data* – dictionary to print

`network_pipeline.start_consumers_for_queue.start_consumers_for_queue` (*prefix\_name*='worker',  
*num\_workers*=2,  
*tasks*=None,  
*queue\_to\_consume*=None,  
*shutdown\_msg*='SHUTDOWN',  
*consumer\_class*=None,  
*need\_response*=False,  
*callback*=None)

**Parameters**

- *prefix\_name* –
- *num\_workers* –
- *tasks* –
- *queue\_to\_consume* –
- *shutdown\_msg* –
- *consumer\_class* –
- *need\_response* –
- *callback* –

`class network_pipeline.network_packet_task.NetworkPacketTask` (*source*='localhost',  
*payload*=None)

`network_pipeline.shutdown_consumers.shutdown_consumers` (*num\_workers*=2,  
*tasks*=None, *shutdown\_msg*='SHUTDOWN')

**Parameters**

- *num\_workers* –
- *tasks* –
- *shutdown\_msg* –

`class network_pipeline.simulated_work_task.SimulatedWorkTask` (*a*, *b*)

`class network_pipeline.worker_to_process_packets.WorkerToProcessPackets` (*name*,  
*task\_queue*,  
*re-sult\_queue*,  
*shutdown\_msg*='SHUTDOWN',  
*need\_response*=False,  
*callback*=None)

`run()`

## 2.1.4 Network Pipeline Scripts

### Capture Agents

Here are the AntiNex Network Pipeline Capture Agents. These scripts allow for capturing traffic on a network device and flattening it into JSON dictionaries before publishing to the aggregation message broker. Please refer to the `handle_packets` method for more details.

**Warning:** These tools will capture network traffic. Please be careful where you deploy them.

### ARP

```
network_pipeline.scripts.capture_arp.capture_arp_packets()
```

Capture ARP packets and call the `handle_packets` method

Change the network interface by `export CAP_DEVICE=eth0`

### ICMP

```
network_pipeline.scripts.capture_icmp.capture_icmp_packets()
```

Capture ICMP packets and call the `handle_packets` method

Change the network interface by `export CAP_DEVICE=eth0`

### TCP

```
network_pipeline.scripts.capture_ssh.capture_tcp_packets_over_ssh()
```

Capture TCP packets over ssh and call the `handle_packets` method

Change the network interface by `export CAP_DEVICE=eth0`

```
network_pipeline.scripts.capture_tcp.capture_tcp_packets()
```

Capture TCP packets and call the `handle_packets` method

Change the network interface by `export CAP_DEVICE=eth0`

```
network_pipeline.scripts.capture_telnet.capture_tcp_packets_over_telnet()
```

Capture TCP packets over telnet and call the `handle_packets` method

Change the network interface by `export CAP_DEVICE=eth0`

### UDP

```
network_pipeline.scripts.capture_udp.capture_udp_packets()
```

Capture UDP packets and call the `handle_packets` method

Change the network interface by `export CAP_DEVICE=eth0`

## Publishers

These tools are designed to show how to save captured packet dictionaries to CSVs and how to publish them for live predictions using a pre-trained Deep Neural Network.

`network_pipeline.scripts.packets_rabbitmq.recv_msg(body, message)`

Handler method - fires when a messages is consumed from the FORWARD\_QUEUE queue running in the FORWARD\_BROKER\_URL broker.

### Parameters

- **body** – message body
- **message** – message object can ack, requeue or reject

`network_pipeline.scripts.packets_rabbitmq.consume_network_packet_messages_from_rabbitmq()`

Setup a celery\_connectors.KombuSubscriber to consume meessages from the FORWARD\_BROKER\_URL broker in the FORWARD\_QUEUE queue.

`network_pipeline.scripts.packets_redis.recv_msg(body, message)`

Handler method - fires when a messages is consumed from the FORWARD\_QUEUE queue running in the FORWARD\_BROKER\_URL broker.

### Parameters

- **body** – message body
- **message** – message object can ack, requeue or reject

`network_pipeline.scripts.packets_redis.consume_network_packet_messages_from_redis()`

Setup a celery\_connectors.KombuSubscriber to consume meessages from the FORWARD\_BROKER\_URL broker in the FORWARD\_QUEUE queue.

