LANbeacon

Generated by Doxygen 1.8.13

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 interfaces Struct Reference

Contains all variables, that are needed to access sockets on interfaces.

```
#include <receiver.h>
```

Data Fields

- int sockfd [20]
- int sockopt [20]
- int maxSockFd
- int numInterfaces
- struct ifreq if_idx [20]
- struct ifreq if_mac [20]
- unsigned short etherType
- unsigned short sendOrReceive

3.1.1 Detailed Description

Contains all variables, that are needed to access sockets on interfaces.

3.1.2 Field Documentation

3.1.2.1 etherType

```
unsigned short interfaces::etherType
```

EtherType to send or receive on interface.

3.1.2.2 if_idx

```
struct ifreq interfaces::if_idx[20]
```

Interface IDs.

3.1.2.3 if_mac

```
struct ifreq interfaces::if_mac[20]
```

Interface MACs.

3.1.2.4 maxSockFd

```
int interfaces::maxSockFd
```

Needed for select function.

3.1.2.5 numInterfaces

```
int interfaces::numInterfaces
```

Number of used interfaces.

3.1.2.6 sendOrReceive

```
unsigned short interfaces::sendOrReceive
```

Switch for send or receive mode.

3.1.2.7 sockfd

```
int interfaces::sockfd[20]
```

File descriptors of raw sockets.

3.1.2.8 sockopt

```
int interfaces::sockopt[20]
```

. Options for each raw socket.

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

- receiver.h

3.2 open_ssl_keys Struct Reference

Key locations, password and further configurations.

```
#include <openssl_sign.h>
```

Data Fields

- char path_To_Verifying_Key [KEY_PATHLENGTH_MAX+1]
- char path_To_Signing_Key [KEY_PATHLENGTH_MAX+1]
- char pcszPassphrase [1024]
- int generate_keys
- int sender_or_receiver_mode

3.2.1 Detailed Description

Key locations, password and further configurations.

3.2.2 Field Documentation

3.2.2.1 generate_keys

```
int open_ssl_keys::generate_keys
```

Flag that determines, if keys should be generated.

3.2.2.2 path_To_Signing_Key

```
char open_ssl_keys::path_To_Signing_Key[KEY_PATHLENGTH_MAX+1]
```

Specified path of private key location.

3.2.2.3 path_To_Verifying_Key

```
char open_ssl_keys::path_To_Verifying_Key[KEY_PATHLENGTH_MAX+1]
```

Specified path of public key location.

3.2.2.4 pcszPassphrase

```
char open_ssl_keys::pcszPassphrase[1024]
```

Specified password for private key.

3.2.2.5 sender_or_receiver_mode

```
int open_ssl_keys::sender_or_receiver_mode
```

Flag for corresponding client mode.

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

- openssl_sign.h

3.3 received_lan_beacon_frame Struct Reference

Contains all the information related to one received frame.

```
#include <receiver.h>
```

Data Fields

- unsigned char lan_beacon_ReceivedPayload [LAN_BEACON_BUF_SIZ]
- ssize_t payloadSize
- unsigned long challenge
- unsigned char current_destination_mac [6]
- int successfullyAuthenticated
- int times_left_to_display
- char ** parsedBeaconContents

3.3.1 Detailed Description

Contains all the information related to one received frame.

3.3.2 Field Documentation

3.3.2.1 challenge

```
unsigned long received_lan_beacon_frame::challenge
```

The challange, that has been sent to the server.

3.3.2.2 current_destination_mac

```
unsigned char received_lan_beacon_frame::current_destination_mac[6]
```

The MAC address of the server, which the frame was received from.

3.3.2.3 lan_beacon_ReceivedPayload

```
unsigned char received_lan_beacon_frame::lan_beacon_ReceivedPayload[LAN_BEACON_BUF_SIZ]
```

Contains the raw received payload from a LAN-Beacon frame.

3.3.2.4 parsedBeaconContents

```
char** received_lan_beacon_frame::parsedBeaconContents
```

Contains the parsed contents, that will be used to print something to the display.

3.3.2.5 payloadSize

```
ssize_t received_lan_beacon_frame::payloadSize
```

The size of the raw payload.

3.3.2.6 successfullyAuthenticated

```
int received_lan_beacon_frame::successfullyAuthenticated
```

Has frame already been authenticated?

