smudge Documentation

David Quartarolo

Tue Jan 3 15:14:38 EST 2023

Contents

Color text is not available on this platform. Color text is not available on this platform. — description: | API documentation for modules: **init**, **main**, **version**, command_line, utils, test_passive, test_signature_matching.

lang: en

classoption: oneside geometry: margin=1in papersize: a4

linkcolor: blue links-as-notes: true …

Module `__init__`

Smudge init file.

Module `__main__`

Module `__version__`

SMUDGE module listing versions.

Module `command_line`

Command Line Utils

Classes

Class `CommandLineUtility`

```
class CommandLineUtility
```

The utility methods needed for the command line tool reside here.

Static methods

Method calculate_dist

```
def calculate_dist(
    time_to_live
)
```

Takes a value of TTL and calculates hop distance.

Method cprint

```
def cprint(
    out
)
```

If enabled, prints main colored output.

Method handle_packet

```
def handle_packet(
    packet
)
```

Read packet capture from file and SMUDGE.

Method import_signatures

```
def import_signatures(
    argument
)
```

Import Signatures from Github

Method list_interfaces

```
def list_interfaces(
    argument
)
```

List all available Network Interfaces.

Method mprint

```
def mprint(
    out
)
```

If enabled, prints secondary colored output.

Method read_pcap

```
def read_pcap(
    args
)
```

Read packet capture from file and SMUDGE.

Method verify_interface

```
def verify_interface(
    interface
)
```

Verify interface argument is valid.

Method verify_pause

```
def verify_pause(
    flo
)
```

Verify pause argument is between 0 and 1.

Method verify_signatures

```
def verify_signatures()
```

Verifies that signature db file exists.

Module utils

Signature Matching

Classes

Class Matching

```
class Matching
```

This class be matching.

Static methods

Method create_con

```
def create_con()
```

Create Database Connection

Method match

```
def match(
    signature_obj
)
```

Match.

Method sig_match_eighty

```
def sig_match_eighty(
    conn,
    signature_options
)
```

Select 80%

Method sig_match_fourty

```
def sig_match_fourty(
    conn,
    signature_options
)
```

Select 40%

Method sig_match_one

```
def sig_match_one(
    conn,
    sig_obj
)
```

Select 100%

Method sig_match_sixty

```
def sig_match_sixty(
    conn,
    signature_options
)
```

Select 60%

Method sig_match_twenty

```
def sig_match_twenty(
    conn,
    signature_options
)
```

Select 20%

Class `PassiveData`

```
class PassiveData
```

A class filled with static methods that interacts with the sqlite database.

Static methods

`Method create_con`

```
def create_con()
```

Create Database Connection

`Method setup_db`

```
def setup_db()
```

Create Sqlite3 DB with all required tables

`Method signature_insert`

```
def signature_insert(
    conn,
    sig_obj
)
```

Insert Statement for the Signature Table.

`Method test_github_con`

```
def test_github_con()
```

Tests Internet Connection to Github.com

Class `PullData`

```
class PullData
```

A class that contains a method that: * Loads a json file from github into memory. * Dumps the json into the sqlite database.

The use of class methods is used so that class variables can be overrided for testing. …

Class Variables url : str URL of raw json file that contains TCP Signatures.

Class variables

Variable `url`

Static methods

`Method import_data`

```
def import_data()
```

Imports TCP Signatures from raw JSON file hosted on Github.

`Method import_local_data`

```
def import_local_data(
    json_file
)
```

Imports TCP Signatures from local raw JSON file.

Class `QueryObject`

```
class QueryObject(
    acid,
    platform,
    tcp_flag,
    comments,
    version,
    ittl,
    olen,
    mss,
    wsize,
    scale,
    olayout,
    quirks,
    pclass
)
```

Data mapping class that takes a TCP Signature object and inserts it into the sqlite database.

Instance variables

Variable `qstring` Query String.

Class `Quirk`

```
class Quirk(
    packet
)
```

Creates quirks - comma-delimited properties and quirks observed in IP or TCP headers. If a signature scoped to both IPv4 and IPv6 contains quirks valid for just one of these protocols, such quirks will be ignored for on packets using the other protocol. For example, any combination of 'df', 'id+', and 'id-' is always matched by any IPv6 packet.

Takes a packet as an argument.

Instance variables

Variable `ack_minus` Sets ack- - ACK number is zero, but ACK flag set.

Variable `ack_plus` Sets ack+ - ACK number is non-zero, but ACK flag not set.

Variable `bad` Sets bad attribute - malformed TCP options.

Variable `df_flag` Sets df attribute based on flag - "don't fragment" set (probably PMTUD); ignored for IPv6.

Variable `ecn` Sets ecn attribute - explicit congestion notification support.

