

Reference Manual

Generated by Doxygen 1.8.6

Wed Mar 4 2015 12:10:55

Contents

1	Module Documentation	1
1.1	Abstract Data Types	1
1.1.1	Detailed Description	2
1.1.2	Function Documentation	2
1.2	Bytecode Configuration	9
1.2.1	Detailed Description	9
1.2.2	Macro Definition Documentation	9
1.2.3	Enumeration Type Documentation	11
1.3	Debugging	13
1.3.1	Detailed Description	13
1.3.2	Function Documentation	13
1.4	Disassembly	16
1.4.1	Detailed Description	16
1.4.2	Function Documentation	16
1.5	Engine Queries	17
1.5.1	Detailed Description	17
1.5.2	Function Documentation	17
1.6	Environment	19
1.6.1	Detailed Description	19
1.6.2	Function Documentation	19
1.7	File Operations	23
1.7.1	Detailed Description	23
1.7.2	Enumeration Type Documentation	23
1.7.3	Function Documentation	23
1.8	Global Variables	27
1.8.1	Detailed Description	27
1.8.2	Variable Documentation	27
1.9	JavaScript Normalization	28
1.9.1	Detailed Description	28
1.9.2	Function Documentation	28
1.10	JSON Querying	29
1.10.1	Detailed Description	29
1.10.2	Enumeration Type Documentation	29
1.10.3	Function Documentation	29
1.11	Icon Matcher	32
1.11.1	Detailed Description	32
1.11.2	Function Documentation	32
1.12	Math Operation	33

1.12.1 Detailed Description	33
1.12.2 Function Documentation	33
1.13 PDF Handling	35
1.13.1 Detailed Description	35
1.13.2 Enumeration Type Documentation	35
1.13.3 Function Documentation	35
1.14 PE Operations	40
1.14.1 Detailed Description	41
1.14.2 Function Documentation	41
1.15 Scan Control	49
1.15.1 Detailed Description	49
1.15.2 Function Documentation	49
1.16 String Operations	51
1.16.1 Detailed Description	51
1.16.2 Function Documentation	51
2 Data Structure Documentation	54
2.1 cli_exe_info Struct Reference	54
2.1.1 Detailed Description	54
2.1.2 Field Documentation	54
2.2 cli_exe_section Struct Reference	54
2.2.1 Detailed Description	55
2.2.2 Field Documentation	55
2.3 cli_pe_hook_data Struct Reference	55
2.3.1 Detailed Description	56
2.3.2 Field Documentation	56
2.4 DIS_arg Struct Reference	56
2.4.1 Detailed Description	56
2.4.2 Field Documentation	57
2.5 DIS_fixed Struct Reference	57
2.5.1 Detailed Description	57
2.5.2 Field Documentation	57
2.6 DIS_mem_arg Struct Reference	58
2.6.1 Detailed Description	58
2.6.2 Field Documentation	58
2.7 DISASM_RESULT Struct Reference	58
2.7.1 Detailed Description	58
2.8 pe_image_data_dir Struct Reference	58
2.8.1 Detailed Description	58
2.9 pe_image_file_hdr Struct Reference	58

2.9.1	Detailed Description	59
2.9.2	Field Documentation	59
2.10	pe_image_optional_hdr32 Struct Reference	59
2.10.1	Detailed Description	60
2.10.2	Field Documentation	60
2.11	pe_image_optional_hdr64 Struct Reference	61
2.11.1	Detailed Description	61
2.11.2	Field Documentation	61
2.12	pe_image_section_hdr Struct Reference	62
2.12.1	Detailed Description	62
2.12.2	Field Documentation	62
3	File Documentation	63
3.1	bytecode_api.h File Reference	63
3.1.1	Enumeration Type Documentation	65
3.1.2	Function Documentation	65
3.2	bytecode_disasm.h File Reference	66
3.2.1	Enumeration Type Documentation	68
3.3	bytecode_execs.h File Reference	75
3.4	bytecode_local.h File Reference	75
3.4.1	Macro Definition Documentation	77
3.4.2	Function Documentation	77
3.5	bytecode_pe.h File Reference	77
	Index	78

1 Module Documentation

1.1 Abstract Data Types

Functions

- void * [malloc](#) (uint32_t size)
- int32_t [hashset_new](#) (void)
- int32_t [hashset_add](#) (int32_t hs, uint32_t key)
- int32_t [hashset_remove](#) (int32_t hs, uint32_t key)
- int32_t [hashset_contains](#) (int32_t hs, uint32_t key)
- int32_t [hashset_done](#) (int32_t id)
- int32_t [hashset_empty](#) (int32_t id)
- int32_t [buffer_pipe_new](#) (uint32_t size)
- int32_t [buffer_pipe_new_fromfile](#) (uint32_t pos)
- uint32_t [buffer_pipe_read_avail](#) (int32_t id)
- const uint8_t * [buffer_pipe_read_get](#) (int32_t id, uint32_t amount)
- int32_t [buffer_pipe_read_stopped](#) (int32_t id, uint32_t amount)
- uint32_t [buffer_pipe_write_avail](#) (int32_t id)

- `uint8_t * buffer_pipe_write_get` (`int32_t id`, `uint32_t size`)
- `int32_t buffer_pipe_write_stopped` (`int32_t id`, `uint32_t amount`)
- `int32_t buffer_pipe_done` (`int32_t id`)
- `int32_t inflate_init` (`int32_t from_buffer`, `int32_t to_buffer`, `int32_t windowBits`)
- `int32_t inflate_process` (`int32_t id`)
- `int32_t inflate_done` (`int32_t id`)
- `int32_t map_new` (`int32_t keysize`, `int32_t valuesize`)
- `int32_t map_addkey` (`const uint8_t *key`, `int32_t ksize`, `int32_t id`)
- `int32_t map_setvalue` (`const uint8_t *value`, `int32_t vsize`, `int32_t id`)
- `int32_t map_remove` (`const uint8_t *key`, `int32_t ksize`, `int32_t id`)
- `int32_t map_find` (`const uint8_t *key`, `int32_t ksize`, `int32_t id`)
- `int32_t map_getvaluesize` (`int32_t id`)
- `uint8_t * map_getvalue` (`int32_t id`, `int32_t size`)
- `int32_t map_done` (`int32_t id`)

1.1.1 Detailed Description

1.1.2 Function Documentation

1.1.2.1 `int32_t buffer_pipe_done (int32_t id)`

Deallocate memory used by buffer. After this all attempts to use this buffer will result in error. All `buffer_pipes` are automatically deallocated when bytecode finishes execution.

Parameters

<code>in</code>	<code>id</code>	ID of <code>buffer_pipe</code>
-----------------	-----------------	--------------------------------

Returns

0 on success

1.1.2.2 `int32_t buffer_pipe_new (uint32_t size)`

Creates a new pipe with the specified buffer size

Parameters

<code>in</code>	<code>size</code>	size of buffer
-----------------	-------------------	----------------

Returns

ID of newly created `buffer_pipe`

1.1.2.3 `int32_t buffer_pipe_new_fromfile (uint32_t pos)`

Creates a new pipe with the specified buffer size w/ tied input to the current file, at the specified position.

Parameters

<code>in</code>	<code>pos</code>	starting position of pipe input in current file
-----------------	------------------	-------------------------------------------------

Returns

ID of newly created `buffer_pipe`

1.1.2.4 `uint32_t buffer_pipe_read_avail (int32_t id)`

Returns the amount of bytes available to read.

Parameters

<i>in</i>	<i>id</i>	ID of <code>buffer_pipe</code>
-----------	-----------	--------------------------------

Returns

amount of bytes available to read

1.1.2.5 `const uint8_t* buffer_pipe_read_get (int32_t id, uint32_t amount)`

Returns a pointer to the buffer for reading. The 'amount' parameter should be obtained by a call to [buffer_pipe_read_avail\(\)](#).

Parameters

<i>in</i>	<i>id</i>	ID of <code>buffer_pipe</code>
<i>in</i>	<i>amount</i>	to read

Returns

pointer to buffer, or NULL if buffer has less than specified amount

1.1.2.6 `int32_t buffer_pipe_read_stopped (int32_t id, uint32_t amount)`

Updates read cursor in `buffer_pipe`.

Parameters

<i>in</i>	<i>id</i>	ID of <code>buffer_pipe</code>
<i>in</i>	<i>amount</i>	amount of bytes to move read cursor

Returns

0 on success

1.1.2.7 `uint32_t buffer_pipe_write_avail (int32_t id)`

Returns the amount of bytes available for writing.

Parameters

<i>in</i>	<i>id</i>	ID of <code>buffer_pipe</code>
-----------	-----------	--------------------------------

Returns

amount of bytes available for writing

1.1.2.8 `uint8_t* buffer_pipe_write_get (int32_t id, uint32_t size)`

Returns pointer to writable buffer. The 'size' parameter should be obtained by a call to [buffer_pipe_write_avail\(\)](#).

Parameters

<i>in</i>	<i>id</i>	ID of <code>buffer_pipe</code>
<i>in</i>	<i>size</i>	amount of bytes to write

Returns

pointer to write buffer, or NULL if requested amount is more than what is available in the buffer

1.1.2.9 `int32_t buffer_pipe_write_stopped (int32_t id, uint32_t amount)`

Updates the write cursor in `buffer_pipe`.

Parameters

in	<i>id</i>	ID of <code>buffer_pipe</code>
in	<i>amount</i>	amount of bytes to move write cursor

Returns

0 on success

1.1.2.10 `int32_t` `hashset_add` (`int32_t` *hs*, `uint32_t` *key*)

Add a new 32-bit key to the hashset.

Parameters

in	<i>hs</i>	ID of hashset (from <code>hashset_new</code>)
in	<i>key</i>	the key to add

Returns

0 on success

1.1.2.11 `int32_t` `hashset_contains` (`int32_t` *hs*, `uint32_t` *key*)

Returns whether the hashset contains the specified key.

Parameters

in	<i>hs</i>	ID of hashset (from <code>hashset_new</code>)
in	<i>key</i>	the key to lookup

Returns

1 if found
0 if not found
<0 on invalid hashset ID

1.1.2.12 `int32_t` `hashset_done` (`int32_t` *id*)

Deallocates the memory used by the specified hashset. Trying to use the hashset after this will result in an error. The hashset may not be used after this. All hashsets are automatically deallocated when bytecode finishes execution.

Parameters

in	<i>id</i>	ID of hashset (from <code>hashset_new</code>)
----	-----------	------------------------------------------------

Returns

0 on success

1.1.2.13 `int32_t` `hashset_empty` (`int32_t` *id*)

Returns whether the hashset is empty.

Parameters

<i>in</i>	<i>id</i>	of hashset (from <code>hashset_new</code>)
-----------	-----------	---------------------------------------------

Returns

0 on success

1.1.2.14 `int32_t hashset_new (void)`

Creates a new hashset and returns its id.

Returns

ID for new hashset

1.1.2.15 `int32_t hashset_remove (int32_t hs, uint32_t key)`

Remove a 32-bit key from the hashset.

Parameters

<i>in</i>	<i>hs</i>	ID of hashset (from <code>hashset_new</code>)
<i>in</i>	<i>key</i>	the key to add

Returns

0 on success

1.1.2.16 `int32_t inflate_done (int32_t id)`

Deallocates inflate data structure. Using the inflate data structure after this will result in an error. All inflate data structures are automatically deallocated when bytecode finishes execution.

Parameters

<i>in</i>	<i>id</i>	ID of inflate data structure
-----------	-----------	------------------------------

Returns

0 on success.

1.1.2.17 `int32_t inflate_init (int32_t from_buffer, int32_t to_buffer, int32_t windowBits)`

Initializes inflate data structures for decompressing data 'from_buffer' and writing uncompressed data 'to_buffer'.

Parameters

<i>in</i>	<i>from_buffer</i>	ID of buffer_pipe to read compressed data from
<i>in</i>	<i>to_buffer</i>	ID of buffer_pipe to write decompressed data to
<i>in</i>	<i>windowBits</i>	(see zlib documentation)

Returns

ID of newly created inflate data structure, <0 on failure

1.1.2.18 `int32_t inflate_process (int32_t id)`

Inflate all available data in the input buffer, and write to output buffer. Stops when the input buffer becomes empty, or write buffer becomes full. Also attempts to recover from corrupted inflate stream (via `inflateSync`). This function can be called repeatedly on success after filling the input buffer, and flushing the output buffer. The inflate stream is done processing when 0 bytes are available from output buffer, and input buffer is not empty.

Parameters

<i>in</i>	<i>id</i>	ID of inflate data structure
-----------	-----------	------------------------------

Returns

0 on success, zlib error code otherwise

1.1.2.19 void* malloc (uint32_t size)

Allocates memory. Currently this memory is freed automatically on exit from the bytecode, and there is no way to free it sooner.

Parameters

<i>in</i>	<i>size</i>	amount of memory to allocate in bytes
-----------	-------------	---------------------------------------

Returns

pointer to allocated memory

1.1.2.20 int32_t map_addkey (const uint8_t* key, int32_t ksize, int32_t id)

Inserts the specified key/value pair into the map.

Parameters

<i>in</i>	<i>id</i>	id of table
<i>in</i>	<i>key</i>	key
<i>in</i>	<i>ksize</i>	size of key

Returns

0 - if key existed before
 1 - if key didn't exist before
 <0 - if ksize doesn't match keysize specified at table creation

1.1.2.21 int32_t map_done (int32_t id)

Deallocates the memory used by the specified map. Trying to use the map after this will result in an error. All maps are automatically deallocated when the bytecode finishes execution.

Parameters

<i>in</i>	<i>id</i>	id of map
-----------	-----------	-----------

Returns

0 - success
 -1 - invalid map

1.1.2.22 int32_t map_find (const uint8_t* key, int32_t ksize, int32_t id)

Looks up key in map. The map remember the last looked up key (so you can retrieve the value).

Parameters

in	<i>id</i>	id of map
in	<i>key</i>	key
in	<i>ksize</i>	size of key

Returns

0 - if not found
1 - if found
<0 - if ksize doesn't match the size specified at table creation

1.1.2.23 uint8_t* map_getvalue (int32_t id, int32_t size)

Returns the value obtained during last map_find.

Parameters

in	<i>id</i>	id of map.
in	<i>size</i>	size of value (obtained from map_getvaluesize)

Returns

value

1.1.2.24 int32_t map_getvaluesize (int32_t id)

Returns the size of value obtained during last map_find.

Parameters

in	<i>id</i>	id of map.
----	-----------	------------

Returns

size of value

1.1.2.25 int32_t map_new (int32_t keysize, int32_t valuesize)

Creates a new map and returns its id.

Parameters

in	<i>keysize</i>	size of key
in	<i>valuesize</i>	size of value, if 0 then value is allocated separately

Returns

ID of new map

1.1.2.26 int32_t map_remove (const uint8_t * key, int32_t ksize, int32_t id)

Remove an element from the map.

Parameters

in	<i>id</i>	id of map
in	<i>key</i>	key
in	<i>ksize</i>	size of key

Returns

0 on success, key was present
1 if key was not present
<0 if ksize doesn't match keysize specified at table creation

1.1.2.27 `int32_t map_setvalue (const uint8_t * value, int32_t vsize, int32_t id)`

Sets the value for the last inserted key with `map_addkey`.

Parameters

in	<i>id</i>	id of table
in	<i>value</i>	value
in	<i>vsize</i>	size of value

Returns

0 - if update was successful
<0 - if there is no last key

1.2 Bytecode Configuration

Macros

- `#define VIRUSNAME_PREFIX(name) const char __clambc_virusname_prefix[] = name;`
- `#define VIRUSNAMES(...) const char *const __clambc_virusnames[] = {__VA_ARGS__};`
- `#define PE_UNPACKER_DECLARE const uint16_t __clambc_kind = BC_PE_UNPACKER;`
- `#define PDF_HOOK_DECLARE const uint16_t __clambc_kind = BC_PDF;`
- `#define PE_HOOK_DECLARE const uint16_t __clambc_kind = BC_PE_ALL;`
- `#define PRECLASS_HOOK_DECLARE const uint16_t __clambc_kind = BC_PRECLASS;`
- `#define SIGNATURES_DECL_BEGIN struct __Signatures {`
- `#define DECLARE_SIGNATURE(name)`
- `#define SIGNATURES_DECL_END };`
- `#define TARGET(tgt) const unsigned short __Target = (tgt);`
- `#define COPYRIGHT(c) const char *const __Copyright = (c);`
- `#define ICONGROUP1(group) const char *const __IconGroup1 = (group);`
- `#define ICONGROUP2(group) const char *const __IconGroup2 = (group);`
- `#define FUNCTIONALITY_LEVEL_MIN(m) const unsigned short __FuncMin = (m);`
- `#define FUNCTIONALITY_LEVEL_MAX(m) const unsigned short __FuncMax = (m);`
- `#define SIGNATURES_DEF_BEGIN`
- `#define SIGNATURES_DEF_END };`

Enumerations

- `enum BytecodeKind {`
`BC_GENERIC = 0, BC_STARTUP = 1, BC_LOGICAL = 256, BC_PE_UNPACKER,`
`BC_PDF, BC_PE_ALL, BC_PRECLASS };`
- `enum FunctionalityLevels {`
`FUNC_LEVEL_096 = 51, FUNC_LEVEL_096_1 = 53, FUNC_LEVEL_096_2 = 54, FUNC_LEVEL_096_3`
`= 55,`
`FUNC_LEVEL_096_4 = 56, FUNC_LEVEL_096_5 = 58, FUNC_LEVEL_097 = 60, FUNC_LEVEL_097_1 =`
`61,`
`FUNC_LEVEL_097_2 = 62, FUNC_LEVEL_097_3 = 63, FUNC_LEVEL_097_4 = 64, FUNC_LEVEL_097_5`
`= 65,`
`FUNC_LEVEL_097_6 = 67, FUNC_LEVEL_097_7 = 68, FUNC_LEVEL_097_8 = 69, FUNC_LEVEL_098_1`
`= 76,`
`FUNC_LEVEL_098_2 = 77, FUNC_LEVEL_098_3 = 77, FUNC_LEVEL_098_4 = 77, FUNC_LEVEL_098_5`
`= 79,`
`FUNC_LEVEL_098_6 = 79, FUNC_LEVEL_098_7 = 80 };`

1.2.1 Detailed Description

1.2.2 Macro Definition Documentation

1.2.2.1 `#define COPYRIGHT(c) const char *const __Copyright = (c);`

Defines an alternative copyright for this bytecode.

This will also prevent the sourcecode from being embedded into the bytecode.

1.2.2.2 `#define DECLARE_SIGNATURE(name)`

Value:

```
const char *name##_sig;\n__Signature name;
```

Declares a name for a subsignature.

1.2.2.3 `#define FUNCTIONALITY_LEVEL_MAX(m) const unsigned short __FuncMax = (m);`

Define the maximum engine functionality level required for this bytecode/logical signature.

Engines newer than this will skip loading the bytecode. You can use the [FunctionalityLevels](#) enumeration here.

1.2.2.4 `#define FUNCTIONALITY_LEVEL_MIN(m) const unsigned short __FuncMin = (m);`

Define the minimum engine functionality level required for this bytecode/logical signature.

Engines older than this will skip loading the bytecode. You can use the [FunctionalityLevels](#) enumeration here.

1.2.2.5 `#define ICONGROUP1(group) const char *const __IconGroup1 = (group);`

Define IconGroup1 for logical signature.

See logical signature documentation for what it is.

1.2.2.6 `#define ICONGROUP2(group) const char *const __IconGroup2 = (group);`

Define IconGroup2 for logical signature.

See logical signature documentation for what it is.

1.2.2.7 `#define PDF_HOOK_DECLARE const uint16_t __clambc_kind = BC_PDF;`

Make the current bytecode a PDF hook.

Having a logical signature doesn't make sense here, since the logical signature is evaluated AFTER these hooks run.

This hook is called several times, use [pdf_get_phase\(\)](#) to find out in which phase you got called.

1.2.2.8 `#define PE_HOOK_DECLARE const uint16_t __clambc_kind = BC_PE_ALL;`

Make the current bytecode a PE hook.

Bytecode will be called once the logical signature trigger matches (or always if there is none), and if you have access to all the PE information. By default you only have access to `execs.h` information, and not to PE field information (even for PE files).

1.2.2.9 `#define PE_UNPACKER_DECLARE const uint16_t __clambc_kind = BC_PE_UNPACKER;`

Like `PE_HOOK_DECLARE`, but it is not run for packed files that `pe.c` can unpack (only on the unpacked file).

1.2.2.10 `#define PRECLASS_HOOK_DECLARE const uint16_t __clambc_kind = BC_PRECLASS;`

Make the current bytecode a PRECLASS hook.

Bytecode will be called once the logical signature trigger matches (or always if there is none), and if you have access to all PRECLASS information.

1.2.2.11 `#define SIGNATURES_DECL_BEGIN struct __Signatures {`

Marks the beginning of the subsignature name declaration section.

1.2.2.12 `#define SIGNATURES_DECL_END};`

Marks the end of the subsignature name declaration section.

1.2.2.13 `#define SIGNATURES_DEF_BEGIN`

Value:

```
static const unsigned __signature_bias = __COUNTER__ + 1;\nconst struct __Signatures Signatures = {\n
```

Marks the beginning of subsignature pattern definitions.

See Also

[SIGNATURES_DECL_BEGIN](#)

1.2.2.14 `#define SIGNATURES_DEF_END` ;

Marks the end of the subsignature pattern definitions.

Alternative: `SIGNATURES_END`

1.2.2.15 `#define TARGET(tgt) const unsigned short __Target = (tgt);`

Defines the ClamAV file target.

Parameters

<code>in</code>	<code>tgt</code>	ClamAV signature type (0 - raw, 1 - PE, etc.)
-----------------	------------------	-----------------------------------------------

1.2.2.16 `#define VIRUSNAME_PREFIX(name) const char __clambc_virusname_prefix[] = name;`

Declares the virusname prefix.

Parameters

<code>in</code>	<code>name</code>	the prefix common to all viruses reported by this bytecode
-----------------	-------------------	------------------------------------------------------------

1.2.2.17 `#define VIRUSNAMES(...) const char *const __clambc_virusnames[] = {__VA_ARGS__};`

Declares all the virusnames that this bytecode can report.

