

LANbeacon

Generated by Doxygen 1.8.13



# Contents

<b>1</b>	<b>Data Structure Index</b>	<b>1</b>
1.1	Data Structures . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Data Structure Documentation</b>	<b>5</b>
3.1	interfaces Struct Reference . . . . .	5
3.1.1	Detailed Description . . . . .	5
3.1.2	Field Documentation . . . . .	5
3.1.2.1	etherType . . . . .	5
3.1.2.2	if_idx . . . . .	6
3.1.2.3	if_mac . . . . .	6
3.1.2.4	maxSockFd . . . . .	6
3.1.2.5	numInterfaces . . . . .	6
3.1.2.6	sendOrReceive . . . . .	6
3.1.2.7	sockfd . . . . .	6
3.1.2.8	sockopt . . . . .	6
3.2	open_ssl_keys Struct Reference . . . . .	7
3.2.1	Detailed Description . . . . .	7
3.2.2	Field Documentation . . . . .	7
3.2.2.1	generate_keys . . . . .	7
3.2.2.2	path_To_Signing_Key . . . . .	7
3.2.2.3	path_To_Verifying_Key . . . . .	7

3.2.2.4	<a href="#">pcszPassphrase</a>	7
3.2.2.5	<a href="#">sender_or_receiver_mode</a>	8
3.3	<a href="#">received_lan_beacon_frame Struct Reference</a>	8
3.3.1	<a href="#">Detailed Description</a>	8
3.3.2	<a href="#">Field Documentation</a>	8
3.3.2.1	<a href="#">challenge</a>	8
3.3.2.2	<a href="#">current_destination_mac</a>	8
3.3.2.3	<a href="#">lan_beacon_ReceivedPayload</a>	9
3.3.2.4	<a href="#">parsedBeaconContents</a>	9
3.3.2.5	<a href="#">payloadSize</a>	9
3.3.2.6	<a href="#">successfullyAuthenticated</a>	9
3.3.2.7	<a href="#">times_left_to_display</a>	9
3.4	<a href="#">receiver_information Struct Reference</a>	9
3.4.1	<a href="#">Detailed Description</a>	10
3.4.2	<a href="#">Field Documentation</a>	10
3.4.2.1	<a href="#">authenticated_mode</a>	10
3.4.2.2	<a href="#">current_lan_beacon_pdu_for_printing</a>	10
3.4.2.3	<a href="#">lanbeacon_keys</a>	10
3.4.2.4	<a href="#">my_receiver_interfaces</a>	10
3.4.2.5	<a href="#">number_of_currently_received_frames</a>	10
3.4.2.6	<a href="#">pointers_to_received_frames</a>	11
3.4.2.7	<a href="#">scroll_speed</a>	11
3.5	<a href="#">sender_information Struct Reference</a>	11
3.5.1	<a href="#">Detailed Description</a>	11
3.5.2	<a href="#">Field Documentation</a>	12
3.5.2.1	<a href="#">interface_to_send_on</a>	12
3.5.2.2	<a href="#">lan_beacon_pdu_len</a>	12
3.5.2.3	<a href="#">lanbeacon_keys</a>	12
3.5.2.4	<a href="#">lanBeacon_PDU</a>	12
3.5.2.5	<a href="#">send_frequency</a>	12

<b>4 File Documentation</b>	<b>13</b>
4.1 define.h File Reference	13
4.1.1 Detailed Description	15
4.2 main.h File Reference	15
4.2.1 Detailed Description	16
4.2.2 Function Documentation	16
4.2.2.1 main()	16
4.3 openssl_sign.h File Reference	16
4.3.1 Detailed Description	18
4.3.2 Function Documentation	18
4.3.2.1 make_keys()	18
4.3.2.2 print_it()	18
4.3.2.3 read_keys()	19
4.3.2.4 signlanbeacon()	19
4.3.2.5 verifylanbeacon()	20
4.4 rawsocket_LAN_Beacon.h File Reference	20
4.4.1 Detailed Description	22
4.4.2 Function Documentation	22
4.4.2.1 getInterfaces()	22
4.4.2.2 new_lan_beacon_receiver()	22
4.4.2.3 receiveChallenge()	24
4.4.2.4 send_lan_beacon_rawSock()	24
4.4.2.5 sendRawSocket()	24
4.5 receiver.h File Reference	25
4.5.1 Detailed Description	27
4.5.2 Function Documentation	27
4.5.2.1 bananaPlprint()	27
4.5.2.2 evaluatelanbeacon()	28
4.6 sender.h File Reference	28
4.6.1 Detailed Description	29
4.6.2 Function Documentation	30
4.6.2.1 ipParser()	30
4.6.2.2 mergedlanbeaconCreator()	30
4.6.2.3 transferToCombinedBeacon()	31
4.6.2.4 transferToCombinedBeaconAndString()	31
4.6.2.5 transferToCombinedString()	32
<b>Index</b>	<b>33</b>