Variable `exws` Sets exws attribute - excessive window scaling factor (> 14).

Variable `flow` Sets flow Attribute - non-zero IPv6 flow ID; ignored for IPv4.

Variable `id_minus` Sets id- attribute based on flag and IPID - DF not set but IPID is zero; ignored for IPv6.

Variable `id_plus` Sets id+ attribute based on flag and IPID - DF set but IPID non-zero; ignored for IPv6.

Variable `opt_plus` Sets opt+ attribute - trailing non-zero data in options segment.

Variable `pushf_plus` Sets pushf+ attribute - PUSH flag used.

Variable `qstring` Looks at all attributes and makes quirks.

Variable `seq_minus` Sets seq- attribute - sequence number is zero.

Variable `ts1_minus` Sets ts1- attribute - own timestamp specified as zero.

Variable `ts2_plus` Sets ts2+ attribute - non-zero peer timestamp on initial SYN.

Variable `uptr_plus` Sets uptr+ attribute - URG pointer is non-zero, but URG flag not set.

Variable `urgf_plus` Sets urgf+ attribute - URG flag used.

Variable `zero_plus` Sets 0+ Attribute - "must be zero" field not zero; ignored for IPv6.

Class `Signature`

```
class Signature(
    packet
)
```

Data mapping class that takes a TCP Signature object and inserts it into the sqlite database.

Instance variables

Variable `ittl` Initial TTL used by the OS. Almost all operating systems use 64, 128, or 255; ancient versions of Windows sometimes used 32, and several obscure systems sometimes resort to odd values such as 60.

NEW SIGNATURES: P0f will usually suggest something, using the format of 'observed_ttl+distance' (e.g. 54+10). Consider using traceroute to check that the distance is accurate, then sum up the values. If initial TTL can't be guessed, p0f will output 'nnn+?', and you need to use traceroute to estimate the '?'.

A handful of userspace tools will generate random TTLs. In these cases, determine maximum initial TTL and then add a - suffix to the value to avoid confusion.

Variable `mss` maximum segment size, if specified in TCP options. Special value of '*' can be used to denote that MSS varies depending on the parameters of sender's network link, and should not be a part of the signature. In this case, MSS will be used to guess the type of network hookup according to the [mtu] rules.

NEW SIGNATURES: Use '*' for any commodity OSes where MSS is around 1300 - 1500, unless you know for sure that it's fixed. If the value is outside that range, you can probably copy it literally.

Variable `olayout` comma-delimited layout and ordering of TCP options, if any. This is one of the most valuable TCP fingerprinting signals. Supported values.

Variable `olen` Length of IPv4 options or IPv6 extension headers. Usually zero for normal IPv4 traffic; always zero for IPv6 due to the limitations of libpcap.

Variable `pclass` Payload size classification: '0' for zero, '+' for non-zero, '*' for any. The packets we fingerprint right now normally have no payloads, but some corner cases exist.

Variable `qstring` Create Query String

Variable `quirk` Comma-delimited properties and quirks observed in IP or TCP headers.

Variable `scale` Window scaling factor, if specified in TCP options. Fixed value or '*'. NEW SIGNATURES: Copy literally, unless the value varies randomly. Many systems alter between 2 or 3 scaling factors, in which case, it's better to have several 'sig' lines, rather than a wildcard.

Variable `version` Signature for IPv4 ('4'), IPv6 ('6'), or both ('*').

Variable `window_size` Window size. Can be expressed as a fixed value, but many operating systems set it to a multiple of MSS or MTU, or a multiple of some random integer. P0f automatically detects these cases, and allows notation such as 'mss*4', 'mtu*4', or '%8192' to be used. Wilcard ('*') is possible too.

Static methods

`Method process_options`

```
def process_options(
    option: list
) -> str
```

Static method for processing options.

Class `TcpSig`

```
class TcpSig(
    tcp_sig_obj
)
```

Data mapping class that takes a TCP Signature object and inserts it into the sqlite database.

Instance variables

Variable `qstring` QString.

Module `test_passive`

Pytest Module for Passive Data

Functions

Function `test_setup_db_1`

```
def test_setup_db_1()
```

UT to ensure DB file gets created.

Function `test_setup_db_5`

```
def test_setup_db_5()
```

UT to ensure signatures table was created.

Function `test_test_github_con_1`

```
def test_test_github_con_1()
```

UT to ensure Github connection test is successful.

Module `test_signature_matching`

Smudge Test module for Signature class

Functions

Function `test_process_options`

```
def test_process_options()
```

Pytest for Process Options static method.

Function `test_signature_1`

```
def test_signature_1()
```

Signature.

Function `test_version`

```
def test_version()
```

Pytest for Version property.