Parameters

<code>in</code>	<code>...</code>	a comma-separated list of strings interpreted as virusnames
-----------------	------------------	-------------------------------------------------------------

1.2.3 Enumeration Type Documentation

1.2.3.1 enum BytecodeKind

Specifies the bytecode type and how ClamAV executes it

Enumerator

BC_GENERIC generic bytecode, not tied a specific hook

BC_STARTUP triggered at startup, only one is allowed per ClamAV startup

BC_LOGICAL executed on a logical trigger

BC_PE_UNPACKER specifies a PE unpacker, executed on PE files on a logical trigger

BC_PDF specifies a PDF hook, executes at a predetermined point of PDF parsing for PDF files

BC_PE_ALL specifies a PE hook, executes at a predetermined point in PE parsing for PE files, both packed and unpacked files

BC_PRECLASS specifies a PRECLASS hook, executes at the end of file property collection and operates on the original file targeted for property collection

1.2.3.2 enum FunctionalityLevels

LibClamAV functionality level constants

Enumerator

- FUNC_LEVEL_096*** LibClamAV release 0.96.0: bytecode engine released
- FUNC_LEVEL_096_1*** LibClamAV release 0.96.1: logical signature use of VI/macros requires this minimum functionality level
- FUNC_LEVEL_096_2*** LibClamAV release 0.96.2: PDF Hooks require this minimum level
- FUNC_LEVEL_096_3*** LibClamAV release 0.96.3: BC_PE_ALL bytecodes require this minimum level
- FUNC_LEVEL_096_4*** LibClamAV release 0.96.4: minimum recommended engine version, older versions have quadratic load time
- FUNC_LEVEL_096_5*** LibClamAV release 0.96.5
- FUNC_LEVEL_097*** LibClamAV release 0.97.0: older bytecodes may incorrectly use 57
- FUNC_LEVEL_097_1*** LibClamAV release 0.97.1
- FUNC_LEVEL_097_2*** LibClamAV release 0.97.2
- FUNC_LEVEL_097_3*** LibClamAV release 0.97.3
- FUNC_LEVEL_097_4*** LibClamAV release 0.97.4
- FUNC_LEVEL_097_5*** LibClamAV release 0.97.5
- FUNC_LEVEL_097_6*** LibClamAV release 0.97.6
- FUNC_LEVEL_097_7*** LibClamAV release 0.97.7
- FUNC_LEVEL_097_8*** LibClamAV release 0.97.8
- FUNC_LEVEL_098_1*** LibClamAV release 0.98.1
- FUNC_LEVEL_098_2*** LibClamAV release 0.98.2
- FUNC_LEVEL_098_3*** LibClamAV release 0.98.3
- FUNC_LEVEL_098_4*** LibClamAV release 0.98.4
- FUNC_LEVEL_098_5*** LibClamAV release 0.98.5: JSON reading API requires this minimum level
- FUNC_LEVEL_098_6*** LibClamAV release 0.98.6
- FUNC_LEVEL_098_7*** LibClamAV release 0.98.7: BC_PRECLASS bytecodes require minimum level

1.3 Debugging

Functions

- uint32_t [debug_print_str](#) (const uint8_t *str, uint32_t len)
- uint32_t [debug_print_uint](#) (uint32_t a)
- uint32_t [debug_print_str_start](#) (const uint8_t *str, uint32_t len)
- uint32_t [debug_print_str_nonl](#) (const uint8_t *str, uint32_t len)
- void [debug](#) (...) __attribute__((overloadable))
- static force_inline void
overloadable_func [debug](#) (const char *str)
- static force_inline void
overloadable_func [debug](#) (const uint8_t *str)
- static force_inline void
overloadable_func [debug](#) (uint32_t a)

1.3.1 Detailed Description

1.3.2 Function Documentation

1.3.2.1 `debug (const char * str) [static]`

Prints `str` to clamscan's `-debug` output. This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

in	<i>str</i>	null terminated string
----	------------	------------------------

1.3.2.2 `debug (const uint8_t * str) [static]`

Prints `str` to clamscan's `-debug` output. This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

in	<i>str</i>	null terminated string
----	------------	------------------------

1.3.2.3 `debug (uint32_t a) [static]`

Prints `a` integer to clamscan's `-debug` output. This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

in	<i>a</i>	integer
----	----------	---------

1.3.2.4 `void debug (...)`

`debug` is an overloaded function (yes clang supports that in C!), but it only works on strings, and integers. Give an error on any other type.

See Also

```
debug(const char * str),
debug(const uint8_t* str),
debug(uint32_t a)
```


1.3.2.5 `uint32_t debug_print_str (const uint8_t * str, uint32_t len)`

Prints a debug message string.

Parameters

in	<i>str</i>	Message to print
in	<i>len</i>	length of message to print

Returns

0

1.3.2.6 `uint32_t debug_print_str_nonl (const uint8_t * str, uint32_t len)`

Prints a debug message with a trailing newline, and not preceded by 'LibClamAV debug'.

Parameters

in	<i>str</i>	the string
in	<i>len</i>	length of <i>str</i>

Returns

0

1.3.2.7 `uint32_t debug_print_str_start (const uint8_t * str, uint32_t len)`

Prints a debug message with a trailing newline, but preceded by 'LibClamAV debug'.

Parameters

in	<i>str</i>	the string
in	<i>len</i>	length of <i>str</i>

Returns

0

1.3.2.8 `uint32_t debug_print_uint (uint32_t a)`

Prints a number as a debug message. This is similar to `debug_print_str_nonl`.

Parameters

in	<i>a</i>	number to print
----	----------	-----------------

Returns

0

1.4 Disassembly

Data Structures

- struct [DIS_mem_arg](#)
- struct [DIS_arg](#)
- struct [DIS_fixed](#)

Functions

- uint32_t [disasm_x86](#) (struct [DISASM_RESULT](#) *result, uint32_t len)
- static force_inline uint32_t [DisassembleAt](#) (struct [DIS_fixed](#) *result, uint32_t offset, uint32_t len)

1.4.1 Detailed Description

1.4.2 Function Documentation

1.4.2.1 uint32_t disasm_x86 (struct [DISASM_RESULT](#) * result, uint32_t len)

Disassembles starting from current file position, the specified amount of bytes.

Parameters

out	<i>result</i>	pointer to struct holding result
in	<i>len</i>	how many bytes to disassemble

Returns

0 for success

You can use `lseek` to disassemble starting from a different location. This is a low-level API, the result is in ClamAV type-8 signature format (64 bytes/instruction).

See Also

[DisassembleAt](#)

1.4.2.2 static force_inline uint32_t DisassembleAt (struct [DIS_fixed](#) * result, uint32_t offset, uint32_t len) [static]

Disassembles one X86 instruction starting at the specified offset.

Parameters

out	<i>result</i>	disassembly result
in	<i>offset</i>	start disassembling from this offset, in the current file
in	<i>len</i>	max amount of bytes to disassemble

Returns

offset where disassembly ended

1.5 Engine Queries

Functions

- uint32_t [engine_functionality_level](#) (void)
- uint32_t [engine_dconf_level](#) (void)
- uint32_t [engine_scan_options](#) (void)
- uint32_t [engine_db_options](#) (void)
- int32_t [running_on_jit](#) (void)
- static force_inline uint32_t [count_match](#) (__Signature sig)
- static force_inline uint32_t [matches](#) (__Signature sig)
- static force_inline uint32_t [match_location](#) (__Signature sig, uint32_t goback)
- static force_inline int32_t [match_location_check](#) (__Signature sig, uint32_t goback, const char *static_start, uint32_t static_len)

1.5.1 Detailed Description

1.5.2 Function Documentation

1.5.2.1 static force_inline uint32_t count_match (__Signature *sig*) [static]

Returns how many times the specified signature matched.

Parameters

in	<i>sig</i>	name of subsignature queried
----	------------	------------------------------

Returns

number of times this subsignature matched in the entire file

This is a constant-time operation, the counts for all subsignatures are already computed.

1.5.2.2 uint32_t engine_db_options (void)

Returns the current engine's db options.

Returns

CL_DB_* flags

1.5.2.3 uint32_t engine_dconf_level (void)

Returns the current engine (dconf) functionality level. Usually identical to [engine_functionality_level\(\)](#), unless distro backported patches. Compare with [FunctionalityLevels](#).

Returns

an integer representing the DCONF (security fixes) level.

1.5.2.4 uint32_t engine_functionality_level (void)

Returns the current engine (feature) functionality level. To map these to ClamAV releases, compare it with [FunctionalityLevels](#).

Returns

an integer representing current engine functionality level.

1.5.2.5 uint32_t engine_scan_options (void)

Returns the current engine's scan options.

Returns

CL_SCAN* flags

1.5.2.6 static force_inline uint32_t match_location (__Signature sig, uint32_t goback) [static]

Returns the offset of the match.

Parameters

in	<i>sig</i>	- Signature
in	<i>goback</i>	- max length of signature

Returns

offset of match

1.5.2.7 static force_inline int32_t match_location_check (__Signature sig, uint32_t goback, const char * static_start, uint32_t static_len) [static]

Like [match_location\(\)](#), but also checks that the match starts with the specified hex string.

It is recommended to use this for safety and compatibility with 0.96.1

Parameters

in	<i>sig</i>	- signature
in	<i>goback</i>	- maximum length of signature (till start of last subsig)
in	<i>static_start</i>	- static string that sig must begin with
in	<i>static_len</i>	- static string that sig must begin with - length

Returns

>=0 - offset of match

-1 - no match

1.5.2.8 static force_inline uint32_t matches (__Signature sig) [static]

Returns whether the specified subsignature has matched at least once.

Parameters

in	<i>sig</i>	name of subsignature queried
----	------------	------------------------------

Returns

1 if subsignature one or more times, 0 otherwise

1.5.2.9 int32_t running_on_jit (void)

Returns whether running on JIT. As side-effect it disables interp / JIT comparisons in test mode (errors are still checked)

Returns

1 - running on JIT

0 - running on ClamAV interpreter

1.6 Environment

Functions

- uint32_t [get_environment](#) (struct cli_environment *env, uint32_t len)
- uint32_t [disable_bytecode_if](#) (const int8_t *reason, uint32_t len, uint32_t cond)
- uint32_t [disable_jit_if](#) (const int8_t *reason, uint32_t len, uint32_t cond)
- int32_t [version_compare](#) (const uint8_t *lhs, uint32_t lhs_len, const uint8_t *rhs, uint32_t rhs_len)
- uint32_t [check_platform](#) (uint32_t a, uint32_t b, uint32_t c)
- bool [__is_bigendian](#) (void) [__attribute__\(\(const\)\)](#) [__attribute__\(\(nothrow\)\)](#)
- static uint32_t force_inline [le32_to_host](#) (uint32_t v)
- static uint32_t force_inline [be32_to_host](#) (uint32_t v)
- static uint64_t force_inline [le64_to_host](#) (uint64_t v)
- static uint64_t force_inline [be64_to_host](#) (uint64_t v)
- static uint16_t force_inline [le16_to_host](#) (uint16_t v)
- static uint16_t force_inline [be16_to_host](#) (uint16_t v)
- static uint32_t force_inline [cli_readint32](#) (const void *buff)
- static uint16_t force_inline [cli_readint16](#) (const void *buff)
- static void force_inline [cli_writeint32](#) (void *offset, uint32_t v)

1.6.1 Detailed Description

1.6.2 Function Documentation

1.6.2.1 bool [__is_bigendian](#) (void) const

Returns true if the bytecode is executing on a big-endian CPU.

Returns

true if executing on bigendian CPU, false otherwise

This will be optimized away in libclamav, but it must be used when dealing with endianness for portability reasons.

For example whenever you read a 32-bit integer from a file, it can be written in little-endian convention (x86 CPU for example), or big-endian convention (PowerPC CPU for example).

If the file always contains little-endian integers, then conversion might be needed.

ClamAV bytecodes by their nature must only handle known-endian integers, if endianness can change, then both situations must be taken into account (based on a 1-byte field for example).

1.6.2.2 static uint16_t force_inline [be16_to_host](#) (uint16_t v) [static]

Converts the specified value if needed, knowing it is in big endian order.

Parameters

in	v	16-bit integer as read from a file
----	---	------------------------------------

Returns

integer converted to host's endianness

1.6.2.3 static uint32_t force_inline [be32_to_host](#) (uint32_t v) [static]

Converts the specified value if needed, knowing it is in big endian order.

Parameters

<i>in</i>	<i>v</i>	32-bit integer as read from a file
-----------	----------	------------------------------------

Returns

integer converted to host's endianness

1.6.2.4 `static uint64_t force_inline be64_to_host (uint64_t v) [static]`

Converts the specified value if needed, knowing it is in big endian order.

Parameters

<i>in</i>	<i>v</i>	64-bit integer as read from a file
-----------	----------	------------------------------------

Returns

integer converted to host's endianness

1.6.2.5 `uint32_t check_platform (uint32_t a, uint32_t b, uint32_t c)`

Disables the JIT if the platform id matches. 0xff can be used instead of a field to mark ANY.

Parameters

<i>in</i>	<i>a</i>	- os_category << 24 arch << 20 compiler << 16 flevel << 8 dconf
<i>in</i>	<i>b</i>	- big_endian << 28 sizeof_ptr << 24 cpp_version
<i>in</i>	<i>c</i>	- os_features << 24 c_version

Returns

0 - no match
1 - match

1.6.2.6 `static uint16_t force_inline cli_readint16 (const void * buff) [static]`

Reads from the specified buffer a 16-bit of little-endian integer.

Parameters

<i>in</i>	<i>buff</i>	pointer to buffer
-----------	-------------	-------------------

Returns

16-bit little-endian integer converted to host endianness

1.6.2.7 `static uint32_t force_inline cli_readint32 (const void * buff) [static]`

Reads from the specified buffer a 32-bit of little-endian integer.

Parameters

<i>in</i>	<i>buff</i>	pointer to buffer
-----------	-------------	-------------------

Returns

32-bit little-endian integer converted to host endianness

1.6.2.8 `static void force_inline cli_writeint32 (void * offset, uint32_t v) [static]`

Writes the specified value into the specified buffer in little-endian order

Parameters

out	offset	pointer to buffer to write to
in	v	value to write

1.6.2.9 uint32_t disable_bytecode_if (const int8_t * reason, uint32_t len, uint32_t cond)

Disables the bytecode completely if condition is true. Can only be called from the BC_STARTUP bytecode.

Parameters

in	reason	- why the bytecode had to be disabled
in	len	- length of reason
in	cond	- condition

Returns

- 0 - auto mode
- 1 - JIT disabled
- 2 - fully disabled

1.6.2.10 uint32_t disable_jit_if (const int8_t * reason, uint32_t len, uint32_t cond)

Disables the JIT completely if condition is true. Can only be called from the BC_STARTUP bytecode.

Parameters

in	reason	- why the JIT had to be disabled
in	len	- length of reason
in	cond	- condition

Returns

- 0 - auto mode
- 1 - JIT disabled
- 2 - fully disabled

1.6.2.11 uint32_t get_environment (struct cli_environment * env, uint32_t len)

Queries the environment this bytecode runs in. Used by BC_STARTUP to disable bytecode when bugs are known for the current platform.

Parameters

out	env	- the full environment
in	len	- size of env

Returns

- 0

1.6.2.12 static uint16_t force_inline le16_to_host (uint16_t v) [static]

Converts the specified value if needed, knowing it is in little endian order.

Parameters

<i>in</i>	<i>v</i>	16-bit integer as read from a file
-----------	----------	------------------------------------

Returns

integer converted to host's endianness

1.6.2.13 `static uint32_t force_inline le32_to_host (uint32_t v) [static]`

Converts the specified value if needed, knowing it is in little endian order.

Parameters

<i>in</i>	<i>v</i>	32-bit integer as read from a file
-----------	----------	------------------------------------

Returns

integer converted to host's endianness

1.6.2.14 `static uint64_t force_inline le64_to_host (uint64_t v) [static]`

Converts the specified value if needed, knowing it is in little endian order.

Parameters

<i>in</i>	<i>v</i>	64-bit integer as read from a file
-----------	----------	------------------------------------

Returns

integer converted to host's endianness

1.6.2.15 `int32_t version_compare (const uint8_t * lhs, uint32_t lhs_len, const uint8_t * rhs, uint32_t rhs_len)`

Compares two version numbers.

Parameters

<i>in</i>	<i>lhs</i>	- left hand side of comparison
<i>in</i>	<i>lhs_len</i>	- length of <i>lhs</i>
<i>in</i>	<i>rhs</i>	- right hand side of comparison
<i>in</i>	<i>rhs_len</i>	- length of <i>rhs</i>

Returns

-1 - *lhs* < *rhs*
 0 - *lhs* == *rhs*
 1 - *lhs* > *rhs*

1.7 File Operations

Enumerations

- enum { [SEEK_SET](#) =0, [SEEK_CUR](#), [SEEK_END](#) }

Functions

- int32_t [read](#) (uint8_t *data, int32_t size)
- int32_t [write](#) (uint8_t *data, int32_t size)
- int32_t [seek](#) (int32_t pos, uint32_t whence)
- int32_t [file_find](#) (const uint8_t *data, uint32_t len)
- int32_t [file_byteat](#) (uint32_t offset)
- int32_t [fill_buffer](#) (uint8_t *buffer, uint32_t len, uint32_t filled, uint32_t cursor, uint32_t fill)
- int32_t [read_number](#) (uint32_t radix)
- int32_t [file_find_limit](#) (const uint8_t *data, uint32_t len, int32_t maxpos)
- int32_t [get_file_reliability](#) (void)
- static force_inline uint32_t [getFilesize](#) (void)

1.7.1 Detailed Description

1.7.2 Enumeration Type Documentation

1.7.2.1 anonymous enum

Enumerator

- [SEEK_SET](#)** set file position to specified absolute position
- [SEEK_CUR](#)** set file position relative to current position
- [SEEK_END](#)** set file position relative to file end

1.7.3 Function Documentation

1.7.3.1 int32_t file_byteat (uint32_t offset)

Read a single byte from current file

Parameters

in	<i>offset</i>	file offset
----	---------------	-------------

Returns

byte at offset *off* in the current file, or -1 if offset is invalid

1.7.3.2 int32_t file_find (const uint8_t * data, uint32_t len)

Looks for the specified sequence of bytes in the current file.

Parameters

in	<i>data</i>	the sequence of bytes to look for
in	<i>len</i>	length of <i>data</i> , cannot be more than 1024

Returns

offset in the current file if match is found, -1 otherwise

1.7.3.3 `int32_t file_find_limit (const uint8_t * data, uint32_t len, int32_t maxpos)`

Looks for the specified sequence of bytes in the current file, up to the specified position.

Parameters

in	<i>data</i>	the sequence of bytes to look for
in	<i>len</i>	length of <i>data</i> , cannot be more than 1024
in	<i>maxpos</i>	maximum position to look for a match, note that this is 1 byte after the end of last possible match: $\text{match_pos} + \text{len} < \text{maxpos}$

Returns

offset in the current file if match is found, -1 otherwise

1.7.3.4 `int32_t fill_buffer (uint8_t * buffer, uint32_t len, uint32_t filled, uint32_t cursor, uint32_t fill)`

Fills the specified buffer with at least `fill` bytes.

Parameters

out	<i>buffer</i>	the buffer to fill
in	<i>len</i>	length of buffer
in	<i>filled</i>	how much of the buffer is currently filled
in	<i>cursor</i>	position of cursor in buffer
in	<i>fill</i>	amount of bytes to fill in (0 is valid)

Returns

<0 on error

0 on EOF

number bytes available in buffer (starting from 0)

The character at the cursor will be at position 0 after this call.

1.7.3.5 `int32_t get_file_reliability (void)`

Get file reliability flag, higher value means less reliable. When >0 import tables and such are not reliable

Returns

0 - normal

1 - embedded PE

2 - unpacker created file (not impl. yet)

1.7.3.6 `static force_inline uint32_t getFileSize (void) [static]`

Returns the currently scanned file's size.

Returns

file size as 32-bit unsigned integer

1.7.3.7 `int32_t read (uint8_t * data, int32_t size)`

Reads specified amount of bytes from the current file into a buffer. Also moves current position in the file.

Parameters

in	<i>size</i>	amount of bytes to read
----	-------------	-------------------------

<i>out</i>	<i>data</i>	pointer to buffer where data is read into
------------	-------------	-------------------------------------------

Returns

amount read.

1.7.3.8 int32_t read_number (uint32_t radix)

Reads a number in the specified radix starting from the current position. Non-numeric characters are ignored.

Parameters

<i>in</i>	<i>radix</i>	10 or 16
-----------	--------------	----------

Returns

the number read

1.7.3.9 int32_t seek (int32_t pos, uint32_t whence)

Changes the current file position to the specified one.

See Also

[SEEK_SET](#), [SEEK_CUR](#), [SEEK_END](#)

Parameters

<i>in</i>	<i>pos</i>	offset (absolute or relative depending on <i>whence</i> param)
<i>in</i>	<i>whence</i>	one of SEEK_SET, SEEK_CUR, SEEK_END

Returns

absolute position in file

1.7.3.10 int32_t write (uint8_t * data, int32_t size)

Writes the specified amount of bytes from a buffer to the current temporary file.

Parameters

<i>in</i>	<i>data</i>	pointer to buffer of data to write
<i>in</i>	<i>size</i>	amount of bytes to write <i>size</i> bytes to temporary file, from the buffer pointed to byte

Returns

amount of bytes successfully written

1.8 Global Variables

Variables

- `const uint32_t __clambc_match_counts [64]`
This is a low-level variable, use the Macros in [bytecode_local.h](#) instead to access it.
- `const uint32_t __clambc_match_offsets [64]`
This is a low-level variable, use the Macros in [bytecode_local.h](#) instead to access it.
- `const struct cli_pe_hook_data __clambc_pedata`
- `const uint32_t __clambc_filesize [1]`
- `const uint16_t __clambc_kind`

1.8.1 Detailed Description

1.8.2 Variable Documentation

1.8.2.1 `const uint32_t __clambc_filesize[1]`

File size (max 4G).

1.8.2.2 `const uint16_t __clambc_kind`

Kind of the bytecode, affects LibClamAV usage

1.8.2.3 `const uint32_t __clambc_match_counts[64]`

This is a low-level variable, use the Macros in [bytecode_local.h](#) instead to access it.

Logical signature match counts

1.8.2.4 `const uint32_t __clambc_match_offsets[64]`

This is a low-level variable, use the Macros in [bytecode_local.h](#) instead to access it.

Logical signature match offsets

1.8.2.5 `const struct cli_pe_hook_data __clambc_pedata`

PE data, if this is a PE hook.