3.3.2.7 times_left_to_display

```
int received_lan_beacon_frame::times_left_to_display
```

Countdown, how many more times the frame will be displayed. Is updated, if frame with same content is received again.

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

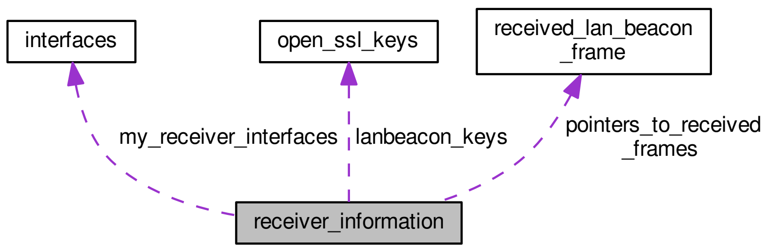
- receiver.h

3.4 receiver_information Struct Reference

Receiver configurations.

```
#include <receiver.h>
```

Collaboration diagram for receiver_information:



Data Fields

- int authenticated_mode
- int scroll_speed
- int current_lan_beacon_pdu_for_printing
- struct received_lan_beacon_frame ∗ pointers_to_received_frames [20]
- int number_of_currently_received_frames
- struct open_ssl_keys lanbeacon_keys
- struct interfaces my_receiver_interfaces

3.4.1 Detailed Description

Receiver configurations.

3.4.2 Field Documentation

3.4.2.1 authenticated_mode

```
int receiver_information::authenticated_mode
```

Has user specified using the authenticated mode?

3.4.2.2 current_lan_beacon_pdu_for_printing

```
int receiver_information::current_lan_beacon_pdu_for_printing
```

The currently printed PDU.

3.4.2.3 lanbeacon_keys

```
struct open_ssl_keys receiver_information::lanbeacon_keys
```

The paths to the keys.

3.4.2.4 my_receiver_interfaces

```
struct interfaces receiver_information::my_receiver_interfaces
```

Interfaces, that are used for LAN-Beacon reception.

3.4.2.5 number_of_currently_received_frames

```
int receiver_information::number_of_currently_received_frames
```

How many frames are currently stored for displaying.

3.4.2.6 pointers_to_received_frames

```
struct received_lan_beacon_frame* receiver_information::pointers_to_received_frames[20]
```

Frames, that currently are stored for displaying.

3.4.2.7 scroll_speed

```
int receiver_information::scroll_speed
```

How fast the display should switch to the next display page.

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

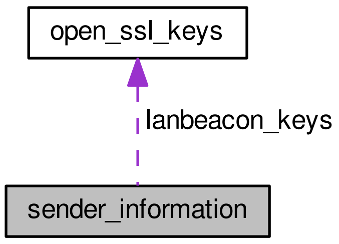
- receiver.h

3.5 sender_information Struct Reference

Sender configurations.

```
#include <sender.h>
```

Collaboration diagram for sender_information:



Data Fields

- char ∗ lanBeacon_PDU
- int lan_beacon_pdu_len
- int send_frequency
- char ∗ interface_to_send_on
- struct open_ssl_keys lanbeacon_keys

3.5.1 Detailed Description

Sender configurations.

3.5.2 Field Documentation

3.5.2.1 interface_to_send_on

```
char* sender_information::interface_to_send_on
```

If specified, interface that is used for sending.

3.5.2.2 lan_beacon_pdu_len

```
int sender_information::lan_beacon_pdu_len
```

Length of the combined PDU.

3.5.2.3 lanbeacon_keys

```
struct open_ssl_keys sender_information::lanbeacon_keys
```

Keys configuration.

3.5.2.4 lanBeacon_PDU

```
char* sender_information::lanBeacon_PDU
```

The combinded payload of a PDU, that is being sent.

3.5.2.5 send_frequency

```
int sender_information::send_frequency
```

Number of seconds between each sent PDU.

