

Mercury

Generated by Doxygen 1.8.17

1 Data Structure Index	1
1.1 Data Structures	1
2 File Index	3
2.1 File List	3
3 Data Structure Documentation	5
3.1 libmerc_config Struct Reference	5
3.1.1 Detailed Description	5
3.1.2 Field Documentation	5
3.1.2.1 certs_json_output	6
3.1.2.2 dns_json_output	6
3.1.2.3 do_analysis	6
3.1.2.4 fp_proc_threshold	6
3.1.2.5 metadata_output	6
3.1.2.6 output_tcp_initial_data	6
3.1.2.7 output_udp_initial_data	6
3.1.2.8 packet_filter_cfg	6
3.1.2.9 proc_dst_threshold	7
3.1.2.10 report_os	7
3.1.2.11 resources	7
3.2 os_information Struct Reference	7
3.2.1 Detailed Description	7
3.2.2 Field Documentation	7
3.2.2.1 os_name	7
3.2.2.2 os_prevalence	7
4 File Documentation	9
4.1 src/libmerc/libmerc.h File Reference	9
4.1.1 Macro Definition Documentation	10
4.1.1.1 libmerc_config_init	10
4.1.2 Typedef Documentation	11
4.1.2.1 mercury_packet_processor	11
4.1.3 Enumeration Type Documentation	11
4.1.3.1 fingerprint_status	11
4.1.3.2 fingerprint_type	11
4.1.3.3 status	12
4.1.4 Function Documentation	12
4.1.4.1 analysis_context_get_fingerprint_status()	12
4.1.4.2 analysis_context_get_fingerprint_string()	12
4.1.4.3 analysis_context_get_fingerprint_type()	13
4.1.4.4 analysis_context_get_malware_info()	13
4.1.4.5 analysis_context_get_os_info()	13

4.1.4.6 analysis_context_get_process_info()	14
4.1.4.7 analysis_context_get_server_name()	14
4.1.4.8 mercury_finalize()	15
4.1.4.9 mercury_get_license_string()	15
4.1.4.10 mercury_init()	15
4.1.4.11 mercury_packet_processor_construct()	16
4.1.4.12 mercury_packet_processor_destruct()	16
4.1.4.13 mercury_packet_processor_get_analysis_context()	16
4.1.4.14 mercury_packet_processor_ip_get_analysis_context()	16
4.1.4.15 mercury_packet_processor_ip_write_json()	17
4.1.4.16 mercury_packet_processor_write_json()	17
4.1.4.17 mercury_print_version_string()	18
4.1.4.18 proto_ident_config()	18
4.1.4.19 static_data_config()	18

Index	19
--------------	-----------

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

libmerc_config	5
os_information	7

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

src/libmerc/ libmerc.h	9
--	---

Chapter 3

Data Structure Documentation

3.1 libmerc_config Struct Reference

```
#include <libmerc.h>
```

Data Fields

- bool [dns_json_output](#)
- bool [certs_json_output](#)
- bool [metadata_output](#)
- bool [do_analysis](#)
- bool [report_os](#)
- bool [output_tcp_initial_data](#)
- bool [output_udp_initial_data](#)
- char * [resources](#)
- char * [packet_filter_cfg](#)
- float [fp_proc_threshold](#)
- float [proc_dst_threshold](#)

3.1.1 Detailed Description

@brief struct [libmerc_config](#) represents the complete configuration of the libmerc library.

To initialize libmerc, create a [libmerc_config](#) structure and pass it to the [mercury_init\(\)](#) function. To create a [libmerc_config](#) structure, you can use the #define [libmerc_config_init\(\)](#), which represents a minimal, default configuration.