1.9 JavaScript Normalization

Functions

- `int32_t jsnorm_init (int32_t from_buffer)`
- `int32_t jsnorm_process (int32_t id)`
- `int32_t jsnorm_done (int32_t id)`

1.9.1 Detailed Description

1.9.2 Function Documentation

1.9.2.1 `int32_t jsnorm_done (int32_t id)`

Flushes JS normalizer.

Parameters

<i>in</i>	<i>id</i>	ID of js normalizer to flush
-----------	-----------	------------------------------

Returns

0 on success, <0 on failure

1.9.2.2 `int32_t jsnorm_init (int32_t from_buffer)`

Initializes JS normalizer for reading 'from_buffer'. Normalized JS will be written to a single tempfile, one normalized JS per line, and automatically scanned when the bytecode finishes execution.

Parameters

<i>in</i>	<i>from_buffer</i>	ID of buffer_pipe to read javascript from
-----------	--------------------	-------------------------------------------

Returns

ID of JS normalizer, <0 on failure

1.9.2.3 `int32_t jsnorm_process (int32_t id)`

Normalize all javascript from the input buffer, and write to tempfile. You can call this function repeatedly on success, if you (re)fill the input buffer.

Parameters

<i>in</i>	<i>id</i>	ID of JS normalizer
-----------	-----------	---------------------

Returns

0 on success, <0 on failure

1.10 JSON Querying

Enumerations

- enum [bc_json_type](#)

Functions

- `int32_t json_is_active` (void)
- `int32_t json_get_object` (const `int8_t *name`, `int32_t name_len`, `int32_t objid`)
- `int32_t json_get_type` (`int32_t objid`)
- `int32_t json_get_array_length` (`int32_t objid`)
- `int32_t json_get_array_idx` (`int32_t idx`, `int32_t objid`)
- `int32_t json_get_string_length` (`int32_t objid`)
- `int32_t json_get_string` (`int8_t *str`, `int32_t str_len`, `int32_t objid`)
- `int32_t json_get_boolean` (`int32_t objid`)
- `int32_t json_get_int` (`int32_t objid`)

1.10.1 Detailed Description

1.10.2 Enumeration Type Documentation

1.10.2.1 enum `bc_json_type`

JSON types

1.10.3 Function Documentation

1.10.3.1 `int32_t json_get_array_idx (int32_t idx, int32_t objid)`

Returns

objid of json object at idx of json array of objid
 0 if invalid idx
 -1 if an error has occurred
 -2 if object is not JSON_TYPE_ARRAY

Parameters

<code>in</code>	<code>idx</code>	- index of array to query, must be ≥ 0 and less than array length
<code>in</code>	<code>objid</code>	- id value of json object (should be JSON_TYPE_ARRAY) to query

1.10.3.2 `int32_t json_get_array_length (int32_t objid)`

Returns

number of elements in the json array of objid
 -1 if an error has occurred
 -2 if object is not JSON_TYPE_ARRAY

Parameters

<code>in</code>	<code>objid</code>	- id value of json object (should be JSON_TYPE_ARRAY) to query
-----------------	--------------------	----------------------------------------------------------------

1.10.3.3 int32_t json_get_boolean (int32_t objid)

Returns

boolean value of queried objid; will force other types to boolean

Parameters

in	objid	- id value of json object to query
----	-------	------------------------------------

1.10.3.4 int32_t json_get_int (int32_t objid)

Returns

integer value of queried objid; will force other types to integer

Parameters

in	objid	- id value of json object to query
----	-------	------------------------------------

1.10.3.5 int32_t json_get_object (const int8_t * name, int32_t name_len, int32_t objid)

Returns

objid of json object with specified name
 0 if json object of specified name cannot be found
 -1 if an error has occurred

Parameters

in	name	- name of object in ASCII
in	name_len	- length of specified name (not including terminating NULL), must be >= 0
in	objid	- id value of json object to query

1.10.3.6 int32_t json_get_string (int8_t * str, int32_t str_len, int32_t objid)

Returns

number of characters transferred (capped by str_len), including terminating null-character
 -1 if an error has occurred
 -2 if object is not JSON_TYPE_STRING

Parameters

out	str	- user location to store string data; will be null-terminated
in	str_len	- length of str or limit of string data to read, including terminating null-character
in	objid	- id value of json object (should be JSON_TYPE_STRING) to query

1.10.3.7 int32_t json_get_string_length (int32_t objid)

Returns

length of json string of objid, not including terminating null-character
 -1 if an error has occurred
 -2 if object is not JSON_TYPE_STRING

Parameters

in	<i>objid</i>	- id value of json object (should be JSON_TYPE_STRING) to query
----	--------------	-----------------------------------------------------------------

1.10.3.8 int32_t json_get_type (int32_t *objid*)

Returns

type (json_type) of json object specified
-1 if type unknown or invalid id

Parameters

in	<i>objid</i>	- id value of json object to query
----	--------------	------------------------------------

1.10.3.9 int32_t json_is_active (void)

Returns

0 - json is disabled or option not specified
1 - json is active and properties are available

1.11 Icon Matcher

Functions

- `int32_t matchicon (const uint8_t *group1, int32_t group1_len, const uint8_t *group2, int32_t group2_len)`

1.11.1 Detailed Description

1.11.2 Function Documentation

1.11.2.1 `int32_t matchicon (const uint8_t * group1, int32_t group1_len, const uint8_t * group2, int32_t group2_len)`

Attempts to match current executable's icon against the specified icon groups.

Parameters

<code>in</code>	<code>group1</code>	- same as GROUP1 in LDB signatures
<code>in</code>	<code>group1_len</code>	- length of <code>group1</code>
<code>in</code>	<code>group2</code>	- same as GROUP2 in LDB signatures
<code>in</code>	<code>group2_len</code>	- length of <code>group2</code>

Returns

- 1 - invalid call, or sizes (only valid for PE hooks)
- 0 - not a match
- 1 - match

1.12 Math Operation

Functions

- `int32_t ilog2` (`uint32_t a`, `uint32_t b`)
- `int32_t ipow` (`int32_t a`, `int32_t b`, `int32_t c`)
- `uint32_t iexp` (`int32_t a`, `int32_t b`, `int32_t c`)
- `int32_t isin` (`int32_t a`, `int32_t b`, `int32_t c`)
- `int32_t icos` (`int32_t a`, `int32_t b`, `int32_t c`)

1.12.1 Detailed Description

1.12.2 Function Documentation

1.12.2.1 `int32_t icos (int32_t a, int32_t b, int32_t c)`

Returns $c \cdot \cos(a/b)$.

Parameters

<code>in</code>	<code>a</code>	integer
<code>in</code>	<code>b</code>	integer
<code>in</code>	<code>c</code>	integer

Returns

$c \cdot \sin(a/b)$

1.12.2.2 `uint32_t iexp (int32_t a, int32_t b, int32_t c)`

Returns $\exp(a/b) \cdot c$

Parameters

<code>in</code>	<code>a</code>	integer
<code>in</code>	<code>b</code>	integer
<code>in</code>	<code>c</code>	integer

Returns

$c \cdot \exp(a/b)$

1.12.2.3 `int32_t ilog2 (uint32_t a, uint32_t b)`

Returns $2^{26} \cdot \log_2(a/b)$

Parameters

<code>in</code>	<code>a</code>	input
<code>in</code>	<code>b</code>	input

Returns

$2^{26} \cdot \log_2(a/b)$

1.12.2.4 `int32_t ipow (int32_t a, int32_t b, int32_t c)`

Returns $c \cdot a^b$.

Parameters

in	<i>a</i>	integer
in	<i>b</i>	integer
in	<i>c</i>	integer

Returns

$c \cdot \text{pow}(a, b)$

1.12.2.5 `int32_t` `isin (int32_t a, int32_t b, int32_t c)`

Returns $c \cdot \sin(a/b)$.

Parameters

in	<i>a</i>	integer
in	<i>b</i>	integer
in	<i>c</i>	integer

Returns

$c \cdot \sin(a/b)$

1.13 PDF Handling

Enumerations

- enum `pdf_phase` {
`PDF_PHASE_NONE`, `PDF_PHASE_PARSED`, `PDF_PHASE_POSTDUMP`, `PDF_PHASE_END`,
`PDF_PHASE_PRE` }
- enum `pdf_flag`
- enum `pdf_objflags`

Functions

- `int32_t pdf_get_obj_num` (void)
- `int32_t pdf_get_flags` (void)
- `int32_t pdf_set_flags` (int32_t flags)
- `int32_t pdf_lookupobj` (uint32_t id)
- `uint32_t pdf_getobjsize` (int32_t objidx)
- `const uint8_t * pdf_getobj` (int32_t objidx, uint32_t amount)
- `int32_t pdf_getobjid` (int32_t objidx)
- `int32_t pdf_getobjflags` (int32_t objidx)
- `int32_t pdf_setobjflags` (int32_t objidx, int32_t flags)
- `int32_t pdf_get_offset` (int32_t objidx)
- `int32_t pdf_get_phase` (void)
- `int32_t pdf_get_dumpedobjid` (void)

1.13.1 Detailed Description

1.13.2 Enumeration Type Documentation

1.13.2.1 enum `pdf_flag`

PDF flags

1.13.2.2 enum `pdf_objflags`

PDF obj flags

1.13.2.3 enum `pdf_phase`

Phase of PDF parsing used for PDF Hooks

Enumerator

`PDF_PHASE_NONE` not a PDF

`PDF_PHASE_PARSED` after parsing a PDF, object flags can be set etc.

`PDF_PHASE_POSTDUMP` after an obj was dumped and scanned

`PDF_PHASE_END` after the pdf scan finished

`PDF_PHASE_PRE` before pdf is parsed at all

1.13.3 Function Documentation

1.13.3.1 `int32_t pdf_get_dumpedobjid` (void)

Return the currently dumped obj index. Valid only in `PDF_PHASE_POSTDUMP`.

Returns

>=0 - object index
 -1 - invalid phase

1.13.3.2 int32_t pdf_get_flags (void)

Return the flags for the entire PDF (as set so far).

Returns

-1 - if not called from PDF hook
 >=0 - pdf flags

1.13.3.3 int32_t pdf_get_obj_num (void)

Return number of pdf objects

Returns

-1 - if not called from PDF hook
 >=0 - number of PDF objects

1.13.3.4 int32_t pdf_get_offset (int32_t objidx)

Return the object's offset in the PDF.

Parameters

in	<i>objidx</i>	- object index (from 0)
----	---------------	-------------------------

Returns

-1 - object index invalid
 >=0 - offset

1.13.3.5 int32_t pdf_get_phase (void)

Return an 'enum pdf_phase'. Identifies at which phase this bytecode was called.

Returns

the current [pdf_phase](#)

1.13.3.6 const uint8_t* pdf_getobj (int32_t objidx, uint32_t amount)

Return the undecoded object. Meant only for reading, write modifies the fmap buffer, so avoid!

Parameters

in	<i>objidx</i>	- object index (from 0), not object id!
in	<i>amount</i>	- size returned by pdf_getobjsize (or smaller)

Returns

NULL - invalid objidx/amount
 pointer - pointer to original object

1.13.3.7 int32_t pdf_getobjflags (int32_t objidx)

Return the object flags for the specified object index.

Parameters

<i>in</i>	<i>objidx</i>	- object index (from 0)
-----------	---------------	-------------------------

Returns

- 1 - object index invalid
- ≥ 0 - object flags

1.13.3.8 int32_t pdf_getobjid (int32_t objidx)

Return the object id for the specified object index.

Parameters

<i>in</i>	<i>objidx</i>	- object index (from 0)
-----------	---------------	-------------------------

Returns

- 1 - object index invalid
- ≥ 0 - object id (obj id \ll 8 | generation id)

1.13.3.9 uint32_t pdf_getobjsize (int32_t objidx)

Return the size of the specified PDF obj.

Parameters

<i>in</i>	<i>objidx</i>	- object index (from 0), not object id!
-----------	---------------	-----------------------------------------

Returns

- 0 - if not called from PDF hook, or invalid objnum
- ≥ 0 - size of object

1.13.3.10 int32_t pdf_lookupobj (uint32_t id)

Lookup pdf object with specified id.

Parameters

<i>in</i>	<i>id</i>	- pdf id (objnumber \ll 8 generationid)
-----------	-----------	---------------------------------------------

Returns

- 1 - if object id doesn't exist
- ≥ 0 - object index

1.13.3.11 int32_t pdf_set_flags (int32_t flags)

Sets the flags for the entire PDF. It is recommended that you retrieve old flags, and just add new ones.

Parameters

<i>in</i>	<i>flags</i>	- flags to set.
-----------	--------------	-----------------

Returns

- 0 - success -1 - invalid phase

1.13.3.12 int32_t pdf_setobjflags (int32_t *objidx*, int32_t *flags*)

Sets the object flags for the specified object index. This can be used to force dumping of a certain obj, by setting the OBJ_FORCEDUMP flag for example.

Parameters

<i>in</i>	<i>objidx</i>	- object index (from 0)
<i>in</i>	<i>flags</i>	- value to set flags

Returns

-1 - object index invalid
≥0 - flags set

1.14 PE Operations

Data Structures

- struct [cli_exe_section](#)
- struct [cli_exe_info](#)
- struct [pe_image_file_hdr](#)
- struct [pe_image_data_dir](#)
- struct [pe_image_optional_hdr32](#)
- struct [pe_image_optional_hdr64](#)
- struct [pe_image_section_hdr](#)
- struct [cli_pe_hook_data](#)

Functions

- uint32_t [pe_rawaddr](#) (uint32_t rva)
- int32_t [get_pe_section](#) (struct [cli_exe_section](#) *section, uint32_t num)
- static force_inline bool [hasExeInfo](#) (void)
- static force_inline bool [hasPEInfo](#) (void)
- static force_inline bool [isPE64](#) (void)
- static force_inline uint8_t [getPEMajorLinkerVersion](#) (void)
- static force_inline uint8_t [getPEMinorLinkerVersion](#) (void)
- static force_inline uint32_t [getPESizeOfCode](#) (void)
- static force_inline uint32_t [getPESizeOfInitializedData](#) (void)
- static force_inline uint32_t [getPESizeOfUninitializedData](#) (void)
- static force_inline uint32_t [getPEBaseOfCode](#) (void)
- static force_inline uint32_t [getPEBaseOfData](#) (void)
- static force_inline uint64_t [getPEImageBase](#) (void)
- static force_inline uint32_t [getPESectionAlignment](#) (void)
- static force_inline uint32_t [getPEFileAlignment](#) (void)
- static force_inline uint16_t [getPEMajorOperatingSystemVersion](#) (void)
- static force_inline uint16_t [getPEMinorOperatingSystemVersion](#) (void)
- static force_inline uint16_t [getPEMajorImageVersion](#) (void)
- static force_inline uint16_t [getPEMinorImageVersion](#) (void)
- static force_inline uint16_t [getPEMajorSubsystemVersion](#) (void)
- static force_inline uint16_t [getPEMinorSubsystemVersion](#) (void)
- static force_inline uint32_t [getPEWin32VersionValue](#) (void)
- static force_inline uint32_t [getPESizeOfImage](#) (void)
- static force_inline uint32_t [getPESizeOfHeaders](#) (void)
- static force_inline uint32_t [getPEChecksum](#) (void)
- static force_inline uint16_t [getPESubsystem](#) (void)
- static force_inline uint16_t [getPEDllCharacteristics](#) (void)
- static force_inline uint32_t [getPESizeOfStackReserve](#) (void)
- static force_inline uint32_t [getPESizeOfStackCommit](#) (void)
- static force_inline uint32_t [getPESizeOfHeapReserve](#) (void)
- static force_inline uint32_t [getPESizeOfHeapCommit](#) (void)
- static force_inline uint32_t [getPELoaderFlags](#) (void)
- static force_inline uint16_t [getPEMachine](#) ()
- static force_inline uint32_t [getPETimeDateStamp](#) ()
- static force_inline uint32_t [getPEPointerToSymbolTable](#) ()
- static force_inline uint32_t [getPENumberOfSymbols](#) ()
- static force_inline uint16_t [getPESizeOfOptionalHeader](#) ()
- static force_inline uint16_t [getPECharacteristics](#) ()
- static force_inline bool [getPEisDLL](#) ()

- static force_inline uint32_t [getPEDirRVA](#) (unsigned n)
- static force_inline uint32_t [getPEDirSize](#) (unsigned n)
- static force_inline uint16_t [getNumberOfSections](#) (void)
- static uint32_t [getPELFANew](#) (void)
- static force_inline int [readPESectionName](#) (unsigned char name[8], unsigned n)
- static force_inline uint32_t [getEntryPoint](#) (void)
- static force_inline uint32_t [getExeOffset](#) (void)
- static force_inline uint32_t [getImageBase](#) (void)
- static uint32_t [getVirtualEntryPoint](#) (void)
- static uint32_t [getSectionRVA](#) (unsigned i)
- static uint32_t [getSectionVirtualSize](#) (unsigned i)
- static force_inline bool [readRVA](#) (uint32_t rva, void *buf, size_t bufsize)

1.14.1 Detailed Description

1.14.2 Function Documentation

1.14.2.1 int32_t get_pe_section (struct cli_exe_section * section, uint32_t num)

Gets information about the specified PE section.

Parameters

out	<i>section</i>	PE section information will be stored here
in	<i>num</i>	PE section number

Returns

- 0 - success
- 1 - failure

1.14.2.2 static force_inline uint32_t getEntryPoint (void) [static]

Returns the offset of the EntryPoint in the executable file.

Returns

offset of EP as 32-bit unsigned integer

1.14.2.3 static force_inline uint32_t getExeOffset (void) [static]

Returns the offset of the executable in the file.

Returns

offset of embedded executable inside file

1.14.2.4 static force_inline uint32_t getImageBase (void) [static]

Returns the ImageBase with the correct endian conversion.

Only works if the bytecode is a PE hook (i.e. you invoked PE_UNPACKER_DECLARE).

Returns

ImageBase of PE file, 0 - for non-PE hook

1.14.2.5 `static force_inline uint16_t getNumberOfSections (void) [static]`

Returns the number of sections in this executable file.

Returns

number of sections as 16-bit unsigned integer

1.14.2.6 `static force_inline uint32_t getPEBaseOfCode (void) [static]`

Return the PE BaseOfCode.

Returns

PE BaseOfCode, or 0 if not in PE hook

1.14.2.7 `static force_inline uint32_t getPEBaseOfData (void) [static]`

Return the PE BaseOfData.

Returns

PE BaseOfData, or 0 if not in PE hook

1.14.2.8 `static force_inline uint16_t getPECharacteristics () [static]`

Returns PE characteristics.

For example you can use this to check whether it is a DLL (0x2000).

Returns

characteristic of PE file, or 0 if not in PE hook

1.14.2.9 `static force_inline uint32_t getPEChecksum (void) [static]`

Return the PE CheckSum.

Returns

PE CheckSum, or 0 if not in PE hook

1.14.2.10 `static force_inline uint32_t getPEDataDirRVA (unsigned n) [static]`

Gets the virtual address of specified image data directory.

Parameters

<code>in</code>	<code>n</code>	image directory requested
-----------------	----------------	---------------------------

Returns

Virtual Address of requested image directory

1.14.2.11 `static force_inline uint32_t getPEDataDirSize (unsigned n) [static]`

Gets the size of the specified image data directory.

Parameters

<i>in</i>	<i>n</i>	image directory requested
-----------	----------	---------------------------

Returns

Size of requested image directory

1.14.2.12 `static force_inline uint16_t getPEDllCharacteristics (void) [static]`

Return the PE DllCharacteristics.

Returns

PE DllCharacteristics, or 0 if not in PE hook

1.14.2.13 `static force_inline uint32_t getPEFileAlignment (void) [static]`

Return the PE FileAlignment.

Returns

PE FileAlignment, or 0 if not in PE hook

1.14.2.14 `static force_inline uint64_t getPEImageBase (void) [static]`

Return the PE ImageBase as 64-bit integer.

Returns

PE ImageBase as 64-bit int, or 0 if not in PE hook

1.14.2.15 `static force_inline bool getPEisDLL () [static]`

Returns whether this is a DLL. Use this only in a PE hook!

Returns

true - the file is a DLL
false - file is not a DLL

1.14.2.16 `static uint32_t getPELFANew (void) [static]`

Gets the offset to the PE header.

Returns

offset to the PE header, or 0 if not in PE hook

1.14.2.17 `static force_inline uint32_t getPELoaderFlags (void) [static]`

Return the PE LoaderFlags.

Returns

PE LoaderFlags or 0 if not in PE hook

1.14.2.18 `static force_inline uint16_t getPEMachine () [static]`

Returns the CPU this executable runs on, see libclamav/pe.c for possible values.

Returns

PE Machine or 0 if not in PE hook

1.14.2.19 `static force_inline uint16_t getPEMajorImageVersion (void) [static]`

Return the PE MajorImageVersion.

Returns

PE MajorImageVersion, or 0 if not in PE hook

1.14.2.20 `static force_inline uint8_t getPEMajorLinkerVersion (void) [static]`

Returns MajorLinkerVersion for this PE file.

Returns

PE MajorLinkerVersion or 0 if not in PE hook

1.14.2.21 `static force_inline uint16_t getPEMajorOperatingSystemVersion (void) [static]`

Return the PE MajorOperatingSystemVersion.

Returns

PE MajorOperatingSystemVersion, or 0 if not in PE hook

1.14.2.22 `static force_inline uint16_t getPEMajorSubsystemVersion (void) [static]`

Return the PE MajorSubsystemVersion.

Returns

PE MajorSubsystemVersion or 0 if not in PE hook

1.14.2.23 `static force_inline uint16_t getPEMinorImageVersion (void) [static]`

Return the PE MinorImageVersion.

Returns

PE MinorImageVersion, or 0 if not in PE hook

1.14.2.24 `static force_inline uint8_t getPEMinorLinkerVersion (void) [static]`

Returns MinorLinkerVersion for this PE file.

Returns

PE MinorLinkerVersion or 0 if not in PE hook

1.14.2.25 `static force_inline uint16_t getPEMinorOperatingSystemVersion (void) [static]`

Return the PE MinorOperatingSystemVersion.

Returns

PE MinorOperatingSystemVersion, or 0 if not in PE hook

1.14.2.26 `static force_inline uint16_t getPEMinorSubsystemVersion (void) [static]`

Return the PE MinorSubsystemVersion.

Returns

PE MinorSubsystemVersion, or 0 if not in PE hook

1.14.2.27 `static force_inline uint32_t getPENumberOfSymbols () [static]`

Returns the PE number of debug symbols

Returns

PE NumberOfSymbols or 0 if not in PE hook

1.14.2.28 `static force_inline uint32_t getPEPointerToSymbolTable () [static]`

Returns pointer to the PE debug symbol table

Returns

PE PointerToSymbolTable or 0 if not in PE hook

1.14.2.29 `static force_inline uint32_t getPESectionAlignment (void) [static]`

Return the PE SectionAlignment.