## Test Tools

These will send mock traffic data to the targeted network device.

`network_pipeline.scripts.base_capture.example_capture()`

An example capture script

Change the network interface by `export CAP_DEVICE=eth0`

`network_pipeline.scripts.arp_send_msg.send_arp_msg()`

Send an ARP message to the network device (enp0s3 by default).

`network_pipeline.scripts.tcp_send_large_msg.send_tcp_large_message()`

Send a large TCP message to port 80 by default.

`network_pipeline.scripts.tcp_send_msg.send_tcp_message()`

Send a TCP message to port 80 by default.

`network_pipeline.scripts.udp_send_msg.send_udp_message()`

Send a UDP message to port 80 by default.

Environment variables:

UDP\_SEND\_TO\_HOST - host ip address UDP\_SEND\_TO\_PORT - send to this UDP port

`network_pipeline.scripts.listen_tcp_port.listen_on_tcp_port()`

Run a simple server for processing messages over TCP.

LISTEN\_ON\_HOST - listen on this host ip address

LISTEN\_ON\_PORT - listen on this TCP port

LISTEN\_SIZE - listen on to packets of this size

LISTEN\_SLEEP - sleep this number of seconds per loop

LISTEN\_SHUTDOWN\_HOOK - shutdown if file is found on disk

`network_pipeline.scripts.listen_udp_port.listen_on_udp_port()`

Run a simple server for processing messages over UDP.

UDP\_LISTEN\_ON\_HOST - listen on this host ip address

UDP\_LISTEN\_ON\_PORT - listen on this UDP port

UDP\_LISTEN\_SIZE - listen on to packets of this size

UDP\_LISTEN\_SLEEP - sleep this number of seconds per loop

UDP\_LISTEN\_SHUTDOWN\_HOOK - shutdown if file is found on disk

`network_pipeline.scripts.builders.prepare_dataset.find_all_headers` (*pipeline\_files=[]*,  
*la-*  
*bel\_rules=None*)

#### Parameters

- **pipeline\_files** – files to process
- **label\_rules** – labeling rules

`network_pipeline.scripts.builders.prepare_dataset.build_csv` (*pipeline\_files=[]*,  
*fulldata\_file=None*,  
*clean\_file=None*,  
*post\_proc\_rules=None*,  
*label\_rules=None*,  
*meta-*  
*data\_filename='metadata.json'*)

#### Parameters

- **pipeline\_files** – files to process
- **fulldata\_file** – output all columns to this csv file
- **clean\_file** – output all numeric-ready columns to this csv file
- **post\_proc\_rules** – rules after building the DataFrame
- **label\_rules** – labeling rules
- **metadata\_filename** – metadata

`network_pipeline.scripts.builders.prepare_dataset.find_all_pipeline_csvs` (*csv\_glob\_path='/opt/anti-*