3.1.2 Field Documentation

3.1.2.1 certs_json_output

```
bool libmerc_config::certs_json_output
```

3.1.2.2 dns_json_output

```
bool libmerc_config::dns_json_output
```

3.1.2.3 do_analysis

```
bool libmerc_config::do_analysis
```

3.1.2.4 fp_proc_threshold

```
float libmerc_config::fp_proc_threshold
```

3.1.2.5 metadata_output

```
bool libmerc_config::metadata_output
```

3.1.2.6 output_tcp_initial_data

```
bool libmerc_config::output_tcp_initial_data
```

3.1.2.7 output_udp_initial_data

```
bool libmerc_config::output_udp_initial_data
```

3.1.2.8 packet_filter_cfg

```
char* libmerc_config::packet_filter_cfg
```

3.1.2.9 proc_dst_threshold

```
float libmerc_config::proc_dst_threshold
```

3.1.2.10 report_os

```
bool libmerc_config::report_os
```

3.1.2.11 resources

```
char* libmerc_config::resources
```

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

- [src/libmerc/libmerc.h](#)

3.2 os_information Struct Reference

```
#include <libmerc.h>
```

Data Fields

- `char *` [os_name](#)
- `uint64_t` [os_prevalence](#)

3.2.1 Detailed Description

[os_information](#) holds the name of an operating system and the prevalence with which it has been observed with a particular fingerprint.

3.2.2 Field Documentation

3.2.2.1 os_name

```
char* os_information::os_name
```

printable, null-terminated string holding OS name

3.2.2.2 os_prevalence

```
uint64_t os_information::os_prevalence
```

prevalence with which this OS is associated with fingerprint

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

- [src/libmerc/libmerc.h](#)

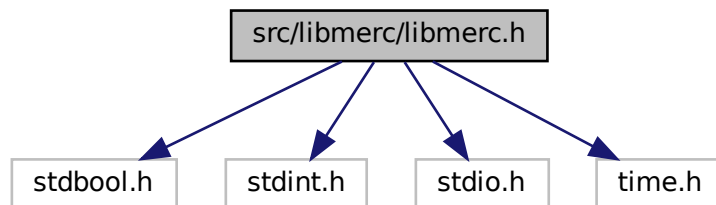
Chapter 4

File Documentation

4.1 src/libmerc/libmerc.h File Reference

```
#include <stdbool.h>
#include <stdint.h>
#include <stdio.h>
#include <time.h>
```

Include dependency graph for libmerc.h:



Data Structures

- struct [libmerc_config](#)
- struct [os_information](#)

Macros

- #define [libmerc_config_init\(\)](#) {false,false,false,false,false,false,false,NULL,NULL,0.0,0.0}

Typedefs

- typedef struct stateful_pkt_proc * [mercury_packet_processor](#)

Enumerations

- enum `fingerprint_status` { `fingerprint_status_no_info_available` = 0, `fingerprint_status_labeled` = 1, `fingerprint_status_randomized` = 2, `fingerprint_status_unlabeled` = 3 }
- enum `fingerprint_type` { `fingerprint_type_unknown` = 0, `fingerprint_type_tls` = 1 }
- enum `status` { `status_ok` = 0, `status_err` = 1, `status_err_no_more_data` = 2 }

Functions

- int `mercury_init` (const struct `libmerc_config` *vars, int verbosity)
initializes libmerc
- int `mercury_finalize` ()
finalizes libmerc
- `mercury_packet_processor` `mercury_packet_processor_construct` ()
- void `mercury_packet_processor_destruct` (`mercury_packet_processor` mpp)
- size_t `mercury_packet_processor_write_json` (`mercury_packet_processor` processor, void *buffer, size_t buffer_size, uint8_t *packet, size_t length, struct timespec *ts)
- size_t `mercury_packet_processor_ip_write_json` (`mercury_packet_processor` processor, void *buffer, size_t buffer_size, uint8_t *packet, size_t length, struct timespec *ts)
- const struct analysis_context * `mercury_packet_processor_ip_get_analysis_context` (`mercury_packet_processor` processor, uint8_t *packet, size_t length, struct timespec *ts)
- const struct analysis_context * `mercury_packet_processor_get_analysis_context` (`mercury_packet_processor` processor, uint8_t *packet, size_t length, struct timespec *ts)
- enum `fingerprint_status` `analysis_context_get_fingerprint_status` (const struct analysis_context *ac)
- enum `fingerprint_type` `analysis_context_get_fingerprint_type` (const struct analysis_context *ac)
- const char * `analysis_context_get_fingerprint_string` (const struct analysis_context *ac)
- const char * `analysis_context_get_server_name` (const struct analysis_context *ac)
- bool `analysis_context_get_process_info` (const struct analysis_context *ac, const char **probable_process, double *probability_score)
- bool `analysis_context_get_malware_info` (const struct analysis_context *ac, bool *probable_process_is_malware, double *probability_malware)
- bool `analysis_context_get_os_info` (const struct analysis_context *ac, const struct `os_information` **os_info, size_t *os_info_len)
- const char * `mercury_get_license_string` ()
returns the mercury license string
- void `mercury_print_version_string` (FILE *f)
prints the mercury semantic version
- enum `status` `proto_ident_config` (const char *config_string)
- enum `status` `static_data_config` (const char *config_string)