Returns

PE SectionAlignment, or 0 if not in PE hook

1.14.2.30 `static force_inline uint32_t getPESizeOfCode (void) [static]`

Return the PE SizeOfCode.

Returns

PE SizeOfCode or 0 if not in PE hook

1.14.2.31 `static force_inline uint32_t getPESizeOfHeaders (void) [static]`

Return the PE SizeOfHeaders.

Returns

PE SizeOfHeaders, or 0 if not in PE hook

1.14.2.32 `static force_inline uint32_t getPESizeOfHeapCommit (void) [static]`

Return the PE SizeOfHeapCommit.

Returns

PE SizeOfHeapCommit, or 0 if not in PE hook

1.14.2.33 `static force_inline uint32_t getPESizeOfHeapReserve (void) [static]`

Return the PE SizeOfHeapReserve.

Returns

PE SizeOfHeapReserve, or 0 if not in PE hook

1.14.2.34 `static force_inline uint32_t getPESizeOfImage (void) [static]`

Return the PE SizeOfImage.

Returns

PE SizeOfImage, or 0 if not in PE hook

1.14.2.35 `static force_inline uint32_t getPESizeOfInitializedData (void) [static]`

Return the PE SizeofInitializedData.

Returns

PE SizeOfInitializeData or 0 if not in PE hook

1.14.2.36 `static force_inline uint16_t getPESizeOfOptionalHeader () [static]`

Returns the size of PE optional header.

Returns

size of PE optional header, or 0 if not in PE hook

1.14.2.37 `static force_inline uint32_t getPESizeOfStackCommit (void) [static]`

Return the PE SizeOfStackCommit.

Returns

PE SizeOfStackCommit, or 0 if not in PE hook

1.14.2.38 `static force_inline uint32_t getPESizeOfStackReserve (void) [static]`

Return the PE SizeOfStackReserve.

Returns

PE SizeOfStackReserver, or 0 if not in PE hook

1.14.2.39 `static force_inline uint32_t getPESizeOfUninitializedData (void) [static]`

Return the PE SizeofUninitializedData.

Returns

PE SizeofUninitializedData or 0 if not in PE hook

1.14.2.40 `static force_inline uint16_t getPESubsystem (void) [static]`

Return the PE Subsystem.

Returns

PE subsystem, or 0 if not in PE hook

1.14.2.41 `static force_inline uint32_t getPETimeDateStamp () [static]`

Returns the PE TimeDateStamp from headers

Returns

PE TimeDateStamp or 0 if not in PE hook

1.14.2.42 `static force_inline uint32_t getPEWin32VersionValue (void) [static]`

Return the PE Win32VersionValue.

Returns

PE Win32VersionValue, or 0 if not in PE hook

1.14.2.43 `static uint32_t getSectionRVA (unsigned i) [static]`

Return the RVA of the specified section.

Parameters

<i>i</i>	section index (from 0)
----------	------------------------

Returns

RVA of section, or -1 if invalid

1.14.2.44 `static uint32_t getSectionVirtualSize (unsigned i) [static]`

Return the virtual size of the specified section.

Parameters

<i>i</i>	section index (from 0)
----------	------------------------

Returns

VSZ of section, or -1 if invalid

1.14.2.45 `static uint32_t getVirtualEntryPoint (void) [static]`

The address of the EntryPoint. Use this for matching EP against sections.

Returns

virtual address of EntryPoint, or 0 if not in PE hook

1.14.2.46 `static force_inline bool hasExeInfo (void) [static]`

Returns whether the current file has executable information.

Returns

true if the file has exe info, false otherwise

1.14.2.47 `static force_inline bool hasPEInfo (void) [static]`

Returns whether PE information is available

Returns

true if PE information is available (in PE hooks)

1.14.2.48 `static force_inline bool isPE64 (void) [static]`

Returns whether this is a PE32+ executable.

Returns

true if this is a PE32+ executable

1.14.2.49 `uint32_t pe_rawaddr (uint32_t rva)`

Converts a RVA (Relative Virtual Address) to an absolute PE file offset.

Parameters

in	<i>rva</i>	a rva address from the PE file
----	------------	--------------------------------

Returns

absolute file offset mapped to the *rva*, or PE_INVALID_RVA if the *rva* is invalid.

1.14.2.50 `static force_inline int readPESectionName (unsigned char name[8], unsigned n) [static]`

Read name of requested PE section.

Parameters

out	<i>name</i>	name of PE section
in	<i>n</i>	PE section requested

Returns

0 if successful,
<0 otherwise

1.14.2.51 `static force_inline bool readRVA (uint32_t rva, void * buf, size_t bufsize) [static]`

read the specified amount of bytes from the PE file, starting at the address specified by RVA.

Parameters

in	<i>rva</i>	the Relative Virtual Address you want to read from (will be converted to file offset)
out	<i>buf</i>	destination buffer
in	<i>bufsize</i>	size of buffer

Returns

true on success (full read)
false on any failure

1.15 Scan Control

Functions

- uint32_t [setvirusname](#) (const uint8_t *name, uint32_t len)
- int32_t [extract_new](#) (int32_t id)
- int32_t [bytecode_rt_error](#) (int32_t locationid)
- int32_t [extract_set_container](#) (uint32_t container)
- int32_t [input_switch](#) (int32_t extracted_file)
- static force_inline overloadable_func void [foundVirus](#) (const char *virusname)

1.15.1 Detailed Description

1.15.2 Function Documentation

1.15.2.1 int32_t [bytecode_rt_error](#) (int32_t *locationid*)

Report a runtime error at the specified locationID.

Parameters

in	<i>locationid</i>	(line << 8) (column&0xff)
----	-------------------	-----------------------------

Returns

0

1.15.2.2 int32_t [extract_new](#) (int32_t *id*)

Prepares for extracting a new file, if we've already extracted one it scans it.

Parameters

in	<i>id</i>	an id for the new file (for example position in container)
----	-----------	------------------------------------------------------------

Returns

1 if previous extracted file was infected

1.15.2.3 int32_t [extract_set_container](#) (uint32_t *container*)

Sets the container type for the currently extracted file.

Parameters

in	<i>container</i>	container type (CL_TYPE_*)
----	------------------	----------------------------

Returns

current setting for container (CL_TYPE_ANY default)

1.15.2.4 static force_inline overloadable_func void [foundVirus](#) (const char * *virusname*) [static]

Sets the specified virusname as the virus detected by this bytecode.

Parameters

<i>in</i>	<i>virusname</i>	the name of the virus, excluding the prefix, must be one of the virusnames declared in <code>VIRUSNAMES</code> .
-----------	------------------	------------------------------------------------------------------------------------------------------------------

See Also[VIRUSNAMES](#)**1.15.2.5 int32_t input_switch (int32_t *extracted_file*)**

Toggles the read/seek API to read from the currently extracted file, and back. You must call seek after switching inputs to position the cursor to a valid position.

Parameters

<i>in</i>	<i>extracted_file</i>	1 - switch to reading from extracted file 0 - switch back to original input
-----------	-----------------------	--------------------------------------------------------------------------------

Returns

-1 on error (if no extracted file exists)

0 on success

1.15.2.6 uint32_t setvirusname (const uint8_t * *name*, uint32_t *len*)

Sets the name of the virus found.

Parameters

<i>in</i>	<i>name</i>	the name of the virus
<i>in</i>	<i>len</i>	length of the virusname

Returns

0

1.16 String Operations

Functions

- `int32_t memstr` (`const uint8_t *haystack`, `int32_t haysize`, `const uint8_t *needle`, `int32_t needlesize`)
- `int32_t hex2ui` (`uint32_t hex1`, `uint32_t hex2`)
- `int32_t atoi` (`const uint8_t *str`, `int32_t size`)
- `uint32_t entropy_buffer` (`uint8_t *buffer`, `int32_t size`)
- static force_inline void * `memchr` (`const void *s`, `int c`, `size_t n`)
- void * `memset` (`void *src`, `int c`, `uintptr_t n`) `__attribute__((nothrow)) __attribute__((__nonnull__(1)))`
- void * `memmove` (`void *dst`, `const void *src`, `uintptr_t n`) `__attribute__((nothrow)) __attribute__((__nonnull__(1)))`
- void void * `memcpy` (`void *restrict dst`, `const void *restrict src`, `uintptr_t n`) `__attribute__((nothrow)) __attribute__((__nonnull__(1)))`
- void void int `memcmp` (`const void *s1`, `const void *s2`, `uint32_t n`) `__attribute__((nothrow)) __attribute__((__pure__)) __attribute__((__nonnull__(1)))`

1.16.1 Detailed Description

1.16.2 Function Documentation

1.16.2.1 `int32_t atoi (const uint8_t * str, int32_t size)`

Converts string to positive number.

Parameters

<code>in</code>	<code>str</code>	buffer
<code>in</code>	<code>size</code>	size of <code>str</code>

Returns

>0 string converted to number if possible, -1 on error

1.16.2.2 `uint32_t entropy_buffer (uint8_t * buffer, int32_t size)`

Returns an approximation for the entropy of `buffer`.

Parameters

<code>in</code>	<code>buffer</code>	input buffer
<code>in</code>	<code>size</code>	size of buffer

Returns

entropy estimation * 2²⁶

1.16.2.3 `int32_t hex2ui (uint32_t hex1, uint32_t hex2)`

Returns hexadecimal characters `hex1` and `hex2` converted to 8-bit number.

Parameters

<code>in</code>	<code>hex1</code>	hexadecimal character
-----------------	-------------------	-----------------------

in	hex2	hexadecimal character
----	------	-----------------------

Returns

hex1 hex2 converted to 8-bit integer, -1 on error

1.16.2.4 static force_inline void* memchr (const void * s, int c, size_t n) [static]

Scan the first *n* bytes of the buffer *s*, for the character *c*.

Parameters

in	<i>s</i>	buffer to scan
in	<i>c</i>	character to look for
in	<i>n</i>	size of buffer

Returns

a pointer to the first byte to match, or NULL if not found.

1.16.2.5 void void int memcmp (const void * s1, const void * s2, uint32_t n)

[LLVM Intrinsic] Compares two memory buffers, *s1* and *s2* to length *n*.

Parameters

in	<i>s1</i>	buffer one
in	<i>s2</i>	buffer two
in	<i>n</i>	amount of bytes to copy

Returns

an integer less than, equal to, or greater than zero if the first *n* bytes of *s1* are found, respectively, to be less than, to match, or be greater than the first *n* bytes of *s2*.

1.16.2.6 void void* memcpy (void *restrict dst, const void *restrict src, uintptr_t n)

[LLVM Intrinsic] Copies data between two non-overlapping buffers, from *src* to *dst* to length *n*.

Parameters

out	<i>dst</i>	destination buffer
in	<i>src</i>	source buffer
in	<i>n</i>	amount of bytes to copy

Returns

dst

1.16.2.7 void* memmove (void * dst, const void * src, uintptr_t n)

[LLVM Intrinsic] Copies data between overlapping buffers, from *src* to *dst* to length *n*.

Parameters

out	<i>dst</i>	destination buffer
in	<i>src</i>	source buffer
in	<i>n</i>	amount of bytes to copy

Returns

dst

1.16.2.8 void* memset (void * *src*, int *c*, uintptr_t *n*)

[LLVM Intrinsic] Fills *src* location with *c* up to length *n*.

Parameters

out	<i>src</i>	pointer to buffer
in	<i>c</i>	character to fill buffer with
in	<i>n</i>	length of buffer

Returns

src

1.16.2.9 int32_t memstr (const uint8_t * *haystack*, int32_t *haysize*, const uint8_t * *needle*, int32_t *needlesize*)

Return position of match, -1 otherwise.

Parameters

in	<i>haystack</i>	buffer to search
in	<i>haysize</i>	size of <i>haystack</i>
in	<i>needle</i>	substring to search
in	<i>needlesize</i>	size of <i>needle</i>

Returns

location of match, -1 otherwise

2 Data Structure Documentation

2.1 cli_exe_info Struct Reference

Data Fields

- struct [cli_exe_section](#) * [section](#)
- uint32_t [offset](#)
- uint32_t [ep](#)
- uint16_t [nsections](#)
- uint32_t [res_addr](#)
- uint32_t [hdr_size](#)

2.1.1 Detailed Description

Executable file information

2.1.2 Field Documentation

2.1.2.1 uint32_t ep

Entrypoint of executable

2.1.2.2 uint32_t hdr_size

Address size - PE ONLY

2.1.2.3 uint16_t nsections

Number of sections

2.1.2.4 uint32_t offset

Offset where this executable start in file (nonzero if embedded)

2.1.2.5 uint32_t res_addr

Resources RVA - PE ONLY

2.1.2.6 struct cli_exe_section* section

Information about all the sections of this file. This array has `nsection` elements

2.2 cli_exe_section Struct Reference

Data Fields

- uint32_t [rva](#)
- uint32_t [vsz](#)
- uint32_t [raw](#)
- uint32_t [rsz](#)
- uint32_t [chr](#)
- uint32_t [urva](#)
- uint32_t [uvsz](#)
- uint32_t [uraw](#)
- uint32_t [ursz](#)

2.2.1 Detailed Description

Section of executable file.

2.2.2 Field Documentation

2.2.2.1 uint32_t chr

Section characteristics

2.2.2.2 uint32_t raw

Raw offset (in file)

2.2.2.3 uint32_t rsz

Raw size (in file)

2.2.2.4 uint32_t rva

Relative VirtualAddress

2.2.2.5 uint32_t uraw

PE - unaligned PointerToRawData

2.2.2.6 uint32_t ursz

PE - unaligned SizeOfRawData

2.2.2.7 uint32_t urva

PE - unaligned VirtualAddress

2.2.2.8 uint32_t uvsz

PE - unaligned VirtualSize

2.2.2.9 uint32_t vsz

VirtualSize

2.3 cli_pe_hook_data Struct Reference

Data Fields

- [uint32_t ep](#)
- [uint16_t nsections](#)
- [struct pe_image_file_hdr file_hdr](#)
- [struct pe_image_optional_hdr32 opt32](#)
- [struct pe_image_optional_hdr64 opt64](#)
- [struct pe_image_data_dir dirs](#) [16]
- [uint32_t e_lfanew](#)
- [uint32_t overlays](#)
- [int32_t overlays_sz](#)
- [uint32_t hdr_size](#)

2.3.1 Detailed Description

Data for the bytecode PE hook

2.3.2 Field Documentation

2.3.2.1 struct `pe_image_data_dir` `dirs[16]`

PE data directory header

2.3.2.2 uint32_t `e_lfanew`

address of new exe header

2.3.2.3 uint32_t `ep`

EntryPoint as file offset

2.3.2.4 struct `pe_image_file_hdr` `file_hdr`

Header for this PE file

2.3.2.5 uint32_t `hdr_size`

internally needed by `rawaddr`

2.3.2.6 uint16_t `nsections`

Number of sections

2.3.2.7 struct `pe_image_optional_hdr32` `opt32`

32-bit PE optional header

2.3.2.8 struct `pe_image_optional_hdr64` `opt64`

64-bit PE optional header

2.3.2.9 uint32_t `overlays`

number of overlays

2.3.2.10 int32_t `overlays_sz`

size of overlays

2.4 DIS_arg Struct Reference

Data Fields

- enum [DIS_ACCESS](#) `access_type`
- enum [DIS_SIZE](#) `access_size`
- struct [DIS_mem_arg](#) `mem`
- enum [X86REGS](#) `reg`
- uint64_t `other`

2.4.1 Detailed Description

Disassembled operand.

2.4.2 Field Documentation

2.4.2.1 enum DIS_SIZE access_size

size of access

2.4.2.2 enum DIS_ACCESS access_type

type of access

2.4.2.3 struct DIS_mem_arg mem

memory operand - member of union 'u'

2.4.2.4 uint64_t other

other operand - member of union 'u'

2.4.2.5 enum X86REGS reg

register operand - member of union 'u'

2.5 DIS_fixed Struct Reference

Data Fields

- enum [X86OPS x86_opcode](#)
- enum [DIS_SIZE operation_size](#)
- enum [DIS_SIZE address_size](#)
- uint8_t [segment](#)
- struct [DIS_arg arg](#) [3]

2.5.1 Detailed Description

Disassembled instruction.

2.5.2 Field Documentation

2.5.2.1 enum DIS_SIZE address_size

size of address

2.5.2.2 struct DIS_arg arg[3]

arguments

2.5.2.3 enum DIS_SIZE operation_size

size of operation

2.5.2.4 uint8_t segment

segment

2.5.2.5 enum X86OPS x86_opcode

opcode of X86 instruction

2.6 DIS_mem_arg Struct Reference

Data Fields

- enum [DIS_SIZE](#) `access_size`
- enum [X86REGS](#) `scale_reg`
- enum [X86REGS](#) `add_reg`
- `uint8_t` `scale`
- `int32_t` `displacement`

2.6.1 Detailed Description

Disassembled memory operand: `scale_reg*scale + add_reg + displacement`.

2.6.2 Field Documentation

2.6.2.1 enum [DIS_SIZE](#) `access_size`

size of access

2.6.2.2 enum [X86REGS](#) `add_reg`

register used as displacement

2.6.2.3 `int32_t` `displacement`

displacement as immediate number

2.6.2.4 `uint8_t` `scale`

scale as immediate number

2.6.2.5 enum [X86REGS](#) `scale_reg`

register used as scale

2.7 DISASM_RESULT Struct Reference

2.7.1 Detailed Description

disassembly result, 64-byte, matched by type-8 signatures

2.8 pe_image_data_dir Struct Reference

2.8.1 Detailed Description

PE data directory header

2.9 pe_image_file_hdr Struct Reference

Data Fields

- `uint32_t` `Magic`
- `uint16_t` `Machine`
- `uint16_t` `NumberOfSections`

- uint32_t [TimeStamp](#)
- uint32_t [PointerToSymbolTable](#)
- uint32_t [NumberOfSymbols](#)
- uint16_t [SizeOfOptionalHeader](#)

2.9.1 Detailed Description

Header for this PE file

2.9.2 Field Documentation

2.9.2.1 uint16_t Machine

CPU this executable runs on, see libclamav/pe.c for possible values

2.9.2.2 uint32_t Magic

PE magic header: PE\0\0

2.9.2.3 uint16_t NumberOfSections

Number of sections in this executable

2.9.2.4 uint32_t NumberOfSymbols

debug

2.9.2.5 uint32_t PointerToSymbolTable

debug

2.9.2.6 uint16_t SizeOfOptionalHeader

== 224

2.9.2.7 uint32_t TimeDateStamp

Unreliable

2.10 pe_image_optional_hdr32 Struct Reference

Data Fields

- uint8_t [MajorLinkerVersion](#)
- uint8_t [MinorLinkerVersion](#)
- uint32_t [SizeOfCode](#)
- uint32_t [SizeOfInitializedData](#)
- uint32_t [SizeOfUninitializedData](#)
- uint32_t [ImageBase](#)
- uint32_t [SectionAlignment](#)
- uint32_t [FileAlignment](#)
- uint16_t [MajorOperatingSystemVersion](#)
- uint16_t [MinorOperatingSystemVersion](#)
- uint16_t [MajorImageVersion](#)
- uint16_t [MinorImageVersion](#)
- uint32_t [Checksum](#)
- uint32_t [NumberOfRvaAndSizes](#)

2.10.1 Detailed Description

32-bit PE optional header

2.10.2 Field Documentation

2.10.2.1 uint32_t CheckSum

NT drivers only

2.10.2.2 uint32_t FileAlignment

usually 32 or 512

2.10.2.3 uint32_t ImageBase

multiple of 64 KB

2.10.2.4 uint16_t MajorImageVersion

unreliable

2.10.2.5 uint8_t MajorLinkerVersion

unreliable

2.10.2.6 uint16_t MajorOperatingSystemVersion

not used

2.10.2.7 uint16_t MinorImageVersion

unreliable

2.10.2.8 uint8_t MinorLinkerVersion

unreliable

2.10.2.9 uint16_t MinorOperatingSystemVersion

not used

2.10.2.10 uint32_t NumberOfRvaAndSizes

unreliable

2.10.2.11 uint32_t SectionAlignment

usually 32 or 4096

2.10.2.12 uint32_t SizeOfCode

unreliable

2.10.2.13 uint32_t SizeOfInitializedData

unreliable

2.10.2.14 uint32_t SizeOfUninitializedData

unreliable

2.11 pe_image_optional_hdr64 Struct Reference

Data Fields

- uint8_t [MajorLinkerVersion](#)
- uint8_t [MinorLinkerVersion](#)
- uint32_t [SizeOfCode](#)
- uint32_t [SizeOfInitializedData](#)
- uint32_t [SizeOfUninitializedData](#)
- uint64_t [ImageBase](#)
- uint32_t [SectionAlignment](#)
- uint32_t [FileAlignment](#)
- uint16_t [MajorOperatingSystemVersion](#)
- uint16_t [MinorOperatingSystemVersion](#)
- uint16_t [MajorImageVersion](#)
- uint16_t [MinorImageVersion](#)
- uint32_t [Checksum](#)
- uint32_t [NumberOfRvaAndSizes](#)

2.11.1 Detailed Description

PE 64-bit optional header

2.11.2 Field Documentation

2.11.2.1 uint32_t CheckSum

NT drivers only

2.11.2.2 uint32_t FileAlignment

usually 32 or 512

2.11.2.3 uint64_t ImageBase

multiple of 64 KB

2.11.2.4 uint16_t MajorImageVersion

unreliable

2.11.2.5 uint8_t MajorLinkerVersion

unreliable

2.11.2.6 uint16_t MajorOperatingSystemVersion

not used

2.11.2.7 uint16_t MinorImageVersion

unreliable

2.11.2.8 uint8_t MinorLinkerVersion

unreliable

2.11.2.9 uint16_t MinorOperatingSystemVersion

not used

2.11.2.10 uint32_t NumberOfRvaAndSizes

unreliable

2.11.2.11 uint32_t SectionAlignment

usually 32 or 4096

2.11.2.12 uint32_t SizeOfCode

unreliable

2.11.2.13 uint32_t SizeOfInitializedData

unreliable

2.11.2.14 uint32_t SizeOfUninitializedData

unreliable

2.12 pe_image_section_hdr Struct Reference

Data Fields

- uint8_t [Name](#) [8]
- uint32_t [SizeOfRawData](#)
- uint32_t [PointerToRawData](#)
- uint32_t [PointerToRelocations](#)
- uint32_t [PointerToLinenumbers](#)
- uint16_t [NumberOfRelocations](#)
- uint16_t [NumberOfLinenumbers](#)