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

- sender.h

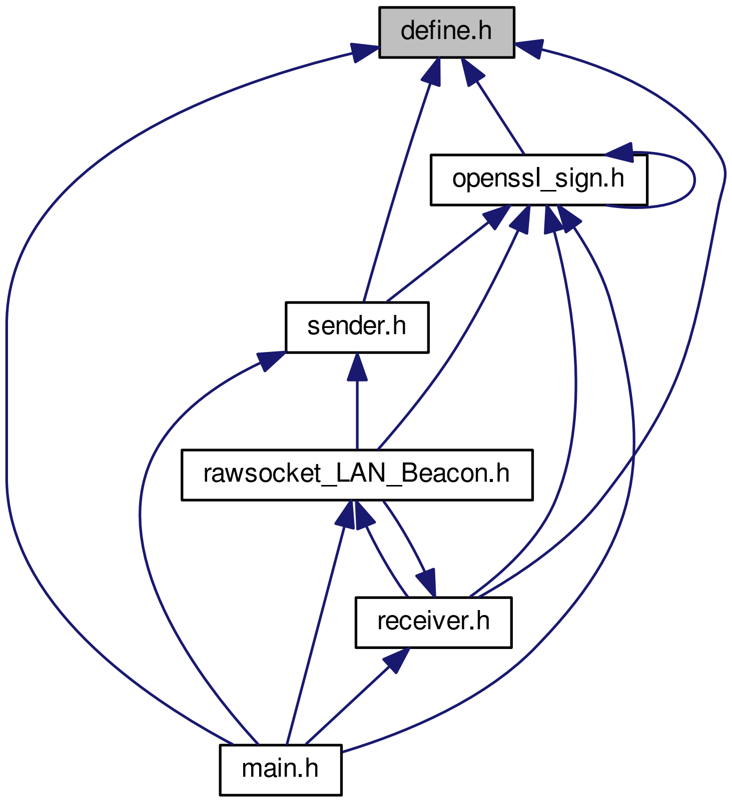
Chapter 4

File Documentation

4.1 define.h File Reference

Contains application-wide includes with information such as addresses and TLV types.

This graph shows which files directly or indirectly include this file:



Macros

- #define _(STRING) gettext(STRING) /∗∗ @name Macro for gettext localization support ∗/
- #define **LAN_BEACON_SEND_FREQUENCY** 5 /∗∗ @name Send frequency in seconds ∗/

LAN-Beacon Multicast addresses and EtherTypes

- #define **LAN_BEACON_DEST_MAC** 0xff, 0xff, 0xff, 0xff, 0xff, 0xff
- #define **LAN_BEACON_ETHER_TYPE** 0x88B5
- #define **CHALLENGE_ETHTYPE** 0x88B6

Buffer sizes

- #define **PARSED_TLVS_MAX_NUMBER** 25
- #define **PARSED_TLVS_MAX_LENGTH** 510
- #define **LAN_BEACON_BUF_SIZ** 2000
- #define **KEY_PATHLENGTH_MAX** 800

Standard paths

- #define **PRIVATE_KEY_STANDARD_PATH** "private_key.pem"
- #define **PUBLIC_KEY_STANDARD_PATH** "public_key.pem"

Display options

- #define **DEFAULT_SCROLLSPEED** 2
- #define **SHOW_FRAMES_X_TIMES** 3
- #define **DESCRIPTOR_WIDTH** 10

Subtype numbers lanbeacon

- #define **SUBTYPE_VLAN_ID** 200
- #define **SUBTYPE_NAME** 201
- #define **SUBTYPE_CUSTOM** 202
- #define **SUBTYPE_IPV4** 203
- #define **SUBTYPE_IPV6** 204
- #define **SUBTYPE_EMAIL** 205
- #define **SUBTYPE_DHCP** 206
- #define **SUBTYPE_ROUTER** 207
- #define **SUBTYPE_SIGNATURE** 216
- #define **SUBTYPE_COMBINED_STRING** 217

Descriptor strings lanbeacon

- #define **DESCRIPTOR_VLAN_ID** gettext("VLAN-ID:")
- #define **DESCRIPTOR_NAME** gettext("VLAN-Name:")
- #define **DESCRIPTOR_CUSTOM** gettext("Freetext:")
- #define **DESCRIPTOR_IPV4** gettext("IPv4:")
- #define **DESCRIPTOR_IPV6** gettext("IPv6:")
- #define **DESCRIPTOR_EMAIL** gettext("Email:")
- #define **DESCRIPTOR_DHCP** gettext("DHCP:")
- #define **DESCRIPTOR_ROUTER** gettext("Router:")
- #define **DESCRIPTOR_SIGNATURE** gettext("Authentication:")
- #define **DESCRIPTOR_COMBINED_STRING** gettext("Combined String:")

4.1.1 Detailed Description

Contains application-wide includes with information such as addresses and TLV types.