4.1.1 Macro Definition Documentation

4.1.1.1 libmerc_config_init

```
#define libmerc_config_init( ) {false,false,false,false,false,false,false, NULL, NULL, 0.0, 0.0}
```

`libmerc_config_init()` initializes a `libmerc_config` structure to a minimal, default configuration.

4.1.2 Typedef Documentation

4.1.2.1 mercury_packet_processor

```
typedef struct stateful_pkt_proc* mercury_packet_processor
```

mercury_packet_processor is an opaque pointer to a threadsafe packet processor.

4.1.3 Enumeration Type Documentation

4.1.3.1 fingerprint_status

```
enum fingerprint_status
```

enum fingerprint_status represents the status of a fingerprint relative to the library's knowledge about fingerprints, based on the data in its resources and the other fingerprints that it has observed.

Enumerator

fingerprint_status_no_info_available	fingerprint status is unknown
fingerprint_status_labeled	fingerprint is in FPDB
fingerprint_status_randomized	fingerprint is in randomized FP set
fingerprint_status_unlabeled	fingerprint is not in FPDB or randomized set

4.1.3.2 fingerprint_type

```
enum fingerprint_type
```

enum fingerprint_type identifies a type of fingerprint for the struct fingerprint.

Enumerator

fingerprint_type_unknown	The fingerprint type is not known.
fingerprint_type_tls	TLS fingerprint

4.1.3.3 status

enum `status`

Enumerator

<code>status_ok</code>	
<code>status_err</code>	
<code>status_err_no_more_data</code>	

4.1.4 Function Documentation

4.1.4.1 `analysis_context_get_fingerprint_status()`

```
enum fingerprint_status analysis_context_get_fingerprint_status (  
    const struct analysis_context * ac )
```

`analysis_context_get_fingerprint_status()` returns the `fingerprint_status` associated with an `analysis_context`.

Parameters

<code>ac</code>	(input) is an <code>analysis_context</code> pointer.
-----------------	--

Returns

a `fingerprint_status` enumeration.

4.1.4.2 `analysis_context_get_fingerprint_string()`

```
const char* analysis_context_get_fingerprint_string (  
    const struct analysis_context * ac )
```

`analysis_context_get_fingerprint_string()` returns the printable, null-terminated string for the fingerprint associated with an `analysis_context`, if there is one.

Parameters

<code>ac</code>	(input) is an <code>analysis_context</code> pointer.
-----------------	--

Returns

a null-terminated, printable character string, if a fingerprint was found by the library; otherwise, `NULL`.

4.1.4.3 analysis_context_get_fingerprint_type()

```
enum fingerprint_type analysis_context_get_fingerprint_type (
    const struct analysis_context * ac )
```

[analysis_context_get_fingerprint_type\(\)](#) returns the fingerprint_status associated with an analysis_context.

Parameters

<i>ac</i>	(input) is an analysis_context pointer.
-----------	---

Returns

a fingerprint_type enumeration.

4.1.4.4 analysis_context_get_malware_info()

```
bool analysis_context_get_malware_info (
    const struct analysis_context * ac,
    bool * probable_process_is_malware,
    double * probability_malware )
```

[analysis_context_get_malware_info\(\)](#) writes the probable_process_is_malware boolean and the probability_malware value into the locations provided, for a given analysis_context.