2.12.1 Detailed Description

PE section header

2.12.2 Field Documentation

2.12.2.1 uint8_t Name[8]

may not end with NULL

2.12.2.2 uint16_t NumberOfLinenumbers

object files only

2.12.2.3 uint16_t NumberOfRelocations

object files only

2.12.2.4 uint32_t PointerToLinenumbers

object files only

2.12.2.5 `uint32_t` `PointerToRawData`

offset to the section's data

2.12.2.6 `uint32_t` `PointerToRelocations`

object files only

2.12.2.7 `uint32_t` `SizeOfRawData`

multiple of `FileAlignment`

3 File Documentation

3.1 `bytecode_api.h` File Reference

Enumerations

- enum `BytecodeKind` {
`BC_GENERIC` = 0, `BC_STARTUP` = 1, `BC_LOGICAL` = 256, `BC_PE_UNPACKER`,
`BC_PDF`, `BC_PE_ALL`, `BC_PRECLASS` }
- enum { `PE_INVALID_RVA` = 0xFFFFFFFF }
- enum `FunctionalityLevels` {
`FUNC_LEVEL_096` = 51, `FUNC_LEVEL_096_1` = 53, `FUNC_LEVEL_096_2` = 54, `FUNC_LEVEL_096_3`
= 55,
`FUNC_LEVEL_096_4` = 56, `FUNC_LEVEL_096_5` = 58, `FUNC_LEVEL_097` = 60, `FUNC_LEVEL_097_1` =
61,
`FUNC_LEVEL_097_2` = 62, `FUNC_LEVEL_097_3` = 63, `FUNC_LEVEL_097_4` = 64, `FUNC_LEVEL_097_5`
= 65,
`FUNC_LEVEL_097_6` = 67, `FUNC_LEVEL_097_7` = 68, `FUNC_LEVEL_097_8` = 69, `FUNC_LEVEL_098_1`
= 76,
`FUNC_LEVEL_098_2` = 77, `FUNC_LEVEL_098_3` = 77, `FUNC_LEVEL_098_4` = 77, `FUNC_LEVEL_098_5`
= 79,
`FUNC_LEVEL_098_6` = 79, `FUNC_LEVEL_098_7` = 80 }
- enum `pdf_phase` {
`PDF_PHASE_NONE`, `PDF_PHASE_PARSED`, `PDF_PHASE_POSTDUMP`, `PDF_PHASE_END`,
`PDF_PHASE_PRE` }
- enum `pdf_flag`
- enum `pdf_objflags`
- enum `bc_json_type`
- enum { `SEEK_SET` = 0, `SEEK_CUR`, `SEEK_END` }

Functions

- `uint32_t test1` (`uint32_t` a, `uint32_t` b)
- `int32_t read` (`uint8_t` *data, `int32_t` size)
- `int32_t write` (`uint8_t` *data, `int32_t` size)
- `int32_t seek` (`int32_t` pos, `uint32_t` whence)
- `uint32_t setvirusname` (const `uint8_t` *name, `uint32_t` len)
- `uint32_t debug_print_str` (const `uint8_t` *str, `uint32_t` len)
- `uint32_t debug_print_uint` (`uint32_t` a)
- `uint32_t disasm_x86` (struct `DISASM_RESULT` *result, `uint32_t` len)
- `uint32_t pe_rawaddr` (`uint32_t` rva)
- `int32_t file_find` (const `uint8_t` *data, `uint32_t` len)
- `int32_t file_byteat` (`uint32_t` offset)

- void * [malloc](#) (uint32_t size)
- uint32_t [test2](#) (uint32_t a)
- int32_t [get_pe_section](#) (struct [cli_exe_section](#) *section, uint32_t num)
- int32_t [fill_buffer](#) (uint8_t *buffer, uint32_t len, uint32_t filled, uint32_t cursor, uint32_t fill)
- int32_t [extract_new](#) (int32_t id)
- int32_t [read_number](#) (uint32_t radix)
- int32_t [hashset_new](#) (void)
- int32_t [hashset_add](#) (int32_t hs, uint32_t key)
- int32_t [hashset_remove](#) (int32_t hs, uint32_t key)
- int32_t [hashset_contains](#) (int32_t hs, uint32_t key)
- int32_t [hashset_done](#) (int32_t id)
- int32_t [hashset_empty](#) (int32_t id)
- int32_t [buffer_pipe_new](#) (uint32_t size)
- int32_t [buffer_pipe_new_fromfile](#) (uint32_t pos)
- uint32_t [buffer_pipe_read_avail](#) (int32_t id)
- const uint8_t * [buffer_pipe_read_get](#) (int32_t id, uint32_t amount)
- int32_t [buffer_pipe_read_stopped](#) (int32_t id, uint32_t amount)
- uint32_t [buffer_pipe_write_avail](#) (int32_t id)
- uint8_t * [buffer_pipe_write_get](#) (int32_t id, uint32_t size)
- int32_t [buffer_pipe_write_stopped](#) (int32_t id, uint32_t amount)
- int32_t [buffer_pipe_done](#) (int32_t id)
- int32_t [inflate_init](#) (int32_t from_buffer, int32_t to_buffer, int32_t windowBits)
- int32_t [inflate_process](#) (int32_t id)
- int32_t [inflate_done](#) (int32_t id)
- int32_t [bytecode_rt_error](#) (int32_t locationid)
- int32_t [jsnorm_init](#) (int32_t from_buffer)
- int32_t [jsnorm_process](#) (int32_t id)
- int32_t [jsnorm_done](#) (int32_t id)
- int32_t [ilog2](#) (uint32_t a, uint32_t b)
- int32_t [ipow](#) (int32_t a, int32_t b, int32_t c)
- uint32_t [iexp](#) (int32_t a, int32_t b, int32_t c)
- int32_t [isin](#) (int32_t a, int32_t b, int32_t c)
- int32_t [icos](#) (int32_t a, int32_t b, int32_t c)
- int32_t [memstr](#) (const uint8_t *haystack, int32_t haysize, const uint8_t *needle, int32_t needlesize)
- int32_t [hex2ui](#) (uint32_t hex1, uint32_t hex2)
- int32_t [atoi](#) (const uint8_t *str, int32_t size)
- uint32_t [debug_print_str_start](#) (const uint8_t *str, uint32_t len)
- uint32_t [debug_print_str_nonl](#) (const uint8_t *str, uint32_t len)
- uint32_t [entropy_buffer](#) (uint8_t *buffer, int32_t size)
- int32_t [map_new](#) (int32_t keysize, int32_t valuesize)
- int32_t [map_addkey](#) (const uint8_t *key, int32_t ksize, int32_t id)
- int32_t [map_setvalue](#) (const uint8_t *value, int32_t vsize, int32_t id)
- int32_t [map_remove](#) (const uint8_t *key, int32_t ksize, int32_t id)
- int32_t [map_find](#) (const uint8_t *key, int32_t ksize, int32_t id)
- int32_t [map_getvaluesize](#) (int32_t id)
- uint8_t * [map_getvalue](#) (int32_t id, int32_t size)
- int32_t [map_done](#) (int32_t id)
- int32_t [file_find_limit](#) (const uint8_t *data, uint32_t len, int32_t maxpos)
- uint32_t [engine_functionality_level](#) (void)
- uint32_t [engine_dconf_level](#) (void)
- uint32_t [engine_scan_options](#) (void)
- uint32_t [engine_db_options](#) (void)
- int32_t [extract_set_container](#) (uint32_t container)
- int32_t [input_switch](#) (int32_t extracted_file)
- uint32_t [get_environment](#) (struct [cli_environment](#) *env, uint32_t len)

- `uint32_t disable_bytecode_if` (const `int8_t` *reason, `uint32_t` len, `uint32_t` cond)
- `uint32_t disable_jit_if` (const `int8_t` *reason, `uint32_t` len, `uint32_t` cond)
- `int32_t version_compare` (const `uint8_t` *lhs, `uint32_t` lhs_len, const `uint8_t` *rhs, `uint32_t` rhs_len)
- `uint32_t check_platform` (`uint32_t` a, `uint32_t` b, `uint32_t` c)
- `int32_t pdf_get_obj_num` (void)
- `int32_t pdf_get_flags` (void)
- `int32_t pdf_set_flags` (`int32_t` flags)
- `int32_t pdf_lookupobj` (`uint32_t` id)
- `uint32_t pdf_getobjsize` (`int32_t` objidx)
- const `uint8_t` * `pdf_getobj` (`int32_t` objidx, `uint32_t` amount)
- `int32_t pdf_getobjid` (`int32_t` objidx)
- `int32_t pdf_getobjflags` (`int32_t` objidx)
- `int32_t pdf_setobjflags` (`int32_t` objidx, `int32_t` flags)
- `int32_t pdf_get_offset` (`int32_t` objidx)
- `int32_t pdf_get_phase` (void)
- `int32_t pdf_get_dumpedobjid` (void)
- `int32_t matchicon` (const `uint8_t` *group1, `int32_t` group1_len, const `uint8_t` *group2, `int32_t` group2_len)
- `int32_t running_on_jit` (void)
- `int32_t get_file_reliability` (void)
- `int32_t json_is_active` (void)
- `int32_t json_get_object` (const `int8_t` *name, `int32_t` name_len, `int32_t` objid)
- `int32_t json_get_type` (`int32_t` objid)
- `int32_t json_get_array_length` (`int32_t` objid)
- `int32_t json_get_array_idx` (`int32_t` idx, `int32_t` objid)
- `int32_t json_get_string_length` (`int32_t` objid)
- `int32_t json_get_string` (`int8_t` *str, `int32_t` str_len, `int32_t` objid)
- `int32_t json_get_boolean` (`int32_t` objid)
- `int32_t json_get_int` (`int32_t` objid)

Variables

- const `uint32_t` `__clambc_match_counts` [64]
This is a low-level variable, use the Macros in `bytecode_local.h` instead to access it.
- const `uint32_t` `__clambc_match_offsets` [64]
This is a low-level variable, use the Macros in `bytecode_local.h` instead to access it.
- const struct `cli_pe_hook_data` `__clambc_pedata`
- const `uint32_t` `__clambc_filesize` [1]
- const `uint16_t` `__clambc_kind`

3.1.1 Enumeration Type Documentation

3.1.1.1 anonymous enum

Enumerator

`PE_INVALID_RVA` Invalid RVA specified

3.1.2 Function Documentation

3.1.2.1 `uint32_t test1 (uint32_t a, uint32_t b)`

Test api.

Parameters

<i>in</i>	<i>a</i>	0xf00dbeef
<i>in</i>	<i>b</i>	0xbeeff00d

Returns

0x12345678 if parameters match, 0x55 otherwise

3.1.2.2 uint32_t test2 (uint32_t a)

Test api2.

Parameters

<i>in</i>	<i>a</i>	0xf00d
-----------	----------	--------

Returns

0xd00f if parameter matches, 0x5555 otherwise

3.2 bytecode_disasm.h File Reference**Data Structures**

- struct [DISASM_RESULT](#)

Generated on Wed Mar 4 2015 12:10:55 by Doxygen

- OP_FYL2XP1 }
- enum DIS_ACCESS {
ACCESS_NOARG, ACCESS_IMM, ACCESS_REL, ACCESS_REG,
ACCESS_MEM }
- enum DIS_SIZE {
SIZEB, SIZEW, SIZED, SIZEF,
SIZEQ, SIZET, SIZEPTR }
- enum X86REGS

3.2.1 Enumeration Type Documentation

3.2.1.1 enum DIS_ACCESS

Access type

Enumerator

ACCESS_NOARG arg not present
ACCESS_IMM immediate
ACCESS_REL +/- immediate
ACCESS_REG register
ACCESS_MEM [memory]

3.2.1.2 enum DIS_SIZE

for mem access, immediate and relative

Enumerator

SIZEB Byte size access
SIZEW Word size access
SIZED Doubleword size access
SIZEF 6-byte access (seg+reg pair)
SIZEQ Quadword access
SIZET 10-byte access
SIZEPTR ptr

3.2.1.3 enum X86OPS

X86 opcode

Enumerator

OP_AAA Ascii Adjust after Addition
OP_AAD Ascii Adjust AX before Division
OP_AAM Ascii Adjust AX after Multiply
OP_AAS Ascii Adjust AL after Subtraction
OP_ADD Add
OP_ADC Add with Carry
OP_AND Logical And
OP_ARPL Adjust Requested Privilege Level
OP_BOUND Check Array Index Against Bounds
OP_BSF Bit Scan Forward

OP_BSR Bit Scan Reverse
OP_BSWAP Byte Swap
OP_BT Bit Test
OPBTC Bit Test and Complement
OPBTR Bit Test and Reset
OPBTS Bit Test and Set
OP_CALL Call
OP_CDQ Convert DoubleWord to QuadWord
OP_CWDE Convert Word to DoubleWord
OP_CBW Convert Byte to Word
OP_CLC Clear Carry Flag
OP_CLD Clear Direction Flag
OP_CLI Clear Interrupt Flag
OP_CLTS Clear Task-Switched Flag in CR0
OP_CMC Complement Carry Flag
OP_CMOVO Conditional Move if Overflow
OP_CMOVNO Conditional Move if Not Overflow
OP_CMOVOC Conditional Move if Carry
OP_CMOVNC Conditional Move if Not Carry
OP_CMOVZ Conditional Move if Zero
OP_CMOVNZ Conditional Move if Non-Zero
OP_CMOVBE Conditional Move if Below or Equal
OP_CMOVA Conditional Move if Above
OP_CMOVS Conditional Move if Sign
OP_CMOVNS Conditional Move if Not Sign
OP_CMOVPP Conditional Move if Parity
OP_CMOVNP Conditional Move if Not Parity
OP_CMOVL Conditional Move if Less
OP_CMOVGE Conditional Move if Greater or Equal
OP_CMOVLE Conditional Move if Less than or Equal
OP_CMOVG Conditional Move if Greater
OP_CMP Compare
OP_CMPSD Compare String DoubleWord
OP_CMPSW Compare String Word
OP_CMPSB Compare String Byte
OP_CMPXCHG Compare and Exchange
OP_CMPXCHG8B Compare and Exchange Bytes
OP_CPUID CPU Identification
OP_DAA Decimal Adjust AL after Addition
OP_DAS Decimal Adjust AL after Subtraction
OP_DEC Decrement by 1
OP_DIV Unsigned Divide
OP_ENTER Make Stack Frame for Procedure Parameters
OP_FWAIT Wait
OP_HLT Halt

OP_IDIV Signed Divide
OP_IMUL Signed Multiply
OP_INC Increment by 1
OP_IN INput from port
OP_INSD INput from port to String Doubleword
OP_INSW INput from port to String Word
OP_INSB INput from port to String Byte
OP_INT INTerrupt
OP_INT3 INTerrupt 3 (breakpoint)
OP_INT0 INTerrupt 4 if Overflow
OP_INVD Invalidate Internal Caches
OP_INVLPG Invalidate TLB Entry
OP_IRET Interrupt Return
OP_JO Jump if Overflow
OP_JNO Jump if Not Overflow
OP_JC Jump if Carry
OP_JNC Jump if Not Carry
OP_JZ Jump if Zero
OP_JNZ Jump if Not Zero
OP_JBE Jump if Below or Equal
OP_JA Jump if Above
OP_JS Jump if Sign
OP_JNS Jump if Not Sign
OP_JP Jump if Parity
OP_JNP Jump if Not Parity
OP_JL Jump if Less
OP_JGE Jump if Greater or Equal
OP_JLE Jump if Less or Equal
OP_JG Jump if Greater
OP_JMP Jump (unconditional)
OP_LAHF Load Status Flags into AH Register
OP_LAR load Access Rights Byte
OP_LDS Load Far Pointer into DS
OP_LES Load Far Pointer into ES
OP_LFS Load Far Pointer into FS
OP_LGS Load Far Pointer into GS
OP_LEA Load Effective Address
OP_LEAVE High Level Procedure Exit
OP_LGDT Load Global Descript Table Register
OP_LIDT Load Interrupt Descriptor Table Register
OP_LLDT Load Local Descriptor Table Register
OP_PREFIX_LOCK Assert LOCK# Signal Prefix
OP_LODSD Load String Dword
OP_LODSW Load String Word
OP_LODSB Load String Byte

OP_LOOP Loop According to ECX Counter

OP_LOOPE Loop According to ECX Counter and ZF=1

OP_LOOPNE Loop According to ECX Counter and ZF=0

OP_JECXZ Jump if ECX is Zero

OP_LSL Load Segment Limit

OP_LSS Load Far Pointer into SS

OP_LTR Load Task Register

OP_MOV Move

OP_MOVSD Move Data from String to String Doubleword

OP_MOVSW Move Data from String to String Word

OP_MOVSB Move Data from String to String Byte

OP_MOVSX Move with Sign-Extension

OP_MOVZX Move with Zero-Extension

OP_MUL Unsigned Multiply

OP_NEG Two's Complement Negation

OP_NOP No Operation

OP_NOT One's Complement Negation

OP_OR Logical Inclusive OR

OP_OUT Output to Port

OP_OUTSD Output String to Port Doubleword

OP_OUTSW Output String to Port Word

OP_OUTSB Output String to Port Bytes

OP_PUSH Push Onto the Stack

OP_PUSHAD Push All Double General Purpose Registers

OP_PUSHFD Push EFLAGS Register onto the Stack

OP_POP Pop a Value from the Stack

OP_POPAD Pop All Double General Purpose Registers from the Stack

OP_POPFD Pop Stack into EFLAGS Register

OP_RCL Rotate Carry Left

OP_RCR Rotate Carry Right

OP_RDMSR Read from Model Specific Register

OP_RDPMC Read Performance Monitoring Counters

OP_RDTSC Read Time-Stamp Counter

OP_PREFIX_REPE Repeat String Operation Prefix while Equal

OP_PREFIX_REPNE Repeat String Operation Prefix while Not Equal

OP_RETF Return from Far Procedure

OP_RETN Return from Near Procedure

OP_ROL Rotate Left

OP_ROR Rotate Right

OP_RSM Resume from System Management Mode

OP_SAHF Store AH into Flags

OP_SAR Shift Arithmetic Right

OP_SBB Subtract with Borrow

OP_SCASD Scan String Doubleword

OP_SCASW Scan String Word

OP_SCASB Scan String Byte
OP_SETO Set Byte on Overflow
OP_SETNO Set Byte on Not Overflow
OP_SETC Set Byte on Carry
OP_SETNC Set Byte on Not Carry
OP_SETZ Set Byte on Zero
OP_SETNZ Set Byte on Not Zero
OP_SETBE Set Byte on Below or Equal
OP_SETA Set Byte on Above
OP_SETS Set Byte on Sign
OP_SETNS Set Byte on Not Sign
OP_SETP Set Byte on Parity
OP_SETNP Set Byte on Not Parity
OP_SETL Set Byte on Less
OP_SETGE Set Byte on Greater or Equal
OP_SETLE Set Byte on Less or Equal
OP_SETG Set Byte on Greater
OP_SGDT Store Global Descriptor Table Register
OP_SIDT Store Interrupt Descriptor Table Register
OP_SHL Shift Left
OP_SHLD Double Precision Shift Left
OP_SHR Shift Right
OP_SHRD Double Precision Shift Right
OP_SLDT Store Local Descriptor Table Register
OP_STOSD Store String Doubleword
OP_STOSW Store String Word
OP_STOSB Store String Byte
OP_STR Store Task Register
OP_STC Set Carry Flag
OP_STD Set Direction Flag
OP_STI Set Interrupt Flag
OP_SUB Subtract
OP_SYSCALL Fast System Call
OP_SYSENTER Fast System Call
OP_SYSEXIT Fast Return from Fast System Call
OP_SYSRET Return from Fast System Call
OP_TEST Logical Compare
OP_UD2 Undefined Instruction
OP_VERR Verify a Segment for Reading
OP_VERRW Verify a Segment for Writing
OP_WBINVD Write Back and Invalidate Cache
OP_WRMSR Write to Model Specific Register
OP_XADD Exchange and Add
OP_XCHG Exchange Register/Memory with Register
OP_XLAT Table Look-up Translation

OP_XOR Logical Exclusive OR

OP_FPU FPU operation

OP_F2XM1 Compute $2x-1$

OP_FABS Absolute Value

OP_FADD Floating Point Add

OP_FADDP Floating Point Add, Pop

OP_FBLD Load Binary Coded Decimal

OP_FBSTP Store BCD Integer and Pop

OP_FCHS Change Sign

OP_FCLEX Clear Exceptions

OP_FCMOVB Floating Point Move on Below

OP_FCMOVBE Floating Point Move on Below or Equal

OP_FCMOVE Floating Point Move on Equal

OP_FCMOVNB Floating Point Move on Not Below

OP_FCMOVNBE Floating Point Move on Not Below or Equal

OP_FCMOVNE Floating Point Move on Not Equal

OP_FCMOVNU Floating Point Move on Not Unordered

OP_FCMOVU Floating Point Move on Unordered

OP_FCOM Compare Floating Pointer Values and Set FPU Flags

OP_FCOMI Compare Floating Pointer Values and Set EFLAGS

OP_FCOMIP Compare Floating Pointer Values and Set EFLAGS, Pop

OP_FCOMP Compare Floating Pointer Values and Set FPU Flags, Pop

OP_FCOMPP Compare Floating Pointer Values and Set FPU Flags, Pop Twice

OP_FCOS Cosine

OP_FDECSTP Decrement Stack Top Pointer

OP_FDIV Floating Point Divide

OP_FDIVP Floating Point Divide, Pop

OP_FDIVR Floating Point Reverse Divide

OP_FDIVRP Floating Point Reverse Divide, Pop

OP_FFREE Free Floating Point Register

OP_FIADD Floating Point Add

OP_FICOM Compare Integer

OP_FICOMP Compare Integer, Pop

OP_FIDIV Floating Point Divide by Integer

OP_FIDIVR Floating Point Reverse Divide by Integer

OP_FILD Load Integer

OP_FIMUL Floating Point Multiply with Integer

OP_FINCSTP Increment Stack-Top Pointer

OP_FINIT Initialize Floating-Point Unit

OP_FIST Store Integer

OP_FISTP Store Integer, Pop

OP_FISTTP Store Integer with Truncation

OP_FISUB Floating Point Integer Subtract

OP_FISUBR Floating Point Reverse Integer Subtract

OP_FLD Load Floating Point Value

OP_FLD1 Load Constant 1
OP_FLDCW Load x87 FPU Control Word
OP_FLDENV Load x87 FPU Environment
OP_FLDL2E Load Constant $\log_2(e)$
OP_FLDL2T Load Constant $\log_2(10)$
OP_FDLG2 Load Constant $\log_{10}(2)$
OP_FLDLN2 Load Constant $\log_e(2)$
OP_FLDPI Load Constant PI
OP_FLDZ Load Constant Zero
OP_FMUL Floating Point Multiply
OP_FMULP Floating Point Multiply, Pop
OP_FNOP No Operation
OP_FPATAN Partial Arctangent
OP_FPREM Partial Remainder
OP_FPREM1 Partial Remainder
OP_FPTAN Partial Tangent
OP_FRNDINT Round to Integer
OP_FRSTOR Restore x86 FPU State
OP_FSCALE Scale
OP_FSINCOS Sine and Cosine
OP_FSQRT Square Root
OP_FSAVE Store x87 FPU State
OP_FST Store Floating Point Value
OP_FSTCW Store x87 FPU Control Word
OP_FSTENV Store x87 FPU Environment
OP_FSTP Store Floating Point Value, Pop
OP_FSTSW Store x87 FPU Status Word
OP_FSUB Floating Point Subtract
OP_FSUBP Floating Point Subtract, Pop
OP_FSUBR Floating Point Reverse Subtract
OP_FSUBRP Floating Point Reverse Subtract, Pop
OP_FTST Floating Point Test
OP_FUCOM Floating Point Unordered Compare
OP_FUCOMI Floating Point Unordered Compare with Integer
OP_FUCOMIP Floating Point Unorder Compare with Integer, Pop
OP_FUCOMP Floating Point Unorder Compare, Pop
OP_FUCOMPP Floating Point Unorder Compare, Pop Twice
OP_FXAM Examine ModR/M
OP_FXCH Exchange Register Contents
OP_FXTRACT Extract Exponent and Significand
OP_FYL2X Compute $y \cdot \log_2 x$
OP_FYL2XP1 Compute $y \cdot \log_2(x+1)$