Author

Dominik Bitzer

Date

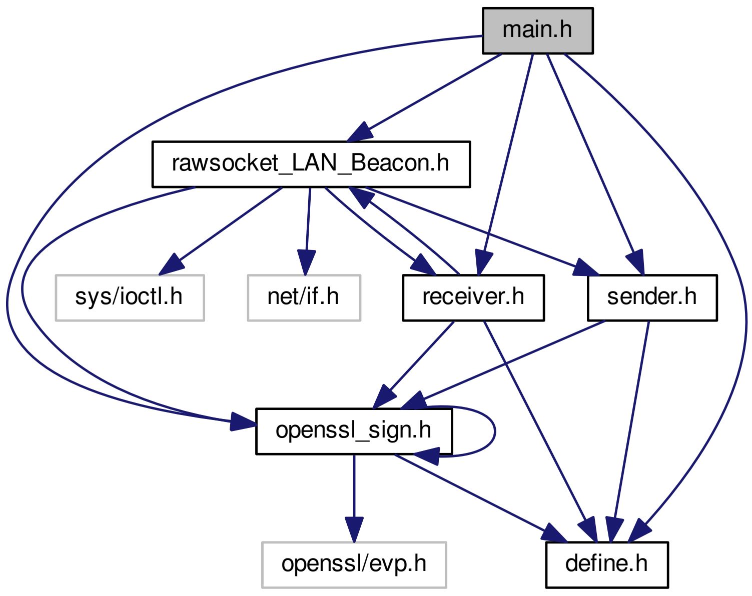
2017

4.2 main.h File Reference

Main function and help function.

```
#include "openssl_sign.h"
#include "sender.h"
#include "rawsocket_LAN_Beacon.h"
#include "receiver.h"
#include "define.h"
```

Include dependency graph for main.h:



Functions

- int main (int argc, char **argv)

  Separates receiver from sender mode and has the main program logic.
- void printHelp ()

  Help function, executed if unknown parameters have been received or user specifically asks for help.

4.2.1 Detailed Description

Main function and help function.

Author

Dominik Bitzer

Date

2017

4.2.2 Function Documentation

4.2.2.1 main()

```
int main (
            int argc,
            char ** argv )
```

Separates receiver from sender mode and has the main program logic.

Returns

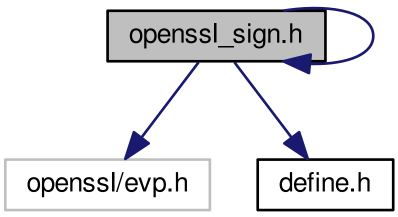
Success or failure code.

4.3 openssl_sign.h File Reference

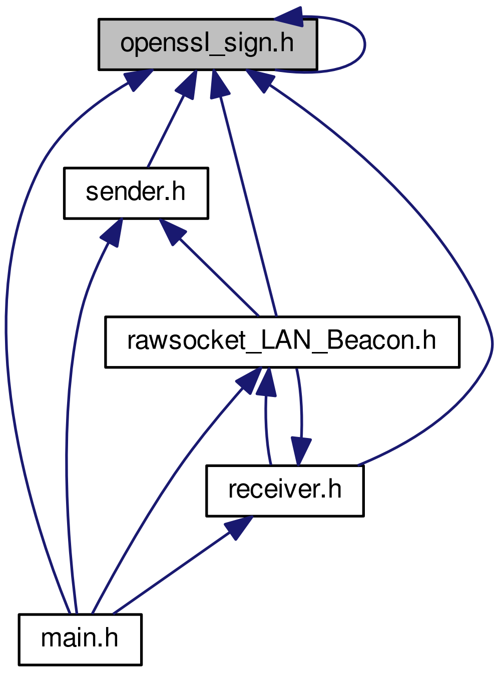
signing, verifying and key I/O

```
#include <openssl/evp.h>
#include "openssl_sign.h"
#include "define.h"
```

Include dependency graph for openssl_sign.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct open_ssl_keys

  Key locations, password and further configurations.