Parameters

<i>ac</i>	(input) is an analysis_context pointer.
<i>probable_process_is_malware</i>	(output) is the location to write the boolean.
<i>probability_malware</i>	(output) is the location to write the probability that the process is malware.

Returns

true if the probable_process_is_malware and probability_malware values are valid after the function returns, and false otherwise.

4.1.4.5 analysis_context_get_os_info()

```
bool analysis_context_get_os_info (
    const struct analysis_context * ac,
    const struct os_information ** os_info,
    size_t * os_info_len )
```

[analysis_context_get_os_info\(\)](#) sets a pointer to an array of os_information structures and the length of that array, for a given analysis_context.

Parameters

<i>ac</i>	(input) is an <code>analysis_context</code> pointer.
<i>os_info</i>	(output) is the location to which the os_information array pointer will be written.
<i>os_info_len</i>	(output) is the location to write the length of the <code>os_info</code> array.

Returns

true if the `os_info` and `os_info_len` locations point to valid data after the function returns, and false otherwise.

4.1.4.6 `analysis_context_get_process_info()`

```
bool analysis_context_get_process_info (
    const struct analysis_context * ac,
    const char ** probable_process,
    double * probability_score )
```

[analysis_context_get_process_info\(\)](#) writes the probable process and its corresponding probability score into the locations provided, given an `analysis_context`.

Parameters

<i>ac</i>	(input) is an <code>analysis_context</code> pointer.
<i>probable_process</i>	(output) is the location to write the <code>probable_process</code> string.
<i>probability_score</i>	(output) is the location to write the probability score.

Returns

true if the `probable_process` and `probability_score` are valid after the function returns, and false otherwise.

4.1.4.7 `analysis_context_get_server_name()`

```
const char* analysis_context_get_server_name (
    const struct analysis_context * ac )
```

[analysis_context_get_server_name\(\)](#) returns the printable, null-terminated string for the TLS client hello server name associated with an `analysis_context`, if there is one.

Parameters

<i>ac</i>	(input) is an <code>analysis_context</code> pointer.
-----------	--

Returns

a null-terminated, printable character string, if a TLS client hello server name was found by the library; otherwise, NULL.

4.1.4.8 mercury_finalize()

```
int mercury_finalize ( )
```

finalizes libmerc

Finalizes the libmerc library, and frees up resources allocated by [mercury_init\(\)](#). Returns zero on success.

Returns

0 on success, -1 on failure

4.1.4.9 mercury_get_license_string()

```
const char* mercury_get_license_string ( )
```

returns the mercury license string

Returns a printable string containing the license for mercury and libmerc.

4.1.4.10 mercury_init()

```
int mercury_init (
    const struct libmerc\_config * vars,
    int verbosity )
```

initializes libmerc

Initializes libmerc to use the configuration as specified with the input parameters. Returns zero on success.

Parameters

<i>vars</i>	libmerc_config
<i>verbosity</i>	higher values increase verbosity sent to stderr
<i>resource_dir</i>	directory of resource files to use in analysis

Returns

0 on success, -1 on failure

4.1.4.11 mercury_packet_processor_construct()

```
mercury_packet_processor mercury_packet_processor_construct ( )
```

[mercury_packet_processor_construct\(\)](#) allocates and initializes a new mercury_packet_processor.

Returns

a valid pointer on success, NULL otherwise.

4.1.4.12 mercury_packet_processor_destruct()

```
void mercury_packet_processor_destruct (
    mercury_packet_processor mpp )
```

[mercury_packet_processor_destruct\(\)](#) deallocates all resources associated with a mercury_packet_processor.

4.1.4.13 mercury_packet_processor_get_analysis_context()

```
const struct analysis_context* mercury_packet_processor_get_analysis_context (
    mercury_packet_processor processor,
    uint8_t * packet,
    size_t length,
    struct timespec * ts )
```

[mercury_packet_processor_get_analysis_context\(\)](#) processes an ethernet packet and timestamp and returns a pointer to an analysis context if a fingerprint was found in the packet, and returns nothing otherwise.