**Parameters** **csv\_glob\_path** – path to csvs

`network_pipeline.scripts.builders.prepare_dataset.prepare_new_dataset()`

**class** `network_pipeline.scripts.tools.arp_send_msg.Ethernet`  
 Generic Ethernet Frame class

**class** `network_pipeline.scripts.tools.arp_send_msg.Arp`  
 Generic ARP Frame class

## 2.1.5 Constants

```

VALID = 0
FILTERED = 1
INVALID = 2
ERROR = 3
UNSUPPORTED = 4
ETH_UNSUPPORTED = 5
IP_UNSUPPORTED = 6

INCLUDED_IGNORE_KEY = "CHANGE_TO_YOUR_OWN_KEY"

ETH_HEADER_FORMAT = "!6s6sH"
IP_HEADER_FORMAT = "!BBHHBBH4s4s"
TCP_HEADER_FORMAT = "!HLLBBHH"
TCP_PSH_FORMAT = "!4s4sBBH"
UDP_HEADER_FORMAT = "!HHHH"
ICMP_HEADER_FORMAT = "!BBH"
ARP_HEADER_FORMAT = "2s2s1s1s2s6s4s6s4s"

SIZE_ETH_HEADER = struct.calcsize(ETH_HEADER_FORMAT)
SIZE_IP_HEADER = struct.calcsize(IP_HEADER_FORMAT)
SIZE_TCP_HEADER = struct.calcsize(TCP_HEADER_FORMAT)
SIZE_UDP_HEADER = struct.calcsize(UDP_HEADER_FORMAT)
SIZE_ICMP_HEADER = struct.calcsize(ICMP_HEADER_FORMAT)
SIZE_ARP_HEADER = struct.calcsize(ARP_HEADER_FORMAT)

UNKNOWN = 0
TCP = 1
UDP = 2
ICMP = 3
ARP = 4

ARP_PROTO_ETH = 9731
ICMP_PROTO_IP = 1
IP_PROTO_ETH = 8
TCP_PROTO_IP = 6
UDP_PROTO_IP = 17

IGNORED_REDIS_PORTS = [6379, 16379]
IGNORED_RABBITMQ_PORTS = [5672, 15672, 25672]

```

## 2.1.6 Environment Variables

```

SOURCE = os.getenv(
    "SOURCE_HOST",
    "localdev").strip().lstrip()
FORWARD_BROKER_URL = os.getenv(
    "FORWARD_BROKER_URL",
    "redis://localhost:6379/15").strip().lstrip()
FORWARD_SSL_OPTIONS = json.loads(os.getenv(
    "FORWARD_SSL_OPTIONS",
    "{}").strip().lstrip())
FORWARD_ENDPOINT_TYPE = os.getenv(
    "FORMAT_ET",

```

(continues on next page)

(continued from previous page)

```
        "redis").strip().strip()
FORWARD_EXCHANGE = os.getenv(
    "FORWARD_EXCHANGE",
    "NEW_PACKETS").strip().lstrip()
FORWARD_ROUTING_KEY = os.getenv(
    "FORWARD_ROUTING_KEY",
    "NEW_PACKETS").strip().lstrip()
FORWARD_QUEUE = os.getenv(
    "FORWARD_QUEUE",
    "NEW_PACKETS").strip().lstrip()
DEBUG_PACKETS = bool(os.getenv(
    "DEBUG_PACKETS",
    "0").strip().lstrip() == "1")
```



## CHAPTER 3

---

### Indices and tables

---

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



---

## Python Module Index

---

### n

network\_pipeline.build\_packet\_key, 6  
network\_pipeline.connect\_forwarder, 6  
network\_pipeline.convert\_pkt\_to\_json, 6  
network\_pipeline.create\_layer\_2\_socket, 6  
network\_pipeline.handle\_packets, 3  
network\_pipeline.network\_packet\_task, 7  
network\_pipeline.parse\_network\_data, 6  
network\_pipeline.publisher, 6  
network\_pipeline.record\_packets\_to\_csv, 3  
network\_pipeline.scripts.arp\_send\_msg, 9  
network\_pipeline.scripts.base\_capture, 9  
network\_pipeline.scripts.builders.prepare\_dataset, 10  
network\_pipeline.scripts.capture\_arp, 8  
network\_pipeline.scripts.capture\_icmp, 8  
network\_pipeline.scripts.capture\_ssh, 8  
network\_pipeline.scripts.capture\_tcp, 8  
network\_pipeline.scripts.capture\_telnet, 8  
network\_pipeline.scripts.capture\_udp, 8  
network\_pipeline.scripts.listen\_tcp\_port, 9  
network\_pipeline.scripts.listen\_udp\_port, 10  
network\_pipeline.scripts.packets\_rabbitmq, 9  
network\_pipeline.scripts.packets\_redis, 9  
network\_pipeline.scripts.tcp\_send\_large\_msg, 9  
network\_pipeline.scripts.tcp\_send\_msg, 9  
network\_pipeline.scripts.tools.arp\_send\_msg, 10  
network\_pipeline.scripts.udp\_send\_msg, 9  
network\_pipeline.shutdown\_consumers, 7  
network\_pipeline.simulated\_work\_task, 7  
network\_pipeline.start\_consumers\_for\_queue, 7  
network\_pipeline.utils, 6  
network\_pipeline.worker\_to\_process\_packets, 7