Macros

- #define **SENDER_MODE** 0
- #define **RECEIVER_MODE** 1

Functions

- int make_keys (EVP_PKEY ∗∗skey, EVP_PKEY ∗∗vkey, struct open_ssl_keys ∗lanbeacon_keys)

  Generate and save keys to specified paths.
- void print_it (const char ∗label, const unsigned char ∗buff, size_t len)

  Prints a buffer to stdout. Label is optional.
- int **passwd_callback** (char ∗pcszBuff, int size, int rwflag, void ∗pPass)
- int signlanbeacon (unsigned char ∗∗sig, size_t ∗slen, const unsigned char ∗msg, size_t qqlen, struct open↩_ssl_keys ∗lanbeacon_keys)

  Create signature for LAN-Beacon PDU.
- int read_keys (EVP_PKEY ∗∗skey, EVP_PKEY ∗∗vkey, struct open_ssl_keys ∗lanbeacon_keys)

  Read stored pem files into memory.
- int verifylanbeacon (const unsigned char ∗msg, size_t mlen, struct open_ssl_keys ∗lanbeacon_keys)

  Verify the signature for LAN-Beacon PDUs.

4.3.1 Detailed Description

signing, verifying and key I/O

Author

Dominik Bitzer

Date

2017

4.3.2 Function Documentation

4.3.2.1 make_keys()

```
int make_keys (
            EVP_PKEY ** skey,
            EVP_PKEY ** vkey,
            struct open_ssl_keys * lanbeacon_keys )
```

Generate and save keys to specified paths.

Parameters

skey	pointer, where private key should be stored
vkey	pointer, where public key should be stored
lanbeacon_keys	configuration for file paths and password

Returns

Returns 0 for success, non-0 otherwise

4.3.2.2 print_it()

```
void print_it (
            const char * label,
            const unsigned char * buff,
            size_t len )
```

Prints a buffer to stdout. Label is optional.

Parameters

label	Descriptor that will be put with contents
buff	Buffer for printing
len	Length of the buffer

4.3.2.3 read_keys()

```
int read_keys (
            EVP_PKEY ** skey,
            EVP_PKEY ** vkey,
            struct open_ssl_keys * lanbeacon_keys )
```

Read stored pem files into memory.

Parameters

skey	Memory address for the private key
vkey	Memory address for the public key
lanbeacon_keys	Configurations of the keys

Returns

Success or error codes

4.3.2.4 signlanbeacon()

```
int signlanbeacon (
            unsigned char ** sig,
            size_t * slen,
            const unsigned char * msg,
            size_t qqlen,
            struct open_ssl_keys * lanbeacon_keys )
```

Create signature for LAN-Beacon PDU.

Parameters

sig	Memory pointer for signature
slen	Length of the created signature
msg	LAN-Beacon PDU that should be signed
qqlen	Size of the passed LAN-Beacon PDU
lanbeacon_keys	Configurations of the keys

Returns

Success or error codes

4.3.2.5 verifylanbeacon()

```
int verifylanbeacon (
            const unsigned char * msg,
            size_t mlen,
            struct open_ssl_keys * lanbeacon_keys )
```

Verify the signature for LAN-Beacon PDUs.

Parameters

msg	Message, that should be verified
mlen	Length of the message, that should be verified
lanbeacon_keys	Configurations of the keys

Returns

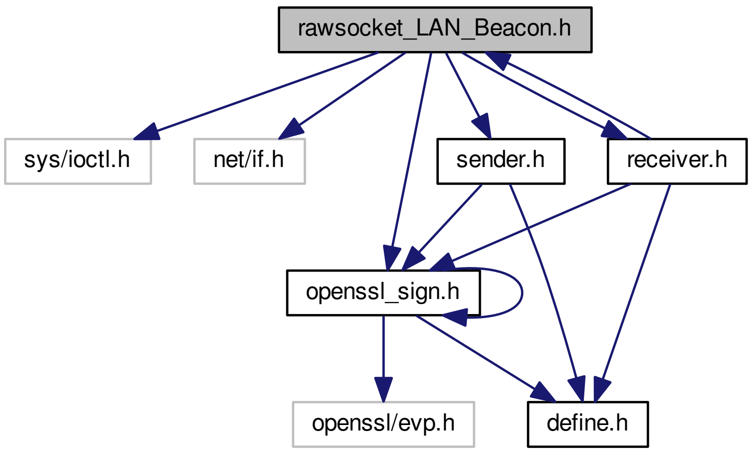
Success or error codes

4.4 rawsocket_LAN_Beacon.h File Reference

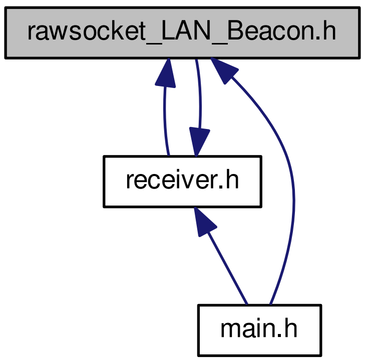
raw-socket sending and receiving

```
#include <sys/ioctl.h>
#include <net/if.h>
#include "openssl_sign.h"
#include "receiver.h"
#include "sender.h"
```

Include dependency graph for rawsocket_LAN_Beacon.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **SEND_SOCKET** 0
- #define **REC_SOCKET** 1

Functions

- void new_lan_beacon_receiver (struct receiver_information ∗my_receiver_information)

    Receives LAN-Beacons and adds them to the structure of received beacons.