Parameters

<i>processor</i>	(input) is a packet processor context to be used
<i>buffer_size</i>	(input) - length of buffer in bytes
<i>packet</i>	(input) - location of packet, starting with the ethernet header
<i>ts</i>	(input) - pointer to timestamp associated with packet

Returns

a pointer to an analysis_context, if a fingerprint was found, otherwise NULL.

4.1.4.14 mercury_packet_processor_ip_get_analysis_context()

```
const struct analysis_context* mercury_packet_processor_ip_get_analysis_context (
    mercury_packet_processor processor,
    uint8_t * packet,
```

```
size_t length,  
struct timespec * ts )
```

[mercury_packet_processor_ip_get_analysis_context\(\)](#) processes an IP packet and timestamp and returns a pointer to an analysis context if a fingerprint was found in the packet, and returns nothing otherwise.

Parameters

<i>processor</i>	(input) is a packet processor context to be used
<i>buffer_size</i>	(input) - length of buffer in bytes
<i>packet</i>	(input) - location of packet, starting with IPv4 or IPv6 header
<i>ts</i>	(input) - pointer to timestamp associated with packet

Returns

a pointer to an analysis_context, if a fingerprint was found, otherwise NULL.

4.1.4.15 mercury_packet_processor_ip_write_json()

```
size_t mercury_packet_processor_ip_write_json (  
    mercury_packet_processor processor,  
    void * buffer,  
    size_t buffer_size,  
    uint8_t * packet,  
    size_t length,  
    struct timespec * ts )
```

[mercury_packet_processor_ip_write_json\(\)](#) processes a packet and timestamp and writes the resulting JSON into a buffer.

Parameters

<i>processor</i>	(input) is a packet processor context to be used
<i>buffer</i>	(output) - location to which JSON will be written
<i>buffer_size</i>	(input) - length of buffer in bytes
<i>packet</i>	(input) - location of packet, starting with IPv4 or IPv6 header
<i>ts</i>	(input) - pointer to timestamp associated with packet

Returns

the number of bytes of JSON output written.

4.1.4.16 mercury_packet_processor_write_json()

```
size_t mercury_packet_processor_write_json (  
    mercury_packet_processor processor,
```

```

void * buffer,
size_t buffer_size,
uint8_t * packet,
size_t length,
struct timespec * ts )

```

[mercury_packet_processor_write_json\(\)](#) processes a packet and timestamp and writes the resulting JSON into a buffer.

Parameters

<i>processor</i>	(input) is a packet processor context to be used
<i>buffer</i>	(output) - location to which JSON will be written
<i>buffer_size</i>	(input) - length of buffer in bytes
<i>packet</i>	(input) - location of packet, starting with ethernet header
<i>ts</i>	(input) - pointer to timestamp associated with packet

Returns

the number of bytes of JSON output written.

4.1.4.17 mercury_print_version_string()

```

void mercury_print_version_string (
    FILE * f )

```

prints the mercury semantic version

Prints the semantic version of mercury/libmerc to the FILE provided as input.

Parameters

<i>in</i>	<i>file</i>	to print semantic version on.
-----------	-------------	-------------------------------

4.1.4.18 proto_ident_config()

```

enum status proto_ident_config (
    const char * config_string )

```

4.1.4.19 static_data_config()

```

enum status static_data_config (
    const char * config_string )

```

Index

analysis_context_get_fingerprint_status
libmerc.h, [12](#)

analysis_context_get_fingerprint_string
libmerc.h, [12](#)

analysis_context_get_fingerprint_type
libmerc.h, [12](#)

analysis_context_get_malware_info
libmerc.h, [13](#)

analysis_context_get_os_info
libmerc.h, [13](#)

analysis_context_get_process_info
libmerc.h, [14](#)

analysis_context_get_server_name
libmerc.h, [14](#)

certs_json_output
libmerc_config, [5](#)

dns_json_output
libmerc_config, [6](#)

do_analysis
libmerc_config, [6](#)

fingerprint_status
libmerc.h, [11](#)

fingerprint_status_labeled
libmerc.h, [11](#)