3.2.1.4 enum X86REGS

X86 registers

3.3 bytecode_execs.h File Reference

Data Structures

- struct [cli_exe_section](#)
- struct [cli_exe_info](#)

3.4 bytecode_local.h File Reference

Data Structures

- struct [DIS_mem_arg](#)
- struct [DIS_arg](#)
- struct [DIS_fixed](#)

Macros

- #define [VIRUSNAME_PREFIX](#)(name) const char __clambc_virusname_prefix[] = name;
- #define [VIRUSNAMES](#)(...) const char *const __clambc_virusnames[] = {__VA_ARGS__};
- #define [PE_UNPACKER_DECLARE](#) const uint16_t __clambc_kind = BC_PE_UNPACKER;
- #define [PDF_HOOK_DECLARE](#) const uint16_t __clambc_kind = BC_PDF;
- #define [BYTECODE_ABORT_HOOK](#) 0xcea5e
- #define [PE_HOOK_DECLARE](#) const uint16_t __clambc_kind = BC_PE_ALL;
- #define [PRECLASS_HOOK_DECLARE](#) const uint16_t __clambc_kind = BC_PRECLASS;
- #define [SIGNATURES_DECL_BEGIN](#) struct __Signatures {
- #define [DECLARE_SIGNATURE](#)(name)
- #define [SIGNATURES_DECL_END](#) };
- #define [TARGET](#)(tgt) const unsigned short __Target = (tgt);
- #define [COPYRIGHT](#)(c) const char *const __Copyright = (c);
- #define [ICONGROUP1](#)(group) const char *const __IconGroup1 = (group);
- #define [ICONGROUP2](#)(group) const char *const __IconGroup2 = (group);
- #define [FUNCTIONALITY_LEVEL_MIN](#)(m) const unsigned short __FuncMin = (m);
- #define [FUNCTIONALITY_LEVEL_MAX](#)(m) const unsigned short __FuncMax = (m);
- #define [SIGNATURES_DEF_BEGIN](#)
- #define [SIGNATURES_END](#) };
- #define [SIGNATURES_DEF_END](#) };

Functions

- static force_inline void
overloadable_func [debug](#) (const char *str)
- static force_inline void
overloadable_func [debug](#) (const uint8_t *str)
- static force_inline void
overloadable_func [debug](#) (uint32_t a)
- void [debug](#) (...) __attribute__((overloadable))
- static force_inline uint32_t [count_match](#) (__Signature sig)
- static force_inline uint32_t [matches](#) (__Signature sig)
- static force_inline uint32_t [match_location](#) (__Signature sig, uint32_t goback)
- static force_inline int32_t [match_location_check](#) (__Signature sig, uint32_t goback, const char *static_start, uint32_t static_len)
- static force_inline
overloadable_func void [foundVirus](#) (const char *virusname)

- static force_inline void overloadable_func [foundVirus](#) (void)
- static force_inline uint32_t [getFileSize](#) (void)
- bool [__is_bigendian](#) (void) __attribute__((const)) __attribute__((nothrow))
- static uint32_t force_inline [le32_to_host](#) (uint32_t v)
- static uint32_t force_inline [be32_to_host](#) (uint32_t v)
- static uint64_t force_inline [le64_to_host](#) (uint64_t v)
- static uint64_t force_inline [be64_to_host](#) (uint64_t v)
- static uint16_t force_inline [le16_to_host](#) (uint16_t v)
- static uint16_t force_inline [be16_to_host](#) (uint16_t v)
- static uint32_t force_inline [cli_readint32](#) (const void *buff)
- static uint16_t force_inline [cli_readint16](#) (const void *buff)
- static void force_inline [cli_writeint32](#) (void *offset, uint32_t v)
- static force_inline bool [hasExeInfo](#) (void)
- static force_inline bool [hasPEInfo](#) (void)
- static force_inline bool [isPE64](#) (void)
- static force_inline uint8_t [getPEMajorLinkerVersion](#) (void)
- static force_inline uint8_t [getPEMinorLinkerVersion](#) (void)
- static force_inline uint32_t [getPESizeOfCode](#) (void)
- static force_inline uint32_t [getPESizeOfInitializedData](#) (void)
- static force_inline uint32_t [getPESizeOfUninitializedData](#) (void)
- static force_inline uint32_t [getPEBaseOfCode](#) (void)
- static force_inline uint32_t [getPEBaseOfData](#) (void)
- static force_inline uint64_t [getPEImageBase](#) (void)
- static force_inline uint32_t [getPESectionAlignment](#) (void)
- static force_inline uint32_t [getPEFileAlignment](#) (void)
- static force_inline uint16_t [getPEMajorOperatingSystemVersion](#) (void)
- static force_inline uint16_t [getPEMinorOperatingSystemVersion](#) (void)
- static force_inline uint16_t [getPEMajorImageVersion](#) (void)
- static force_inline uint16_t [getPEMinorImageVersion](#) (void)
- static force_inline uint16_t [getPEMajorSubsystemVersion](#) (void)
- static force_inline uint16_t [getPEMinorSubsystemVersion](#) (void)
- static force_inline uint32_t [getPEWin32VersionValue](#) (void)
- static force_inline uint32_t [getPESizeOfImage](#) (void)
- static force_inline uint32_t [getPESizeOfHeaders](#) (void)
- static force_inline uint32_t [getPEChecksum](#) (void)
- static force_inline uint16_t [getPESubsystem](#) (void)
- static force_inline uint16_t [getPEDllCharacteristics](#) (void)
- static force_inline uint32_t [getPESizeOfStackReserve](#) (void)
- static force_inline uint32_t [getPESizeOfStackCommit](#) (void)
- static force_inline uint32_t [getPESizeOfHeapReserve](#) (void)
- static force_inline uint32_t [getPESizeOfHeapCommit](#) (void)
- static force_inline uint32_t [getPELoaderFlags](#) (void)
- static force_inline uint16_t [getPEMachine](#) ()
- static force_inline uint32_t [getPETimeDateStamp](#) ()
- static force_inline uint32_t [getPEPointerToSymbolTable](#) ()
- static force_inline uint32_t [getPENumberOfSymbols](#) ()
- static force_inline uint16_t [getPESizeOfOptionalHeader](#) ()
- static force_inline uint16_t [getPECharacteristics](#) ()
- static force_inline bool [getPEisDLL](#) ()
- static force_inline uint32_t [getPEDataDirRVA](#) (unsigned n)
- static force_inline uint32_t [getPEDataDirSize](#) (unsigned n)
- static force_inline uint16_t [getNumberOfSections](#) (void)
- static uint32_t [getPELFANew](#) (void)
- static force_inline int [readPESectionName](#) (unsigned char name[8], unsigned n)

- static force_inline uint32_t [getEntryPoint](#) (void)
- static force_inline uint32_t [getExeOffset](#) (void)
- static force_inline uint32_t [getImageBase](#) (void)
- static uint32_t [getVirtualEntryPoint](#) (void)
- static uint32_t [getSectionRVA](#) (unsigned i)
- static uint32_t [getSectionVirtualSize](#) (unsigned i)
- static force_inline bool [readRVA](#) (uint32_t rva, void *buf, size_t bufsize)
- static force_inline void * [memchr](#) (const void *s, int c, size_t n)
- void * [memset](#) (void *src, int c, uintptr_t n) __attribute__((nothrow)) __attribute__((__nonnull__(1)))
- void * [memmove](#) (void *dst, const void *src, uintptr_t n) __attribute__((nothrow)) __attribute__((__nonnull__(1)))
- void void * [memcpy](#) (void *restrict dst, const void *restrict src, uintptr_t n) __attribute__((nothrow)) __attribute__((__nonnull__(1)))
- void void int [memcmp](#) (const void *s1, const void *s2, uint32_t n) __attribute__((nothrow)) __attribute__((__pure__)) __attribute__((__nonnull__(1)))
- static force_inline uint32_t [DisassembleAt](#) (struct [DIS_fixed](#) *result, uint32_t offset, uint32_t len)
- static int32_t [ilog2_compat](#) (uint32_t a, uint32_t b)

3.4.1 Macro Definition Documentation

3.4.1.1 `#define BYTECODE_ABORT_HOOK 0xcea5e`

`entrypoint()` return code that tells hook invoker that it should skip executing, probably because it'd trigger a bug in it

3.4.1.2 `#define SIGNATURES_END`;

Old macro used to mark the end of the subsignature pattern definitions.

3.4.2 Function Documentation

3.4.2.1 static force_inline void `overloadable_func foundVirus (void)` [static]

Like [foundVirus\(\)](#) but just use the prefix as virusname

3.4.2.2 static int32_t `ilog2_compat (uint32_t a, uint32_t b)` [inline],[static]

`ilog2_compat` for 0.96 compatibility, you should use [ilog2\(\)](#) 0.96.1 API instead of this one!

Parameters

<i>a</i>	input
<i>b</i>	input

Returns

$2^{26 \cdot \log_2(a/b)}$

3.5 `bytecode_pe.h` File Reference

Data Structures

- struct [pe_image_file_hdr](#)
- struct [pe_image_data_dir](#)
- struct [pe_image_optional_hdr32](#)
- struct [pe_image_optional_hdr64](#)
- struct [pe_image_section_hdr](#)
- struct [cli_pe_hook_data](#)

Index

- `__clambc_filesize`
 - Global Variables, [27](#)
 - `__clambc_kind`
 - Global Variables, [27](#)
 - `__clambc_match_counts`
 - Global Variables, [27](#)
 - `__clambc_match_offsets`
 - Global Variables, [27](#)
 - `__clambc_pedata`
 - Global Variables, [27](#)
 - `__is_bigendian`
 - Environment, [19](#)
- ACCESS_IMM
 - `bytecode_disasm.h`, [68](#)
- ACCESS_MEM
 - `bytecode_disasm.h`, [68](#)
- ACCESS_NOARG
 - `bytecode_disasm.h`, [68](#)
- ACCESS_REG
 - `bytecode_disasm.h`, [68](#)
- ACCESS_REL
 - `bytecode_disasm.h`, [68](#)
- Abstract Data Types, [1](#)
 - `buffer_pipe_done`, [2](#)
 - `buffer_pipe_new`, [2](#)
 - `buffer_pipe_new_fromfile`, [2](#)
 - `buffer_pipe_read_avail`, [2](#)
 - `buffer_pipe_read_get`, [3](#)
 - `buffer_pipe_read_stopped`, [3](#)
 - `buffer_pipe_write_avail`, [3](#)
 - `buffer_pipe_write_get`, [3](#)
 - `buffer_pipe_write_stopped`, [3](#)
 - `hashset_add`, [4](#)
 - `hashset_contains`, [4](#)
 - `hashset_done`, [4](#)
 - `hashset_empty`, [4](#)
 - `hashset_new`, [5](#)
 - `hashset_remove`, [5](#)
 - `inflate_done`, [5](#)
 - `inflate_init`, [5](#)
 - `inflate_process`, [5](#)
 - `malloc`, [6](#)
 - `map_addkey`, [6](#)
 - `map_done`, [6](#)
 - `map_find`, [6](#)
 - `map_getvalue`, [7](#)
 - `map_getvaluesize`, [7](#)
 - `map_new`, [7](#)
 - `map_remove`, [7](#)
 - `map_setvalue`, [8](#)
- `access_size`
 - DIS_arg, [57](#)
 - DIS_mem_arg, [58](#)
- `access_type`
 - DIS_arg, [57](#)
- `add_reg`
 - DIS_mem_arg, [58](#)
- `address_size`
 - DIS_fixed, [57](#)
- `arg`
 - DIS_fixed, [57](#)
- `atoi`
 - String Operations, [51](#)
- BC_GENERIC
 - Bytecode Configuration, [11](#)
- BC_LOGICAL
 - Bytecode Configuration, [11](#)
- BC_PDF
 - Bytecode Configuration, [11](#)
- BC_PE_ALL
 - Bytecode Configuration, [11](#)
- BC_PE_UNPACKER
 - Bytecode Configuration, [11](#)
- BC_PRECLASS
 - Bytecode Configuration, [11](#)
- BC_STARTUP
 - Bytecode Configuration, [11](#)
- `bc_json_type`
 - JSON Querying, [29](#)
- `be16_to_host`
 - Environment, [19](#)
- `be32_to_host`
 - Environment, [19](#)
- `be64_to_host`
 - Environment, [20](#)
- `buffer_pipe_done`
 - Abstract Data Types, [2](#)
- `buffer_pipe_new`
 - Abstract Data Types, [2](#)
- `buffer_pipe_new_fromfile`
 - Abstract Data Types, [2](#)
- `buffer_pipe_read_avail`
 - Abstract Data Types, [2](#)
- `buffer_pipe_read_get`
 - Abstract Data Types, [3](#)
- `buffer_pipe_read_stopped`
 - Abstract Data Types, [3](#)
- `buffer_pipe_write_avail`
 - Abstract Data Types, [3](#)
- `buffer_pipe_write_get`
 - Abstract Data Types, [3](#)
- `buffer_pipe_write_stopped`
 - Abstract Data Types, [3](#)
- Bytecode Configuration, [9](#)
 - BC_GENERIC, [11](#)
 - BC_LOGICAL, [11](#)
 - BC_PDF, [11](#)
 - BC_PE_ALL, [11](#)

BC_PE_UNPACKER, 11
BC_PRECLASS, 11
BC_STARTUP, 11
BytecodeKind, 11
COPYRIGHT, 9
DECLARE_SIGNATURE, 9
FUNC_LEVEL_096, 12
FUNC_LEVEL_096_1, 12
FUNC_LEVEL_096_2, 12
FUNC_LEVEL_096_3, 12
FUNC_LEVEL_096_4, 12
FUNC_LEVEL_096_5, 12
FUNC_LEVEL_097, 12
FUNC_LEVEL_097_1, 12
FUNC_LEVEL_097_2, 12
FUNC_LEVEL_097_3, 12
FUNC_LEVEL_097_4, 12
FUNC_LEVEL_097_5, 12
FUNC_LEVEL_097_6, 12
FUNC_LEVEL_097_7, 12
FUNC_LEVEL_097_8, 12
FUNC_LEVEL_098_1, 12
FUNC_LEVEL_098_2, 12
FUNC_LEVEL_098_3, 12
FUNC_LEVEL_098_4, 12
FUNC_LEVEL_098_5, 12
FUNC_LEVEL_098_6, 12
FUNC_LEVEL_098_7, 12
FunctionalityLevels, 11
ICONGROUP1, 10
ICONGROUP2, 10
PDF_HOOK_DECLARE, 10
PE_HOOK_DECLARE, 10
PE_UNPACKER_DECLARE, 10
SIGNATURES_DECL_END, 10
SIGNATURES_DEF_END, 11
TARGET, 11
VIRUSNAME_PREFIX, 11
VIRUSNAMES, 11
bytecode_api.h
 PE_INVALID_RVA, 65
bytecode_disasm.h
 ACCESS_IMM, 68
 ACCESS_MEM, 68
 ACCESS_NOARG, 68
 ACCESS_REG, 68
 ACCESS_REL, 68
 OP_AAA, 68
 OP_AAD, 68
 OP_AAM, 68
 OP_AAS, 68
 OP_ADC, 68
 OP_ADD, 68
 OP_AND, 68
 OP_ARPL, 68
 OP_BOUND, 68
 OP_BSF, 68
 OP_BSR, 68
 OP_BSWAP, 69
 OP_BT, 69
 OPBTC, 69
 OP_BTR, 69
 OP_BTS, 69
 OP_CALL, 69
 OP_CBW, 69
 OP_CDQ, 69
 OP_CLC, 69
 OP_CLD, 69
 OP_CLI, 69
 OP_CLTS, 69
 OP_CMC, 69
 OP_CMOVA, 69
 OP_CMOVBE, 69
 OP_CMOVC, 69
 OP_CMOVG, 69
 OP_CMOVGE, 69
 OP_CMOVL, 69
 OP_CMOVLE, 69
 OP_CMOVNC, 69
 OP_CMOVNO, 69
 OP_CMOVNP, 69
 OP_CMOVNS, 69
 OP_CMOVNZ, 69
 OP_CMOVO, 69
 OP_CMOVP, 69
 OP_CMOVS, 69
 OP_CMOVZ, 69
 OP_CMP, 69
 OP_CMPSB, 69
 OP_CMPSD, 69
 OP_CMPSW, 69
 OP_CMPXCHG, 69
 OP_CMPXCHG8B, 69
 OP_CPUID, 69
 OP_CWDE, 69
 OP_DAA, 69
 OP_DAS, 69
 OP_DEC, 69
 OP_DIV, 69
 OP_ENTER, 69
 OP_F2XM1, 73
 OP_FABS, 73
 OP_FADD, 73
 OP_FADDP, 73
 OP_FBLD, 73
 OP_FBSTP, 73
 OP_FCHS, 73
 OP_FCLEX, 73
 OP_FCMOVB, 73
 OP_FCMOVBE, 73
 OP_FCMOVE, 73
 OP_FCMOVNB, 73
 OP_FCMOVNBE, 73
 OP_FCMOVNE, 73
 OP_FCMOVNU, 73
 OP_FCMOVU, 73

OP_FCOM, 73
OP_FCOMI, 73
OP_FCOMIP, 73
OP_FCOMP, 73
OP_FCOMPP, 73
OP_FCOS, 73
OP_FDECSTP, 73
OP_FDIV, 73
OP_FDIVP, 73
OP_FDIVR, 73
OP_FDIVRP, 73
OP_FFREE, 73
OP_FIADD, 73
OP_FICOM, 73
OP_FICOMP, 73
OP_FIDIV, 73
OP_FIDIVR, 73
OP_FILD, 73
OP_FIMUL, 73
OP_FINCSTP, 73
OP_FINIT, 73
OP_FIST, 73
OP_FISTP, 73
OP_FISTTP, 73
OP_FISUB, 73
OP_FISUBR, 73
OP_FLD, 73
OP_FLD1, 73
OP_FLDCW, 74
OP_FLDENV, 74
OP_FLDL2E, 74
OP_FLDL2T, 74
OP_FLDLG2, 74
OP_FLDLN2, 74
OP_FLDPI, 74
OP_FLDZ, 74
OP_FMUL, 74
OP_FMULP, 74
OP_FNOP, 74
OP_FPATAN, 74
OP_FPREM, 74
OP_FPREM1, 74
OP_FPTAN, 74
OP_FPU, 73
OP_FRNDINT, 74
OP_FRSTOR, 74
OP_FSAVE, 74
OP_FSCALE, 74
OP_FSINCOS, 74
OP_FSQRT, 74
OP_FST, 74
OP_FSTCW, 74
OP_FSTENV, 74
OP_FSTP, 74
OP_FSTSW, 74
OP_FSUB, 74
OP_FSUBP, 74
OP_FSUBR, 74
OP_FSUBRP, 74
OP_FTST, 74
OP_FUCOM, 74
OP_FUCOMI, 74
OP_FUCOMIP, 74
OP_FUCOMP, 74
OP_FUCOMPP, 74
OP_FWAIT, 69
OP_FXAM, 74
OP_FXCH, 74
OP_FXTRACT, 74
OP_FYL2X, 74
OP_FYL2XP1, 74
OP_HLT, 69
OP_IDIV, 69
OP_IMUL, 70
OP_IN, 70
OP_INC, 70
OP_INSB, 70
OP_INSD, 70
OP_INSW, 70
OP_INT, 70
OP_INT3, 70
OP_INT0, 70
OP_INV, 70
OP_INVLPG, 70
OP_IRET, 70
OP_JA, 70
OP_JBE, 70
OP_JC, 70
OP_JECXZ, 71
OP_JG, 70
OP_JGE, 70
OP_JL, 70
OP_JLE, 70
OP_JMP, 70
OP_JNC, 70
OP_JNO, 70
OP_JNP, 70
OP_JNS, 70
OP_JNZ, 70
OP_JO, 70
OP_JP, 70
OP_JS, 70
OP_JZ, 70
OP_LAHF, 70
OP_LAR, 70
OP_LDS, 70
OP_LEA, 70
OP_LEAVE, 70
OP_LES, 70
OP_LFS, 70
OP_LGDT, 70
OP_LGS, 70
OP_LIDT, 70
OP_LLD, 70
OP_LODSB, 70
OP_LODSD, 70