# Chapter 1

## Data Structure Index

### 1.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">interfaces</a>	Contains all variables, that are needed to access sockets on interfaces . . . . .	5
<a href="#">open_ssl_keys</a>	Key locations, password and further configurations . . . . .	7
<a href="#">received_lan_beacon_frame</a>	Contains all the information related to one received frame . . . . .	8
<a href="#">receiver_information</a>	Receiver configurations . . . . .	9
<a href="#">sender_information</a>	Sender configurations . . . . .	11





## Chapter 2

# File Index

### 2.1 File List

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

<a href="#">define.h</a>	Contains application-wide includes with information such as addresses and TLV types . . . . .	13
<a href="#">main.h</a>	Main function and help function . . . . .	15
<a href="#">openssl_sign.h</a>	Signing, verifying and key I/O . . . . .	16
<a href="#">rawsocket_LAN_Beacon.h</a>	Raw-socket sending and receiving . . . . .	20
<a href="#">receiver.h</a>	Receiver-specific functions and structures . . . . .	25
<a href="#">sender.h</a>	Sender-specific functions and structures . . . . .	28



## 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 challenge, 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:

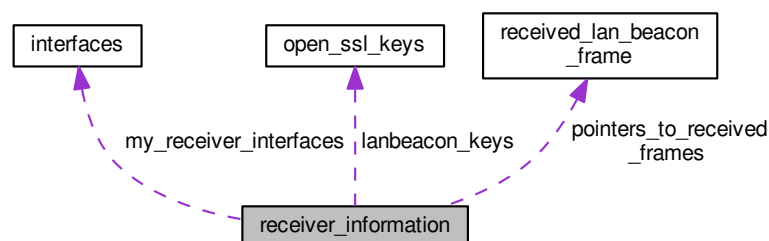
- [receiver.h](#)

## 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:

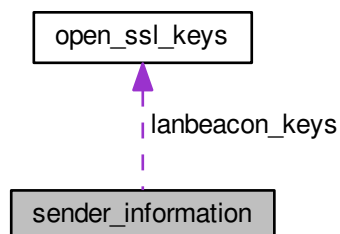
- [receiver.h](#)

## 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 combined 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](#)