fingerprint_status_no_info_available
libmerc.h, [11](#)

fingerprint_status_randomized
libmerc.h, [11](#)

fingerprint_status_unlabeled
libmerc.h, [11](#)

fingerprint_type
libmerc.h, [11](#)

fingerprint_type_tls
libmerc.h, [11](#)

fingerprint_type_unknown
libmerc.h, [11](#)

fp_proc_threshold
libmerc_config, [6](#)

libmerc.h

analysis_context_get_fingerprint_status, [12](#)

analysis_context_get_fingerprint_string, [12](#)

analysis_context_get_fingerprint_type, [12](#)

analysis_context_get_malware_info, [13](#)

analysis_context_get_os_info, [13](#)

analysis_context_get_process_info, [14](#)

analysis_context_get_server_name, [14](#)

fingerprint_status, [11](#)

fingerprint_status_labeled, [11](#)

fingerprint_status_no_info_available, [11](#)

fingerprint_status_randomized, [11](#)

fingerprint_status_unlabeled, [11](#)

fingerprint_type, [11](#)

fingerprint_type_tls, [11](#)

fingerprint_type_unknown, [11](#)

libmerc_config_init, [10](#)

mercury_finalize, [15](#)

mercury_get_license_string, [15](#)

mercury_init, [15](#)

mercury_packet_processor, [11](#)

mercury_packet_processor_construct, [15](#)

mercury_packet_processor_destruct, [16](#)

mercury_packet_processor_get_analysis_context,
[16](#)

mercury_packet_processor_ip_get_analysis_context,
[16](#)

mercury_packet_processor_ip_write_json, [17](#)

mercury_packet_processor_write_json, [17](#)

mercury_print_version_string, [18](#)

proto_ident_config, [18](#)

static_data_config, [18](#)

status, [11](#)

status_err, [12](#)

status_err_no_more_data, [12](#)

status_ok, [12](#)

libmerc_config, [5](#)

certs_json_output, [5](#)

dns_json_output, [6](#)

do_analysis, [6](#)

fp_proc_threshold, [6](#)

metadata_output, [6](#)

output_tcp_initial_data, [6](#)

output_udp_initial_data, [6](#)

packet_filter_cfg, [6](#)

proc_dst_threshold, [6](#)

report_os, [7](#)

resources, [7](#)

libmerc_config_init

libmerc.h, [10](#)

mercury_finalize

libmerc.h, [15](#)

mercury_get_license_string

libmerc.h, [15](#)

mercury_init

libmerc.h, [15](#)

mercury_packet_processor

- libmerc.h, [11](#)
- mercury_packet_processor_construct
 - libmerc.h, [15](#)
- mercury_packet_processor_destruct
 - libmerc.h, [16](#)
- mercury_packet_processor_get_analysis_context
 - libmerc.h, [16](#)
- mercury_packet_processor_ip_get_analysis_context
 - libmerc.h, [16](#)
- mercury_packet_processor_ip_write_json
 - libmerc.h, [17](#)
- mercury_packet_processor_write_json
 - libmerc.h, [17](#)
- mercury_print_version_string
 - libmerc.h, [18](#)
- metadata_output
 - libmerc_config, [6](#)
- os_information, [7](#)
 - os_name, [7](#)
 - os_prevalence, [7](#)
- os_name
 - os_information, [7](#)
- os_prevalence
 - os_information, [7](#)
- output_tcp_initial_data
 - libmerc_config, [6](#)
- output_udp_initial_data
 - libmerc_config, [6](#)
- packet_filter_cfg
 - libmerc_config, [6](#)
- proc_dst_threshold
 - libmerc_config, [6](#)
- proto_ident_config
 - libmerc.h, [18](#)
- report_os
 - libmerc_config, [7](#)
- resources
 - libmerc_config, [7](#)
- src/libmerc/libmerc.h, [9](#)
- static_data_config
 - libmerc.h, [18](#)
- status
 - libmerc.h, [11](#)
- status_err
 - libmerc.h, [12](#)
- status_err_no_more_data
 - libmerc.h, [12](#)
- status_ok
 - libmerc.h, [12](#)