- int send_lan_beacon_rawSock (struct sender_information ∗my_sender_information)

  Shortcut that can be used for sending LAN-Beacons, provides some configuration already.
- unsigned long receiveChallenge (struct interfaces ∗my_challenge_interfaces, char ∗challenge_dest_mac, struct sender_information ∗my_sender_information)

  Listen for any and eventually receive challenges, that clients have sent in response to LAN-Beacon frames.
- void getInterfaces (struct interfaces ∗my_interfaces_struct, char ∗interface_to_send_on)

  Get raw sockets for interfaces.
- void sendRawSocket (unsigned char ∗destination_mac, void ∗payload, int payloadLen, unsigned short etherType, struct open_ssl_keys ∗lanbeacon_keys, char ∗interface_to_send_on, struct sender_information ∗my_sender_information)

  Generic function to send on raw sockets, both handles sending of LAN-Beacon and challenges.

4.4.1 Detailed Description

raw-socket sending and receiving

Author

Dominik Bitzer

Date

2017

4.4.2 Function Documentation

4.4.2.1 getInterfaces()

```
void getInterfaces (
            struct interfaces ∗ my_interfaces_struct,
            char ∗ interface_to_send_on )
```

Get raw sockets for interfaces.

Parameters

my_interfaces_struct	Struct that contains interfaces information and configuration
interface_to_send_on	Specified interfaces for sending

4.4.2.2 new_lan_beacon_receiver()

```
void new_lan_beacon_receiver (
            struct receiver_information ∗ my_receiver_information )
```

Receives LAN-Beacons and adds them to the structure of received beacons.

Parameters

my_receiver_information	Receiver configuration and structs for storing the received beacons

4.4.2.3 receiveChallenge()

```
unsigned long receiveChallenge (
            struct interfaces * my_challenge_interfaces,
            char * challenge_dest_mac,
            struct sender_information * my_sender_information )
```

Listen for any and eventually receive challenges, that clients have sent in response to LAN-Beacon frames.

Parameters

my_challenge_interfaces	Struct with the sockets for receiving challenges
challenge_dest_mac	States the destination to send the authenticated LAN-Beacon
my_sender_information	Sender configurations

Returns

Returnes the value of the received challenge

4.4.2.4 send_lan_beacon_rawSock()

```
int send_lan_beacon_rawSock (
            struct sender_information * my_sender_information )
```

Shortcut that can be used for sending LAN-Beacons, provides some configuration already.

Parameters

my_sender_information	Struct that contains everything needed for sending

Returns

Success or failure code, which is passed on from called function

4.4.2.5 sendRawSocket()

```
void sendRawSocket (
            unsigned char * destination_mac,
            void * payload,
            int payloadLen,
            unsigned short etherType,
            struct open_ssl_keys * lanbeacon_keys,
            char * interface_to_send_on,
            struct sender_information * my_sender_information )
```

Generic function to send on raw sockets, both handles sending of LAN-Beacon and challenges.