## A

Arp (class in network\_pipeline.scripts.tools.arp\_send\_msg), 10

## B

build\_all\_keys\_dict() (network\_pipeline.record\_packets\_to\_csv.RecordPacketsToCSV method), 3

build\_csv() (in module network\_pipeline.scripts.builders.prepare\_dataset), 10

build\_flat\_msg() (network\_pipeline.record\_packets\_to\_csv.RecordPacketsToCSV method), 3

build\_key() (in module network\_pipeline.parse\_network\_data), 6

build\_packet\_key() (in module network\_pipeline.build\_packet\_key), 6

## C

capture\_arp\_packets() (in module network\_pipeline.scripts.capture\_arp), 8

capture\_icmp\_packets() (in module network\_pipeline.scripts.capture\_icmp), 8

capture\_tcp\_packets() (in module network\_pipeline.scripts.capture\_tcp), 8

capture\_tcp\_packets\_over\_ssh() (in module network\_pipeline.scripts.capture\_ssh), 8

capture\_tcp\_packets\_over\_telnet() (in module network\_pipeline.scripts.capture\_telnet), 8

capture\_udp\_packets() (in module network\_pipeline.scripts.capture\_udp), 8

connect\_forwarder() (in module network\_pipeline.connect\_forwarder), 6

consume\_network\_packet\_messages\_from\_rabbitmq() (in module network\_pipeline.scripts.packets\_rabbitmq), 9

consume\_network\_packet\_messages\_from\_redis() (in module network\_pipeline.scripts.packets\_redis), 9

convert\_pkt\_to\_json() (in module network\_pipeline.convert\_pkt\_to\_json), 6

convert\_to\_df() (network\_pipeline.record\_packets\_to\_csv.RecordPacketsToCSV method), 4

create\_json\_archive() (network\_pipeline.record\_packets\_to\_csv.RecordPacketsToCSV method), 4

create\_layer\_2\_socket() (in module network\_pipeline.create\_layer\_2\_socket), 6

## E

ExamplePacketsToCSV (in module network\_pipeline.parse\_network\_data), 6

Ethernet (class in network\_pipeline.scripts.tools.arp\_send\_msg), 10

example\_capture() (in module network\_pipeline.scripts.base\_capture), 9

## F

find\_all\_headers() (in module network\_pipeline.scripts.builders.prepare\_dataset), 10

find\_all\_pipeline\_csvs() (in module network\_pipeline.scripts.builders.prepare\_dataset), 10

flatten\_all() (network\_pipeline.record\_packets\_to\_csv.RecordPacketsToCSV method), 4

## G

get\_publisher() (in module network\_pipeline.publisher), 6

## H

handle\_msg() (network\_pipeline.record\_packets\_to\_csv.RecordPacketsToCSV method), 4

handle\_packets() (in module network\_pipeline.handle\_packets), 3

## L

`listen_on_tcp_port()` (in module `net-work_pipeline.scripts.listen_tcp_port`), 9

`listen_on_udp_port()` (in module `net-work_pipeline.scripts.listen_udp_port`), 10