OP_LODSW, [70](#)
OP_LOOP, [70](#)
OP_LOOPE, [71](#)
OP_LOOPNE, [71](#)
OP_LSL, [71](#)
OP_LSS, [71](#)
OP_LTR, [71](#)
OP_MOV, [71](#)
OP_MOVSB, [71](#)
OP_MOVSD, [71](#)
OP_MOVSW, [71](#)
OP_MOVSX, [71](#)
OP_MOVZX, [71](#)
OP_MUL, [71](#)
OP_NEG, [71](#)
OP_NOP, [71](#)
OP_NOT, [71](#)
OP_OR, [71](#)
OP_OUT, [71](#)
OP_OUTSB, [71](#)
OP_OUTSD, [71](#)
OP_OUTSW, [71](#)
OP_POP, [71](#)
OP_POPAD, [71](#)
OP_POPFD, [71](#)
OP_PREFIX_LOCK, [70](#)
OP_PREFIX_REPE, [71](#)
OP_PREFIX_REPNE, [71](#)
OP_PUSH, [71](#)
OP_PUSHAD, [71](#)
OP_PUSHFD, [71](#)
OP_RCL, [71](#)
OP_RCR, [71](#)
OP_RDMSR, [71](#)
OP_RDPMC, [71](#)
OP_RDTSC, [71](#)
OP_RETF, [71](#)
OP_RETN, [71](#)
OP_ROL, [71](#)
OP_ROR, [71](#)
OP_RSM, [71](#)
OP_SAHF, [71](#)
OP_SAR, [71](#)
OP_SBB, [71](#)
OP_SCASB, [71](#)
OP_SCASD, [71](#)
OP_SCASW, [71](#)
OP_SETA, [72](#)
OP_SETBE, [72](#)
OP_SETC, [72](#)
OP_SETG, [72](#)
OP_SETGE, [72](#)
OP_SETL, [72](#)
OP_SETLE, [72](#)
OP_SETNC, [72](#)
OP_SETNO, [72](#)
OP_SETNP, [72](#)
OP_SETNS, [72](#)
OP_SETNZ, [72](#)
OP_SETO, [72](#)
OP_SETP, [72](#)
OP_SETS, [72](#)
OP_SETZ, [72](#)
OP_SGDT, [72](#)
OP_SHL, [72](#)
OP_SHLD, [72](#)
OP_SHR, [72](#)
OP_SHRD, [72](#)
OP_SIDT, [72](#)
OP_SLDT, [72](#)
OP_STC, [72](#)
OP_STD, [72](#)
OP_STI, [72](#)
OP_STOSB, [72](#)
OP_STOSD, [72](#)
OP_STOSW, [72](#)
OP_STR, [72](#)
OP_SUB, [72](#)
OP_SYSCALL, [72](#)
OP_SYSENTER, [72](#)
OP_SYSEXIT, [72](#)
OP_SYSRET, [72](#)
OP_TEST, [72](#)
OP_UD2, [72](#)
OP_VERR, [72](#)
OP_VERRW, [72](#)
OP_WBINVD, [72](#)
OP_WRMSR, [72](#)
OP_XADD, [72](#)
OP_XCHG, [72](#)
OP_XLAT, [72](#)
OP_XOR, [72](#)
SIZEB, [68](#)
SIZED, [68](#)
SIZEF, [68](#)
SIZEPTR, [68](#)
SIZEQ, [68](#)
SIZET, [68](#)
SIZEW, [68](#)
bytecode_api.h, [63](#)
 test1, [65](#)
 test2, [66](#)
bytecode_disasm.h, [66](#)
 DIS_ACCESS, [68](#)
 DIS_SIZE, [68](#)
 X86OPS, [68](#)
 X86REGS, [74](#)
bytecode_execs.h, [75](#)
bytecode_local.h, [75](#)
 foundVirus, [77](#)
 ilog2_compat, [77](#)
 SIGNATURES_END, [77](#)
bytecode_pe.h, [77](#)
bytecode_rt_error
 Scan Control, [49](#)
BytecodeKind

- Bytecode Configuration, 11
- COPYRIGHT
 - Bytecode Configuration, 9
- check_platform
 - Environment, 20
- Checksum
 - pe_image_optional_hdr32, 60
 - pe_image_optional_hdr64, 61
- chr
 - cli_exe_section, 55
- cli_exe_info, 54
 - ep, 54
 - hdr_size, 54
 - nsections, 54
 - offset, 54
 - res_addr, 54
 - section, 54
- cli_exe_section, 54
 - chr, 55
 - raw, 55
 - rsz, 55
 - rva, 55
 - uraw, 55
 - ursz, 55
 - urva, 55
 - uvsz, 55
 - vsz, 55
- cli_pe_hook_data, 55
 - dirs, 56
 - e_lfanew, 56
 - ep, 56
 - file_hdr, 56
 - hdr_size, 56
 - nsections, 56
 - opt32, 56
 - opt64, 56
 - overlays, 56
 - overlays_sz, 56
- cli_readint16
 - Environment, 20
- cli_readint32
 - Environment, 20
- cli_writeint32
 - Environment, 20
- count_match
 - Engine Queries, 17
- DECLARE_SIGNATURE
 - Bytecode Configuration, 9
- DIS_ACCESS
 - bytecode_disasm.h, 68
- DIS_SIZE
 - bytecode_disasm.h, 68
- DIS_arg, 56
 - access_size, 57
 - access_type, 57
 - mem, 57
 - other, 57
 - reg, 57
- DIS_fixed, 57
 - address_size, 57
 - arg, 57
 - operation_size, 57
 - segment, 57
 - x86_opcode, 57
- DIS_mem_arg, 58
 - access_size, 58
 - add_reg, 58
 - displacement, 58
 - scale, 58
 - scale_reg, 58
- DISASM_RESULT, 58
- debug
 - Debugging, 13
- debug_print_str
 - Debugging, 13
- debug_print_str_nonl
 - Debugging, 15
- debug_print_str_start
 - Debugging, 15
- debug_print_uint
 - Debugging, 15
- Debugging, 13
 - debug, 13
 - debug_print_str, 13
 - debug_print_str_nonl, 15
 - debug_print_str_start, 15
 - debug_print_uint, 15
- dirs
 - cli_pe_hook_data, 56
- disable_bytecode_if
 - Environment, 21
- disable_jit_if
 - Environment, 21
- disasm_x86
 - Disassembly, 16
- DisassembleAt
 - Disassembly, 16
- Disassembly, 16
 - disasm_x86, 16
 - DisassembleAt, 16
- displacement
 - DIS_mem_arg, 58
- e_lfanew
 - cli_pe_hook_data, 56
- Engine Queries, 17
 - count_match, 17
 - engine_db_options, 17
 - engine_dconf_level, 17
 - engine_functionality_level, 17
 - engine_scan_options, 17
 - match_location, 18
 - match_location_check, 18
 - matches, 18
 - running_on_jit, 18
- engine_db_options

- Engine Queries, [17](#)
- engine_dconf_level
 - Engine Queries, [17](#)
- engine_functionality_level
 - Engine Queries, [17](#)
- engine_scan_options
 - Engine Queries, [17](#)
- entropy_buffer
 - String Operations, [51](#)
- Environment, [19](#)
 - __is_bigendian, [19](#)
 - be16_to_host, [19](#)
 - be32_to_host, [19](#)
 - be64_to_host, [20](#)
 - check_platform, [20](#)
 - cli_readint16, [20](#)
 - cli_readint32, [20](#)
 - cli_writeint32, [20](#)
 - disable_bytecode_if, [21](#)
 - disable_jit_if, [21](#)
 - get_environment, [21](#)
 - le16_to_host, [21](#)
 - le32_to_host, [22](#)
 - le64_to_host, [22](#)
 - version_compare, [22](#)
- ep
 - cli_exe_info, [54](#)
 - cli_pe_hook_data, [56](#)
- extract_new
 - Scan Control, [49](#)
- extract_set_container
 - Scan Control, [49](#)
- FUNC_LEVEL_096
 - Bytecode Configuration, [12](#)
- FUNC_LEVEL_096_1
 - Bytecode Configuration, [12](#)
- FUNC_LEVEL_096_2
 - Bytecode Configuration, [12](#)
- FUNC_LEVEL_096_3
 - Bytecode Configuration, [12](#)
- FUNC_LEVEL_096_4
 - Bytecode Configuration, [12](#)
- FUNC_LEVEL_096_5
 - Bytecode Configuration, [12](#)
- FUNC_LEVEL_097
 - Bytecode Configuration, [12](#)
- FUNC_LEVEL_097_1
 - Bytecode Configuration, [12](#)
- FUNC_LEVEL_097_2
 - Bytecode Configuration, [12](#)
- FUNC_LEVEL_097_3
 - Bytecode Configuration, [12](#)
- FUNC_LEVEL_097_4
 - Bytecode Configuration, [12](#)
- FUNC_LEVEL_097_5
 - Bytecode Configuration, [12](#)
- FUNC_LEVEL_097_6
 - Bytecode Configuration, [12](#)
- FUNC_LEVEL_097_7
 - Bytecode Configuration, [12](#)
- FUNC_LEVEL_097_8
 - Bytecode Configuration, [12](#)
- FUNC_LEVEL_098_1
 - Bytecode Configuration, [12](#)
- FUNC_LEVEL_098_2
 - Bytecode Configuration, [12](#)
- FUNC_LEVEL_098_3
 - Bytecode Configuration, [12](#)
- FUNC_LEVEL_098_4
 - Bytecode Configuration, [12](#)
- FUNC_LEVEL_098_5
 - Bytecode Configuration, [12](#)
- FUNC_LEVEL_098_6
 - Bytecode Configuration, [12](#)
- FUNC_LEVEL_098_7
 - Bytecode Configuration, [12](#)
- File Operations, [23](#)
 - file_byteat, [23](#)
 - file_find, [23](#)
 - file_find_limit, [23](#)
 - fill_buffer, [25](#)
 - get_file_reliability, [25](#)
 - getFileSize, [25](#)
 - read, [25](#)
 - read_number, [26](#)
 - SEEK_CUR, [23](#)
 - SEEK_END, [23](#)
 - SEEK_SET, [23](#)
 - seek, [26](#)
 - write, [26](#)
- file_byteat
 - File Operations, [23](#)
- file_find
 - File Operations, [23](#)
- file_find_limit
 - File Operations, [23](#)
- file_hdr
 - cli_pe_hook_data, [56](#)
- FileAlignment
 - pe_image_optional_hdr32, [60](#)
 - pe_image_optional_hdr64, [61](#)
- fill_buffer
 - File Operations, [25](#)
- foundVirus
 - bytecode_local.h, [77](#)
 - Scan Control, [49](#)
- FunctionalityLevels
 - Bytecode Configuration, [11](#)
- get_environment
 - Environment, [21](#)
- get_file_reliability
 - File Operations, [25](#)
- get_pe_section
 - PE Operations, [41](#)
- getEntryPoint
 - PE Operations, [41](#)

- getExeOffset
 - PE Operations, [41](#)
- getFileSize
 - File Operations, [25](#)
- getImageBase
 - PE Operations, [41](#)
- getNumberOfSections
 - PE Operations, [41](#)
- getPEBaseOfCode
 - PE Operations, [42](#)
- getPEBaseOfData
 - PE Operations, [42](#)
- getPECharacteristics
 - PE Operations, [42](#)
- getPEChecksum
 - PE Operations, [42](#)
- getPEDataDirRVA
 - PE Operations, [42](#)
- getPEDataDirSize
 - PE Operations, [42](#)
- getPEDllCharacteristics
 - PE Operations, [43](#)
- getPEFileAlignment
 - PE Operations, [43](#)
- getPEImageBase
 - PE Operations, [43](#)
- getPELFANew
 - PE Operations, [43](#)
- getPELoaderFlags
 - PE Operations, [43](#)
- getPEMachine
 - PE Operations, [43](#)
- getPEMajorImageVersion
 - PE Operations, [44](#)
- getPEMajorLinkerVersion
 - PE Operations, [44](#)
- getPEMajorOperatingSystemVersion
 - PE Operations, [44](#)
- getPEMajorSubsystemVersion
 - PE Operations, [44](#)
- getPEMinorImageVersion
 - PE Operations, [44](#)
- getPEMinorLinkerVersion
 - PE Operations, [44](#)
- getPEMinorOperatingSystemVersion
 - PE Operations, [44](#)
- getPEMinorSubsystemVersion
 - PE Operations, [45](#)
- getPENumberOfSymbols
 - PE Operations, [45](#)
- getPEPointerToSymbolTable
 - PE Operations, [45](#)
- getPESectionAlignment
 - PE Operations, [45](#)
- getPESizeOfCode
 - PE Operations, [45](#)
- getPESizeOfHeaders
 - PE Operations, [45](#)
- getPESizeOfHeapCommit
 - PE Operations, [45](#)
- getPESizeOfHeapReserve
 - PE Operations, [46](#)
- getPESizeOfImage
 - PE Operations, [46](#)
- getPESizeOfInitializedData
 - PE Operations, [46](#)
- getPESizeOfOptionalHeader
 - PE Operations, [46](#)
- getPESizeOfStackCommit
 - PE Operations, [46](#)
- getPESizeOfStackReserve
 - PE Operations, [46](#)
- getPESizeOfUninitializedData
 - PE Operations, [46](#)
- getPESubsystem
 - PE Operations, [46](#)
- getPETimeDateStamp
 - PE Operations, [47](#)
- getPEWin32VersionValue
 - PE Operations, [47](#)
- getPEisDLL
 - PE Operations, [43](#)
- getSectionRVA
 - PE Operations, [47](#)
- getSectionVirtualSize
 - PE Operations, [47](#)
- getVirtualEntryPoint
 - PE Operations, [47](#)
- Global Variables, [27](#)
 - [__clambc_filesize](#), [27](#)
 - [__clambc_kind](#), [27](#)
 - [__clambc_match_counts](#), [27](#)
 - [__clambc_match_offsets](#), [27](#)
 - [__clambc_pedata](#), [27](#)
- hasExeInfo
 - PE Operations, [47](#)
- hasPEInfo
 - PE Operations, [47](#)
- hashset_add
 - Abstract Data Types, [4](#)
- hashset_contains
 - Abstract Data Types, [4](#)
- hashset_done
 - Abstract Data Types, [4](#)
- hashset_empty
 - Abstract Data Types, [4](#)
- hashset_new
 - Abstract Data Types, [5](#)
- hashset_remove
 - Abstract Data Types, [5](#)
- hdr_size
 - [cli_exe_info](#), [54](#)
 - [cli_pe_hook_data](#), [56](#)
- hex2ui
 - String Operations, [51](#)

- ICONGROUP1
 - Bytecode Configuration, [10](#)
- ICONGROUP2
 - Bytecode Configuration, [10](#)
- Icon Matcher, [32](#)
 - matchicon, [32](#)
- icos
 - Math Operation, [33](#)
- iexp
 - Math Operation, [33](#)
- ilog2
 - Math Operation, [33](#)
- ilog2_compat
 - bytecode_local.h, [77](#)
- ImageBase
 - pe_image_optional_hdr32, [60](#)
 - pe_image_optional_hdr64, [61](#)
- inflate_done
 - Abstract Data Types, [5](#)
- inflate_init
 - Abstract Data Types, [5](#)
- inflate_process
 - Abstract Data Types, [5](#)
- input_switch
 - Scan Control, [50](#)
- ipow
 - Math Operation, [33](#)
- isPE64
 - PE Operations, [48](#)
- isin
 - Math Operation, [34](#)
- JSON Querying, [29](#)
 - bc_json_type, [29](#)
 - json_get_array_idx, [29](#)
 - json_get_array_length, [29](#)
 - json_get_boolean, [29](#)
 - json_get_int, [30](#)
 - json_get_object, [30](#)
 - json_get_string, [30](#)
 - json_get_string_length, [30](#)
 - json_get_type, [31](#)
 - json_is_active, [31](#)
- JavaScript Normalization, [28](#)
 - jsnorm_done, [28](#)
 - jsnorm_init, [28](#)
 - jsnorm_process, [28](#)
- jsnorm_done
 - JavaScript Normalization, [28](#)
- jsnorm_init
 - JavaScript Normalization, [28](#)
- jsnorm_process
 - JavaScript Normalization, [28](#)
- json_get_array_idx
 - JSON Querying, [29](#)
- json_get_array_length
 - JSON Querying, [29](#)
- json_get_boolean
 - JSON Querying, [29](#)
- json_get_int
 - JSON Querying, [30](#)
- json_get_object
 - JSON Querying, [30](#)
- json_get_string
 - JSON Querying, [30](#)
- json_get_string_length
 - JSON Querying, [30](#)
- json_get_type
 - JSON Querying, [31](#)
- json_is_active
 - JSON Querying, [31](#)
- le16_to_host
 - Environment, [21](#)
- le32_to_host
 - Environment, [22](#)
- le64_to_host
 - Environment, [22](#)
- Machine
 - pe_image_file_hdr, [59](#)
- Magic
 - pe_image_file_hdr, [59](#)
- MajorImageVersion
 - pe_image_optional_hdr32, [60](#)
 - pe_image_optional_hdr64, [61](#)
- MajorLinkerVersion
 - pe_image_optional_hdr32, [60](#)
 - pe_image_optional_hdr64, [61](#)
- MajorOperatingSystemVersion
 - pe_image_optional_hdr32, [60](#)
 - pe_image_optional_hdr64, [61](#)
- malloc
 - Abstract Data Types, [6](#)
- map_addkey
 - Abstract Data Types, [6](#)
- map_done
 - Abstract Data Types, [6](#)
- map_find
 - Abstract Data Types, [6](#)
- map_getvalue
 - Abstract Data Types, [7](#)
- map_getvaluesize
 - Abstract Data Types, [7](#)
- map_new
 - Abstract Data Types, [7](#)
- map_remove
 - Abstract Data Types, [7](#)
- map_setvalue
 - Abstract Data Types, [8](#)
- match_location
 - Engine Queries, [18](#)
- match_location_check
 - Engine Queries, [18](#)
- matches
 - Engine Queries, [18](#)
- matchicon
 - Icon Matcher, [32](#)

- Math Operation, [33](#)
 - icos, [33](#)
 - iexp, [33](#)
 - ilog2, [33](#)
 - ipow, [33](#)
 - isin, [34](#)
- mem
 - DIS_arg, [57](#)
- memchr
 - String Operations, [52](#)
- memcmp
 - String Operations, [52](#)
- memcpy
 - String Operations, [52](#)
- memmove
 - String Operations, [52](#)
- memset
 - String Operations, [53](#)
- memstr
 - String Operations, [53](#)
- MinorImageVersion
 - pe_image_optional_hdr32, [60](#)
 - pe_image_optional_hdr64, [61](#)
- MinorLinkerVersion
 - pe_image_optional_hdr32, [60](#)
 - pe_image_optional_hdr64, [61](#)
- MinorOperatingSystemVersion
 - pe_image_optional_hdr32, [60](#)
 - pe_image_optional_hdr64, [61](#)
- Name
 - pe_image_section_hdr, [62](#)
- nsections
 - cli_exe_info, [54](#)
 - cli_pe_hook_data, [56](#)
- NumberOfLinenumbers
 - pe_image_section_hdr, [62](#)
- NumberOfRelocations
 - pe_image_section_hdr, [62](#)
- NumberOfRvaAndSizes
 - pe_image_optional_hdr32, [60](#)
 - pe_image_optional_hdr64, [62](#)
- NumberOfSections
 - pe_image_file_hdr, [59](#)
- NumberOfSymbols
 - pe_image_file_hdr, [59](#)
- OP_AAA
 - bytecode_disasm.h, [68](#)
- OP_AAD
 - bytecode_disasm.h, [68](#)
- OP_AAM
 - bytecode_disasm.h, [68](#)
- OP_AAS
 - bytecode_disasm.h, [68](#)
- OP_ADC
 - bytecode_disasm.h, [68](#)
- OP_ADD
 - bytecode_disasm.h, [68](#)
- OP_AND
 - bytecode_disasm.h, [68](#)
- OP_ARPL
 - bytecode_disasm.h, [68](#)
- OP_BOUND
 - bytecode_disasm.h, [68](#)
- OP_BSF
 - bytecode_disasm.h, [68](#)
- OP_BSR
 - bytecode_disasm.h, [68](#)
- OP_BSWAP
 - bytecode_disasm.h, [69](#)
- OP_BT
 - bytecode_disasm.h, [69](#)
- OPBTC
 - bytecode_disasm.h, [69](#)
- OPBTR
 - bytecode_disasm.h, [69](#)
- OPBTS
 - bytecode_disasm.h, [69](#)
- OP_CALL
 - bytecode_disasm.h, [69](#)
- OPCBW
 - bytecode_disasm.h, [69](#)
- OPCDQ
 - bytecode_disasm.h, [69](#)
- OPCLC
 - bytecode_disasm.h, [69](#)
- OPCLD
 - bytecode_disasm.h, [69](#)
- OPCLI
 - bytecode_disasm.h, [69](#)
- OPCLTS
 - bytecode_disasm.h, [69](#)
- OPCMC
 - bytecode_disasm.h, [69](#)
- OPCMOVA
 - bytecode_disasm.h, [69](#)
- OPCMOVBE
 - bytecode_disasm.h, [69](#)
- OPCMOVC
 - bytecode_disasm.h, [69](#)
- OPCMOVG
 - bytecode_disasm.h, [69](#)
- OPCMOVGE
 - bytecode_disasm.h, [69](#)
- OPCMOVL
 - bytecode_disasm.h, [69](#)
- OPCMOVLE
 - bytecode_disasm.h, [69](#)
- OPCMOVNC
 - bytecode_disasm.h, [69](#)
- OPCMOVNO
 - bytecode_disasm.h, [69](#)
- OPCMOVNP
 - bytecode_disasm.h, [69](#)
- OPCMOVNS
 - bytecode_disasm.h, [69](#)