Parameters

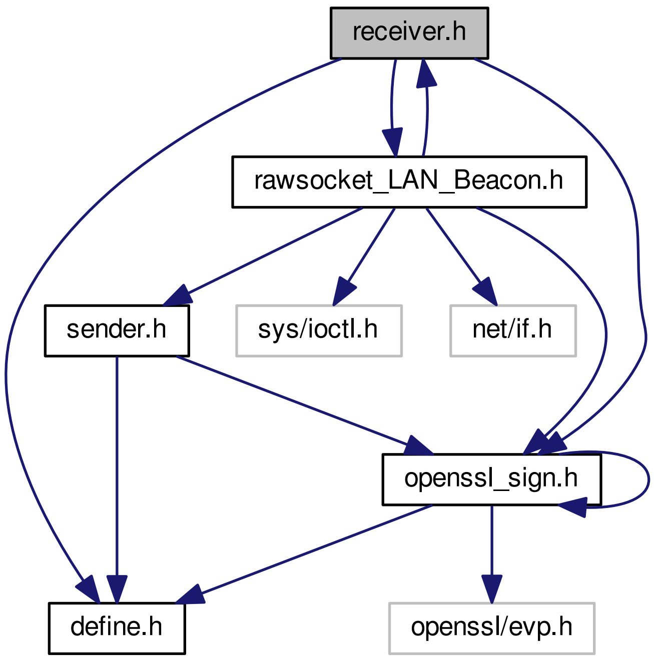
destination_mac	Destination MAC address
payload	Payload that should be sent
payloadLen	Length of payload
etherType	EtherType of payload
lanbeacon_keys	Keys that are used for sending
interface_to_send_on	Interface, that information should be sent on
my_sender_information	Sender configurations

4.5 receiver.h File Reference

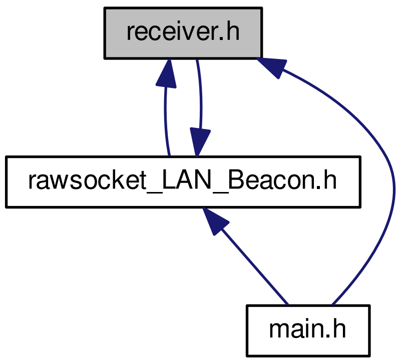
Receiver-specific functions and structures.

```
#include "define.h"
#include "openssl_sign.h"
#include "rawsocket_LAN_Beacon.h"
```

Include dependency graph for receiver.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct received_lan_beacon_frame

Contains all the information related to one received frame.
- struct interfaces

  Contains all variables, that are needed to access sockets on interfaces.
- struct receiver_information

  Receiver configurations.

Functions

- char ∗∗ evaluatelanbeacon (struct received_lan_beacon_frame ∗my_received_lan_beacon_frame, struct open_ssl_keys ∗lanbeacon_keys)

  This function takes raw received LAN-Beacon frames and creates strings from them, that can be used for printing or further processing.
- void bananaPIprint (struct receiver_information ∗my_receiver_information)

  This function prints the received content on the standard output and, if compiler flags are set, also on a C-Berry display.

4.5.1 Detailed Description

Receiver-specific functions and structures.

Author

Dominik Bitzer

Date

2017

4.5.2 Function Documentation

4.5.2.1 bananaPIprint()

```
void bananaPIprint (
            struct receiver_information * my_receiver_information )
```

This function prints the received content on the standard output and, if compiler flags are set, also on a C-Berry display.

Parameters

my_receiver_information	receiver information struct, that contains display settings and contents that should be printed

4.5.2.2 evaluatelanbeacon()

```
char** evaluatelanbeacon (
            struct received_lan_beacon_frame * my_received_lan_beacon_frame,
            struct open_ssl_keys * lanbeacon_keys )
```

This function takes raw received LAN-Beacon frames and creates strings from them, that can be used for printing or further processing.

Parameters

my_received_lan_beacon_frame	Pointer to one single received LAN-Beacon frame, that should be evaluated
lanbeacon_keys	Pointer to struct for keys, needed in order to verify authentication information

Returns

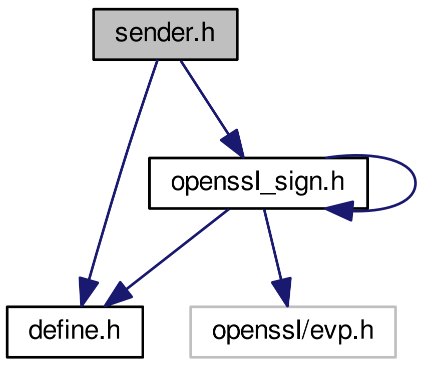
Returns parsed content as an array of TLV-descriptor and TLV-content pairs

4.6 sender.h File Reference

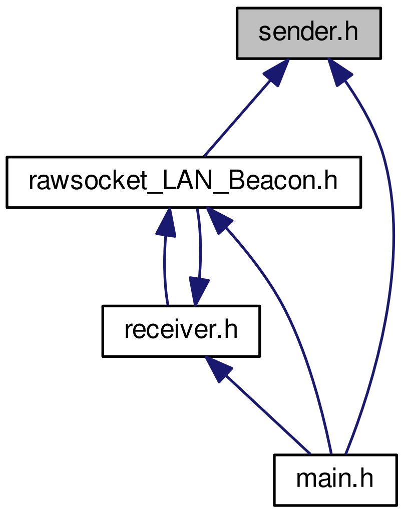
Sender-specific functions and structures.

```
#include "define.h"
#include "openssl_sign.h"
```

Include dependency graph for sender.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct sender_information

  Sender configurations.