## N

`network_pipeline.build_packet_key` (module), 6

`network_pipeline.connect_forwarder` (module), 6

`network_pipeline.convert_pkt_to_json` (module), 6

`network_pipeline.create_layer_2_socket` (module), 6

`network_pipeline.handle_packets` (module), 3

`network_pipeline.network_packet_task` (module), 7

`network_pipeline.parse_network_data` (module), 6

`network_pipeline.publisher` (module), 6

`network_pipeline.record_packets_to_csv` (module), 3

`network_pipeline.scripts.arp_send_msg` (module), 9

`network_pipeline.scripts.base_capture` (module), 9

`network_pipeline.scripts.builders.prepare_dataset` (module), 10

`network_pipeline.scripts.capture_arp` (module), 8

`network_pipeline.scripts.capture_icmp` (module), 8

`network_pipeline.scripts.capture_ssh` (module), 8

`network_pipeline.scripts.capture_tcp` (module), 8

`network_pipeline.scripts.capture_telnet` (module), 8

`network_pipeline.scripts.capture_udp` (module), 8

`network_pipeline.scripts.listen_tcp_port` (module), 9

`network_pipeline.scripts.listen_udp_port` (module), 10

`network_pipeline.scripts.packets_rabbitmq` (module), 9

`network_pipeline.scripts.packets_redis` (module), 9

`network_pipeline.scripts.tcp_send_large_msg` (module), 9

`network_pipeline.scripts.tcp_send_msg` (module), 9

`network_pipeline.scripts.tools.arp_send_msg` (module), 10

`network_pipeline.scripts.udp_send_msg` (module), 9

`network_pipeline.shutdown_consumers` (module), 7

`network_pipeline.simulated_work_task` (module), 7

`network_pipeline.start_consumers_for_queue` (module), 7

`network_pipeline.utils` (module), 6

`network_pipeline.worker_to_process_packets` (module), 7

`NetworkPacketTask` (class in `network_pipeline.network_packet_task`), 7

## P

`parse_network_data()` (in module `network_pipeline.parse_network_data`), 6

`ppj()` (in module `network_pipeline.utils`), 7

`prepare_new_dataset()` (in module `network_pipeline.scripts.builders.prepare_dataset`), 10

`process_arp_frame()` (in module `network_pipeline.record_packets_to_csv.RecordPacketsToCSV` method), 4

`process_dns_frame()` (in module `network_pipeline.record_packets_to_csv.RecordPacketsToCSV` method), 4

`process_ether_frame()` (in module `network_pipeline.record_packets_to_csv.RecordPacketsToCSV` method), 4

`process_icmp_frame()` (in module `network_pipeline.record_packets_to_csv.RecordPacketsToCSV` method), 4

`process_ip_frame()` (in module `network_pipeline.record_packets_to_csv.RecordPacketsToCSV` method), 4

`process_ipv6_frame()` (in module `network_pipeline.record_packets_to_csv.RecordPacketsToCSV` method), 5

`process_pad_frame()` (in module `network_pipeline.record_packets_to_csv.RecordPacketsToCSV` method), 5

`process_raw_frame()` (in module `network_pipeline.record_packets_to_csv.RecordPacketsToCSV` method), 5

`process_tcp_frame()` (in module `network_pipeline.record_packets_to_csv.RecordPacketsToCSV` method), 5

`process_udp_frame()` (in module `network_pipeline.record_packets_to_csv.RecordPacketsToCSV` method), 5

`publish_predictions_to_core()` (in module `network_pipeline.record_packets_to_csv.RecordPacketsToCSV` method), 5

## R

`RecordPacketsToCSV` (class in module `network_pipeline.record_packets_to_csv`), 3

`recv_msg()` (in module `network_pipeline.scripts.packets_rabbitmq`), 9

`recv_msg()` (in module `network_pipeline.scripts.packets_redis`), 9

`rnow()` (in module `network_pipeline.utils`), 6

`run()` (`network_pipeline.worker_to_process_packets.WorkerToProcessPackets` method), 7

## S

`save_data()` (`network_pipeline.record_packets_to_csv.RecordPacketsToCSV` method), 5

`save_df_as_csv()` (`network_pipeline.record_packets_to_csv.RecordPacketsToCSV` method), 5

`send_arp_msg()` (in module `network_pipeline.scripts.arp_send_msg`), 9

`send_tcp_large_message()` (in module `network_pipeline.scripts.tcp_send_large_msg`),  
9

`send_tcp_message()` (in module `network_pipeline.scripts.tcp_send_msg`), 9

`send_udp_message()` (in module `network_pipeline.scripts.udp_send_msg`), 9

`shutdown_consumers()` (in module `network_pipeline.shutdown_consumers`), 7

`SimulatedWorkTask` (class in `network_pipeline.simulated_work_task`), 7

`start_consumers_for_queue()` (in module `network_pipeline.start_consumers_for_queue`),  
7

## U

`unshift_flags()` (in module `network_pipeline.parse_network_data`), 6

## W

`WorkerToProcessPackets` (class in `network_pipeline.worker_to_process_packets`),  
7

`write_to_file()` (`network_pipeline.record_packets_to_csv.RecordPacketsToCSV`  
method), 5