OP_CMOVNZ
 bytecode_disasm.h, 69

OP_CMOVO
 bytecode_disasm.h, 69

OP_CMOVP
 bytecode_disasm.h, 69

OP_CMOVS
 bytecode_disasm.h, 69

OP_CMOVZ
 bytecode_disasm.h, 69

OP_CMP
 bytecode_disasm.h, 69

OP_CMPSB
 bytecode_disasm.h, 69

OP_CMPSD
 bytecode_disasm.h, 69

OP_CMPSW
 bytecode_disasm.h, 69

OP_CMPXCHG
 bytecode_disasm.h, 69

OP_CMPXCHG8B
 bytecode_disasm.h, 69

OP_CPUID
 bytecode_disasm.h, 69

OP_CWDE
 bytecode_disasm.h, 69

OP_DAA
 bytecode_disasm.h, 69

OP_DAS
 bytecode_disasm.h, 69

OP_DEC
 bytecode_disasm.h, 69

OP_DIV
 bytecode_disasm.h, 69

OP_ENTER
 bytecode_disasm.h, 69

OP_F2XM1
 bytecode_disasm.h, 73

OP_FABS
 bytecode_disasm.h, 73

OP_FADD
 bytecode_disasm.h, 73

OP_FADDP
 bytecode_disasm.h, 73

OP_FBLD
 bytecode_disasm.h, 73

OP_FBSTP
 bytecode_disasm.h, 73

OP_FCHS
 bytecode_disasm.h, 73

OP_FCLEX
 bytecode_disasm.h, 73

OP_FCMOVB
 bytecode_disasm.h, 73

OP_FCMOVBE
 bytecode_disasm.h, 73

OP_FCMOVE
 bytecode_disasm.h, 73

OP_FCMOVNB
 bytecode_disasm.h, 73

OP_FCMOVNBE
 bytecode_disasm.h, 73

OP_FCMOVNE
 bytecode_disasm.h, 73

OP_FCMOVNU
 bytecode_disasm.h, 73

OP_FCMOVU
 bytecode_disasm.h, 73

OP_FCOM
 bytecode_disasm.h, 73

OP_FCOMI
 bytecode_disasm.h, 73

OP_FCOMIP
 bytecode_disasm.h, 73

OP_FCOMP
 bytecode_disasm.h, 73

OP_FCOMPP
 bytecode_disasm.h, 73

OP_FCOS
 bytecode_disasm.h, 73

OP_FDECSTP
 bytecode_disasm.h, 73

OP_FDIV
 bytecode_disasm.h, 73

OP_FDIVP
 bytecode_disasm.h, 73

OP_FDIVR
 bytecode_disasm.h, 73

OP_FDIVRP
 bytecode_disasm.h, 73

OP_FFREE
 bytecode_disasm.h, 73

OP_FIADD
 bytecode_disasm.h, 73

OP_FICOM
 bytecode_disasm.h, 73

OP_FICOMP
 bytecode_disasm.h, 73

OP_FIDIV
 bytecode_disasm.h, 73

OP_FIDIVR
 bytecode_disasm.h, 73

OP_FILD
 bytecode_disasm.h, 73

OP_FIMUL
 bytecode_disasm.h, 73

OP_FINCSTP
 bytecode_disasm.h, 73

OP_FINISH
 bytecode_disasm.h, 73

OP_FIST
 bytecode_disasm.h, 73

OP_FISTP
 bytecode_disasm.h, 73

OP_FISTTP
 bytecode_disasm.h, 73

OP_FISUB
 bytecode_disasm.h, 73
OP_FISUBR
 bytecode_disasm.h, 73
OP_FLD
 bytecode_disasm.h, 73
OP_FLD1
 bytecode_disasm.h, 73
OP_FLDCW
 bytecode_disasm.h, 74
OP_FLDENV
 bytecode_disasm.h, 74
OP_FLDL2E
 bytecode_disasm.h, 74
OP_FLDL2T
 bytecode_disasm.h, 74
OP_FLDLG2
 bytecode_disasm.h, 74
OP_FLDLN2
 bytecode_disasm.h, 74
OP_FLDPI
 bytecode_disasm.h, 74
OP_FLDZ
 bytecode_disasm.h, 74
OP_FMUL
 bytecode_disasm.h, 74
OP_FMULP
 bytecode_disasm.h, 74
OP_FNOP
 bytecode_disasm.h, 74
OP_FPATAN
 bytecode_disasm.h, 74
OP_FPREM
 bytecode_disasm.h, 74
OP_FPREM1
 bytecode_disasm.h, 74
OP_FPTAN
 bytecode_disasm.h, 74
OP_FPU
 bytecode_disasm.h, 73
OP_FRNDINT
 bytecode_disasm.h, 74
OP_FRSTOR
 bytecode_disasm.h, 74
OP_FSAVE
 bytecode_disasm.h, 74
OP_FSCALE
 bytecode_disasm.h, 74
OP_FSINCOS
 bytecode_disasm.h, 74
OP_FSQRT
 bytecode_disasm.h, 74
OP_FST
 bytecode_disasm.h, 74
OP_FSTCW
 bytecode_disasm.h, 74
OP_FSTENV
 bytecode_disasm.h, 74

OP_FSTP
 bytecode_disasm.h, 74
OP_FSTSW
 bytecode_disasm.h, 74
OP_FSUB
 bytecode_disasm.h, 74
OP_FSUBP
 bytecode_disasm.h, 74
OP_FSUBR
 bytecode_disasm.h, 74
OP_FSUBRP
 bytecode_disasm.h, 74
OP_FTST
 bytecode_disasm.h, 74
OP_FUCOM
 bytecode_disasm.h, 74
OP_FUCOMI
 bytecode_disasm.h, 74
OP_FUCOMIP
 bytecode_disasm.h, 74
OP_FUCOMP
 bytecode_disasm.h, 74
OP_FUCOMPP
 bytecode_disasm.h, 74
OP_FWAIT
 bytecode_disasm.h, 69
OP_FXAM
 bytecode_disasm.h, 74
OP_FXCH
 bytecode_disasm.h, 74
OP_FXTRACT
 bytecode_disasm.h, 74
OP_FYL2X
 bytecode_disasm.h, 74
OP_FYL2XP1
 bytecode_disasm.h, 74
OP_HLT
 bytecode_disasm.h, 69
OP_IDIV
 bytecode_disasm.h, 69
OP_IMUL
 bytecode_disasm.h, 70
OP_IN
 bytecode_disasm.h, 70
OP_INC
 bytecode_disasm.h, 70
OP_INSB
 bytecode_disasm.h, 70
OP_INSD
 bytecode_disasm.h, 70
OP_INSW
 bytecode_disasm.h, 70
OP_INT
 bytecode_disasm.h, 70
OP_INT3
 bytecode_disasm.h, 70
OP_INT0
 bytecode_disasm.h, 70

OP_INVD
 bytecode_disasm.h, 70

OP_INVLPG
 bytecode_disasm.h, 70

OP_IRET
 bytecode_disasm.h, 70

OP_JA
 bytecode_disasm.h, 70

OP_JBE
 bytecode_disasm.h, 70

OP_JC
 bytecode_disasm.h, 70

OP_JECXZ
 bytecode_disasm.h, 71

OP_JG
 bytecode_disasm.h, 70

OP_JGE
 bytecode_disasm.h, 70

OP_JL
 bytecode_disasm.h, 70

OP_JLE
 bytecode_disasm.h, 70

OP_JMP
 bytecode_disasm.h, 70

OP_JNC
 bytecode_disasm.h, 70

OP_JNO
 bytecode_disasm.h, 70

OP_JNP
 bytecode_disasm.h, 70

OP_JNS
 bytecode_disasm.h, 70

OP_JNZ
 bytecode_disasm.h, 70

OP_JO
 bytecode_disasm.h, 70

OP_JP
 bytecode_disasm.h, 70

OP_JS
 bytecode_disasm.h, 70

OP_JZ
 bytecode_disasm.h, 70

OP_LAHF
 bytecode_disasm.h, 70

OP_LAR
 bytecode_disasm.h, 70

OP_LDS
 bytecode_disasm.h, 70

OP_LEA
 bytecode_disasm.h, 70

OP_LEAVE
 bytecode_disasm.h, 70

OP_LES
 bytecode_disasm.h, 70

OP_LFS
 bytecode_disasm.h, 70

OP_LGDT
 bytecode_disasm.h, 70

OP_LGS
 bytecode_disasm.h, 70

OP_LIDT
 bytecode_disasm.h, 70

OP_LLDT
 bytecode_disasm.h, 70

OP_LODSB
 bytecode_disasm.h, 70

OP_LODSD
 bytecode_disasm.h, 70

OP_LODSW
 bytecode_disasm.h, 70

OP_LOOP
 bytecode_disasm.h, 70

OP_LOOPE
 bytecode_disasm.h, 71

OP_LOOPNE
 bytecode_disasm.h, 71

OP_LSL
 bytecode_disasm.h, 71

OP_LSS
 bytecode_disasm.h, 71

OP_LTR
 bytecode_disasm.h, 71

OP_MOV
 bytecode_disasm.h, 71

OP_MOVSB
 bytecode_disasm.h, 71

OP_MOVSD
 bytecode_disasm.h, 71

OP_MOVSW
 bytecode_disasm.h, 71

OP_MOVSX
 bytecode_disasm.h, 71

OP_MOVZX
 bytecode_disasm.h, 71

OP_MUL
 bytecode_disasm.h, 71

OP_NEG
 bytecode_disasm.h, 71

OP_NOP
 bytecode_disasm.h, 71

OP_NOT
 bytecode_disasm.h, 71

OP_OR
 bytecode_disasm.h, 71

OP_OUT
 bytecode_disasm.h, 71

OP_OUTSB
 bytecode_disasm.h, 71

OP_OUTSD
 bytecode_disasm.h, 71

OP_OUTSW
 bytecode_disasm.h, 71

OP_POP
 bytecode_disasm.h, 71

OP_POPAD
 bytecode_disasm.h, 71

OP_POPFD
 bytecode_disasm.h, 71

OP_PREFIX_LOCK
 bytecode_disasm.h, 70

OP_PREFIX_REPE
 bytecode_disasm.h, 71

OP_PREFIX_REPNE
 bytecode_disasm.h, 71

OP_PUSH
 bytecode_disasm.h, 71

OP_PUSHD
 bytecode_disasm.h, 71

OP_PUSHFD
 bytecode_disasm.h, 71

OP_RCL
 bytecode_disasm.h, 71

OP_RCR
 bytecode_disasm.h, 71

OP_RDMSR
 bytecode_disasm.h, 71

OP_RDPMC
 bytecode_disasm.h, 71

OP_RDTSC
 bytecode_disasm.h, 71

OP_RETF
 bytecode_disasm.h, 71

OP_RETN
 bytecode_disasm.h, 71

OP_ROL
 bytecode_disasm.h, 71

OP_ROR
 bytecode_disasm.h, 71

OP_RSM
 bytecode_disasm.h, 71

OP_SAHF
 bytecode_disasm.h, 71

OP_SAR
 bytecode_disasm.h, 71

OP_SBB
 bytecode_disasm.h, 71

OP_SCASB
 bytecode_disasm.h, 71

OP_SCASD
 bytecode_disasm.h, 71

OP_SCASW
 bytecode_disasm.h, 71

OP_SETA
 bytecode_disasm.h, 72

OP_SETBE
 bytecode_disasm.h, 72

OP_SETC
 bytecode_disasm.h, 72

OP_SETG
 bytecode_disasm.h, 72

OP_SETGE
 bytecode_disasm.h, 72

OP_SETL
 bytecode_disasm.h, 72

OP_SETLE
 bytecode_disasm.h, 72

OP_SETNC
 bytecode_disasm.h, 72

OP_SETNO
 bytecode_disasm.h, 72

OP_SETNP
 bytecode_disasm.h, 72

OP_SETNS
 bytecode_disasm.h, 72

OP_SETNZ
 bytecode_disasm.h, 72

OP_SETO
 bytecode_disasm.h, 72

OP_SETP
 bytecode_disasm.h, 72

OP_SETS
 bytecode_disasm.h, 72

OP_SETZ
 bytecode_disasm.h, 72

OP_SGDT
 bytecode_disasm.h, 72

OP_SHL
 bytecode_disasm.h, 72

OP_SHLD
 bytecode_disasm.h, 72

OP_SHR
 bytecode_disasm.h, 72

OP_SHRD
 bytecode_disasm.h, 72

OP_SIDT
 bytecode_disasm.h, 72

OP_SLDT
 bytecode_disasm.h, 72

OP_STC
 bytecode_disasm.h, 72

OP_STD
 bytecode_disasm.h, 72

OP_STI
 bytecode_disasm.h, 72

OP_STOSB
 bytecode_disasm.h, 72

OP_STOSD
 bytecode_disasm.h, 72

OP_STOSW
 bytecode_disasm.h, 72

OP_STR
 bytecode_disasm.h, 72

OP_SUB
 bytecode_disasm.h, 72

OP_SYSCALL
 bytecode_disasm.h, 72

OP_SYSENTER
 bytecode_disasm.h, 72

OP_SYSEXIT
 bytecode_disasm.h, 72

OP_SYSRET
 bytecode_disasm.h, 72

- OP_TEST
 - bytecode_disasm.h, 72
- OP_UD2
 - bytecode_disasm.h, 72
- OP_VERR
 - bytecode_disasm.h, 72
- OP_VERRW
 - bytecode_disasm.h, 72
- OP_WBINVD
 - bytecode_disasm.h, 72
- OP_WRMSR
 - bytecode_disasm.h, 72
- OP_XADD
 - bytecode_disasm.h, 72
- OP_XCHG
 - bytecode_disasm.h, 72
- OP_XLAT
 - bytecode_disasm.h, 72
- OP_XOR
 - bytecode_disasm.h, 72
- offset
 - cli_exe_info, 54
- operation_size
 - DIS_fixed, 57
- opt32
 - cli_pe_hook_data, 56
- opt64
 - cli_pe_hook_data, 56
- other
 - DIS_arg, 57
- overlays
 - cli_pe_hook_data, 56
- overlays_sz
 - cli_pe_hook_data, 56
- PDF Handling
 - PDF_PHASE_END, 35
 - PDF_PHASE_NONE, 35
 - PDF_PHASE_PARSED, 35
 - PDF_PHASE_POSTDUMP, 35
 - PDF_PHASE_PRE, 35
- PDF_PHASE_END
 - PDF Handling, 35
- PDF_PHASE_NONE
 - PDF Handling, 35
- PDF_PHASE_PARSED
 - PDF Handling, 35
- PDF_PHASE_POSTDUMP
 - PDF Handling, 35
- PDF_PHASE_PRE
 - PDF Handling, 35
- PE_INVALID_RVA
 - bytecode_api.h, 65
- PDF Handling, 35
 - pdf_flag, 35
 - pdf_get_dumpedobjid, 35
 - pdf_get_flags, 36
 - pdf_get_obj_num, 36
 - pdf_get_offset, 36
 - pdf_get_phase, 36
 - pdf_getobj, 36
 - pdf_getobjflags, 36
 - pdf_getobjid, 37
 - pdf_getobjsize, 37
 - pdf_lookupobj, 37
 - pdf_objflags, 35
 - pdf_phase, 35
 - pdf_set_flags, 37
 - pdf_setobjflags, 37
- PDF_HOOK_DECLARE
 - Bytecode Configuration, 10
- PE Operations, 40
 - get_pe_section, 41
 - getEntryPoint, 41
 - getExeOffset, 41
 - getImageBase, 41
 - getNumberOfSections, 41
 - getPEBaseOfCode, 42
 - getPEBaseOfData, 42
 - getPECharacteristics, 42
 - getPEChecksum, 42
 - getPEDataDirRVA, 42
 - getPEDataDirSize, 42
 - getPEDllCharacteristics, 43
 - getPEFileAlignment, 43
 - getPEImageBase, 43
 - getPELFANew, 43
 - getPELoaderFlags, 43
 - getPEMachine, 43
 - getPEMajorImageVersion, 44
 - getPEMajorLinkerVersion, 44
 - getPEMajorOperatingSystemVersion, 44
 - getPEMajorSubsystemVersion, 44
 - getPEMinorImageVersion, 44
 - getPEMinorLinkerVersion, 44
 - getPEMinorOperatingSystemVersion, 44
 - getPEMinorSubsystemVersion, 45
 - getPENumberOfSymbols, 45
 - getPEPointerToSymbolTable, 45
 - getPESectionAlignment, 45
 - getPESizeOfCode, 45
 - getPESizeOfHeaders, 45
 - getPESizeOfHeapCommit, 45
 - getPESizeOfHeapReserve, 46
 - getPESizeOfImage, 46
 - getPESizeOfInitializedData, 46
 - getPESizeOfOptionalHeader, 46
 - getPESizeOfStackCommit, 46
 - getPESizeOfStackReserve, 46
 - getPESizeOfUninitializedData, 46
 - getPESubsystem, 46
 - getPETimeDateStamp, 47
 - getPEWin32VersionValue, 47
 - getPEisDLL, 43
 - getSectionRVA, 47
 - getSectionVirtualSize, 47
 - getVirtualEntryPoint, 47

- hasExeInfo, 47
- hasPEInfo, 47
- isPE64, 48
- pe_rawaddr, 48
- readPESectionName, 48
- readRVA, 48
- PE_HOOK_DECLARE
 - Bytecode Configuration, 10
- PE_UNPACKER_DECLARE
 - Bytecode Configuration, 10
- pdf_flag
 - PDF Handling, 35
- pdf_get_dumpedobjid
 - PDF Handling, 35
- pdf_get_flags
 - PDF Handling, 36
- pdf_get_obj_num
 - PDF Handling, 36
- pdf_get_offset
 - PDF Handling, 36
- pdf_get_phase
 - PDF Handling, 36
- pdf_getobj
 - PDF Handling, 36
- pdf_getobjflags
 - PDF Handling, 36
- pdf_getobjid
 - PDF Handling, 37
- pdf_getobjsize
 - PDF Handling, 37
- pdf_lookupobj
 - PDF Handling, 37
- pdf_objflags
 - PDF Handling, 35
- pdf_phase
 - PDF Handling, 35
- pdf_set_flags
 - PDF Handling, 37
- pdf_setobjflags
 - PDF Handling, 37
- pe_image_data_dir, 58
- pe_image_file_hdr, 58
 - Machine, 59
 - Magic, 59
 - NumberOfSections, 59
 - NumberOfSymbols, 59
 - PointerToSymbolTable, 59
 - SizeOfOptionalHeader, 59
 - TimeStamp, 59
- pe_image_optional_hdr32, 59
 - Checksum, 60
 - FileAlignment, 60
 - ImageBase, 60
 - MajorImageVersion, 60
 - MajorLinkerVersion, 60
 - MajorOperatingSystemVersion, 60
 - MinorImageVersion, 60
 - MinorLinkerVersion, 60
 - MinorOperatingSystemVersion, 60
 - NumberOfRvaAndSizes, 60
 - SectionAlignment, 60
 - SizeOfCode, 60
 - SizeOfInitializedData, 60
 - SizeOfUninitializedData, 60
- pe_image_optional_hdr64, 61
 - Checksum, 61
 - FileAlignment, 61
 - ImageBase, 61
 - MajorImageVersion, 61
 - MajorLinkerVersion, 61
 - MajorOperatingSystemVersion, 61
 - MinorImageVersion, 61
 - MinorLinkerVersion, 61
 - MinorOperatingSystemVersion, 61
 - NumberOfRvaAndSizes, 62
 - SectionAlignment, 62
 - SizeOfCode, 62
 - SizeOfInitializedData, 62
 - SizeOfUninitializedData, 62
- pe_image_section_hdr, 62
 - Name, 62
 - NumberOfLinenumbers, 62
 - NumberOfRelocations, 62
 - PointerToLinenumbers, 62
 - PointerToRawData, 62
 - PointerToRelocations, 63
 - SizeOfRawData, 63
- pe_rawaddr
 - PE Operations, 48
- PointerToLinenumbers
 - pe_image_section_hdr, 62
- PointerToRawData
 - pe_image_section_hdr, 62
- PointerToRelocations
 - pe_image_section_hdr, 63
- PointerToSymbolTable
 - pe_image_file_hdr, 59
- raw
 - cli_exe_section, 55
- read
 - File Operations, 25
- read_number
 - File Operations, 26
- readPESectionName
 - PE Operations, 48
- readRVA
 - PE Operations, 48
- reg
 - DIS_arg, 57
- res_addr
 - cli_exe_info, 54
- rsz
 - cli_exe_section, 55
- running_on_jit
 - Engine Queries, 18
- rva

- cli_exe_section, 55
- SEEK_CUR
 - File Operations, 23
- SEEK_END
 - File Operations, 23
- SEEK_SET
 - File Operations, 23
- SIEB
 - bytecode_disasm.h, 68
- SIZED
 - bytecode_disasm.h, 68
- SIZEF
 - bytecode_disasm.h, 68
- SIZEPTR
 - bytecode_disasm.h, 68
- SIZEQ
 - bytecode_disasm.h, 68
- SIZET
 - bytecode_disasm.h, 68
- SIZEW
 - bytecode_disasm.h, 68
- SIGNATURES_DECL_END
 - Bytecode Configuration, 10
- SIGNATURES_DEF_END
 - Bytecode Configuration, 11
- SIGNATURES_END
 - bytecode_local.h, 77
- scale
 - DIS_mem_arg, 58
- scale_reg
 - DIS_mem_arg, 58
- Scan Control, 49
 - bytecode_rt_error, 49
 - extract_new, 49
 - extract_set_container, 49
 - foundVirus, 49
 - input_switch, 50
 - setvirusname, 50
- section
 - cli_exe_info, 54
- SectionAlignment
 - pe_image_optional_hdr32, 60
 - pe_image_optional_hdr64, 62
- seek
 - File Operations, 26
- segment
 - DIS_fixed, 57
- setvirusname
 - Scan Control, 50
- SizeOfCode
 - pe_image_optional_hdr32, 60
 - pe_image_optional_hdr64, 62
- SizeOfInitializedData
 - pe_image_optional_hdr32, 60
 - pe_image_optional_hdr64, 62
- SizeOfOptionalHeader
 - pe_image_file_hdr, 59
- SizeOfRawData
 - pe_image_section_hdr, 63
- SizeOfUninitializedData
 - pe_image_optional_hdr32, 60
 - pe_image_optional_hdr64, 62
- String Operations, 51
 - atoi, 51
 - entropy_buffer, 51
 - hex2ui, 51
 - memchr, 52
 - memcmp, 52
 - memcpy, 52
 - memmove, 52
 - memset, 53
 - memstr, 53
- TARGET
 - Bytecode Configuration, 11
- test1
 - bytecode_api.h, 65
- test2
 - bytecode_api.h, 66
- TimeStamp
 - pe_image_file_hdr, 59
- uraw
 - cli_exe_section, 55
- ursz
 - cli_exe_section, 55
- urva
 - cli_exe_section, 55
- uvsz
 - cli_exe_section, 55
- VIRUSNAME_PREFIX
 - Bytecode Configuration, 11
- VIRUSNAMES
 - Bytecode Configuration, 11
- version_compare
 - Environment, 22
- vsz
 - cli_exe_section, 55
- write
 - File Operations, 26
- x86_opcode
 - DIS_fixed, 57
- X86OPS
 - bytecode_disasm.h, 68
- X86REGS
 - bytecode_disasm.h, 74