Functions

- char ∗ mergedlanbeaconCreator (int ∗argc, char ∗∗argv, struct sender_information ∗my_sender_information)

  Creates a LAN-Beacon PDU from the command line arguments.
- void transferToCombinedBeaconAndString (unsigned char subtype, char ∗TLVdescription, char ∗∗combined↩ String, char ∗source, char ∗combinedBeacon, int ∗currentByte)

  Shortcut function for cases in which only a string is transferred, no binary format TLVs.
- void transferToCombinedBeacon (unsigned char subtype, void ∗source, char ∗combinedBeacon, int ∗currentByte, unsigned short int currentTLVlength)

  Transferring the content of the field to the combined lanbeacon in binary format.
- void transferToCombinedString (char ∗TLVdescription, char ∗∗combinedString, char ∗source)

  Transfer human-readable information to combined string.
- void ipParser (int ip_V4or6, char ∗optarg, char ∗∗combinedString, char ∗combinedBeacon, int ∗currentByte)

  Parse IPv4 or IPv6 subnets to binary format.

4.6.1 Detailed Description

Sender-specific functions and structures.

Author

Dominik Bitzer

Date

2017

4.6.2 Function Documentation

4.6.2.1 ipParser()

```
void ipParser (
            int ip_V4or6,
            char * optarg,
            char ** combinedString,
            char * combinedBeacon,
            int * currentByte )
```

Parse IPv4 or IPv6 subnets to binary format.

Using regex to get IP-addresses from string input, then convert them to binary representation for transport

Parameters

ip_V4or6	Switch between IPv4 and IPv6 mode
optarg	String, which should be parsed
combinedString	Pointer to the string, that contains text representation of all contents
combinedBeacon	PDU of beacon, that TLVs should be added to
currentByte	current position in the Beacon-PDU

4.6.2.2 mergedlanbeaconCreator()

```
char* mergedlanbeaconCreator (
            int * argc,
            char ** argv,
            struct sender_information * my_sender_information )
```

Creates a LAN-Beacon PDU from the command line arguments.

Howto for adding new fields:

1. Add defines for desired new field in define.h
2. Add desired options in mergedlanbeaconCreator()

Parameters

argc	Number of arguments.
argv	Contents of arguments.

Returns

Returns an array, that contains the payload of a lanBeacon_PDU

4.6.2.3 transferToCombinedBeacon()

```
void transferToCombinedBeacon (
            unsigned char subtype,
            void * source,
            char * combinedBeacon,
            int * currentByte,
            unsigned short int currentTLVlength )
```

Transferring the content of the field to the combined lanbeacon in binary format.

Parameters

subtype	Subtype of the TLV
source	String contents, that should be included to the PDU
combinedBeacon	PDU of beacon, that TLVs should be added to
currentByte	current position in the Beacon-PDU
currentTLVlength	Length of the passed TLV

4.6.2.4 transferToCombinedBeaconAndString()

```
void transferToCombinedBeaconAndString (
            unsigned char subtype,
            char * TLVdescription,
            char ** combinedString,
            char * source,
            char * combinedBeacon,
            int * currentByte )
```

Shortcut function for cases in which only a string is transferred, no binary format TLVs.

Parameters

subtype	Subtype of the TLV
TLVdescription	Descriptor string of the TLV
combinedString	Pointer to the string, that contains text representation of all contents
source	String contents, that should be included to the PDU
combinedBeacon	PDU of beacon, that TLVs should be added to
currentByte	current position in the Beacon-PDU

4.6.2.5 transferToCombinedString()

```
void transferToCombinedString (
             char * TLVdescription,
             char ** combinedString,
             char * source )
```

Transfer human-readable information to combined string.

Transferring the content of the field to the combined string in human-readable format. If one combined string exceeds 507 byte limit of TLV it is put to the next combined string TLV

Parameters

TLVdescription	Descriptor string of the TLV
combinedString	Pointer to the string, that contains text representation of all contents
source	String contents, that should be included to the PDU

Index