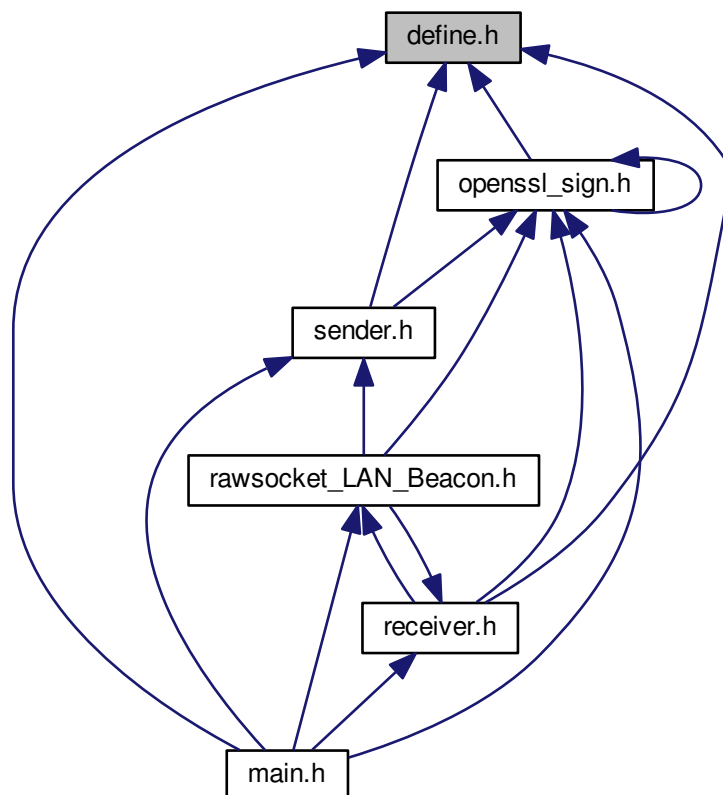
## Chapter 4

# 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**

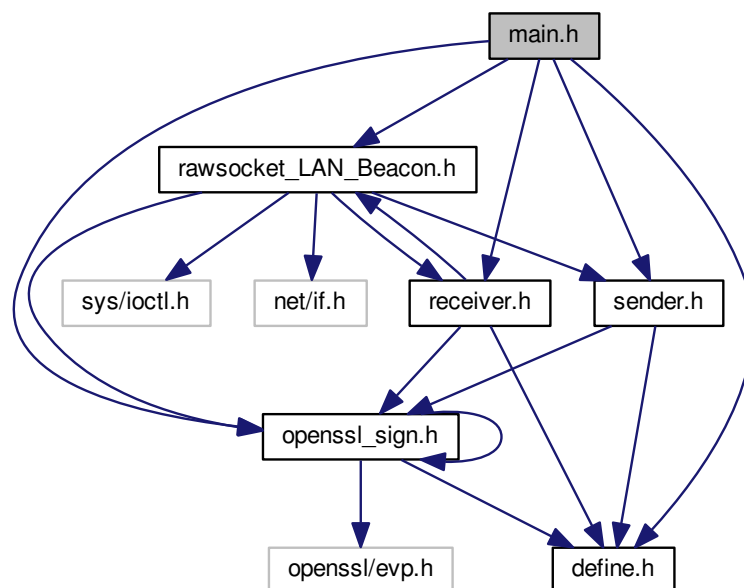
2017

## 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

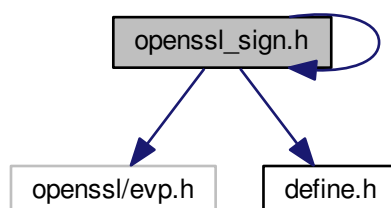
Success or failure code.

## 4.3 openssl\_sign.h File Reference

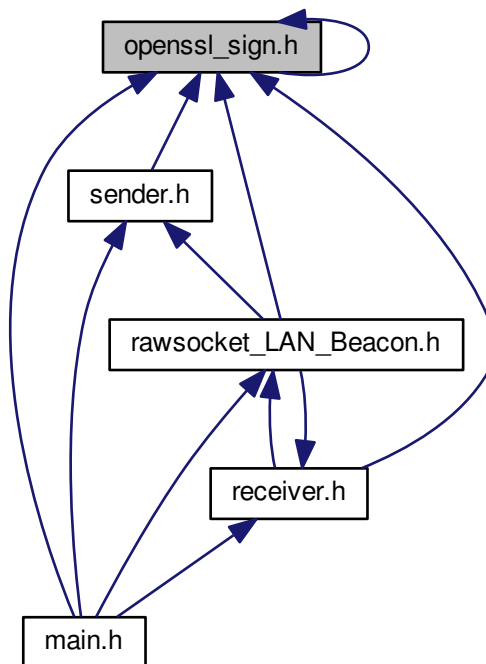
signing, verifying and key I/O

```
#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 qlen, 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

<i>skey</i>	pointer, where private key should be stored
<i>vkey</i>	pointer, where public key should be stored
<i>lanbeacon_keys</i>	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

<i>label</i>	Descriptor that will be put with contents
<i>buff</i>	Buffer for printing
<i>len</i>	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

<i>skey</i>	Memory address for the private key
<i>vkey</i>	Memory address for the public key
<i>lanbeacon_keys</i>	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 qlen,
    struct open_ssl_keys * lanbeacon_keys )
```

Create signature for LAN-Beacon PDU.

## Parameters

<i>sig</i>	Memory pointer for signature
<i>slen</i>	Length of the created signature
<i>msg</i>	LAN-Beacon PDU that should be signed
<i>qlen</i>	Size of the passed LAN-Beacon PDU
<i>lanbeacon_keys</i>	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**

<i>msg</i>	Message, that should be verified
<i>mlen</i>	Length of the message, that should be verified
<i>lanbeacon_keys</i>	Configurations of the keys

**Returns**

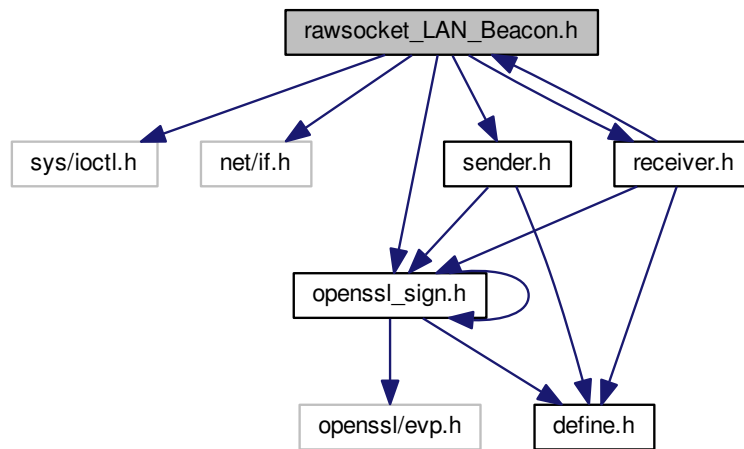
Success or error codes

**4.4 rawsocket\_LAN\_Beacon.h File Reference**

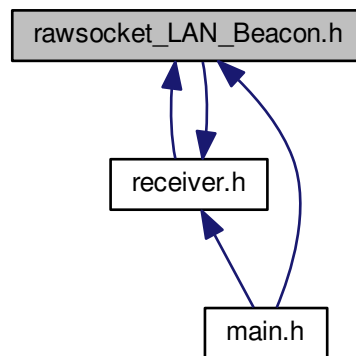
raw-socket sending and receiving

```
#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

<code>my_interfaces_struct</code>	Struct that contains interfaces information and configuration
<code>interface_to_send_on</code>	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

<i>my_receiver_information</i>	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

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

## Returns

Returns 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

<i>my_sender_information</i>	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

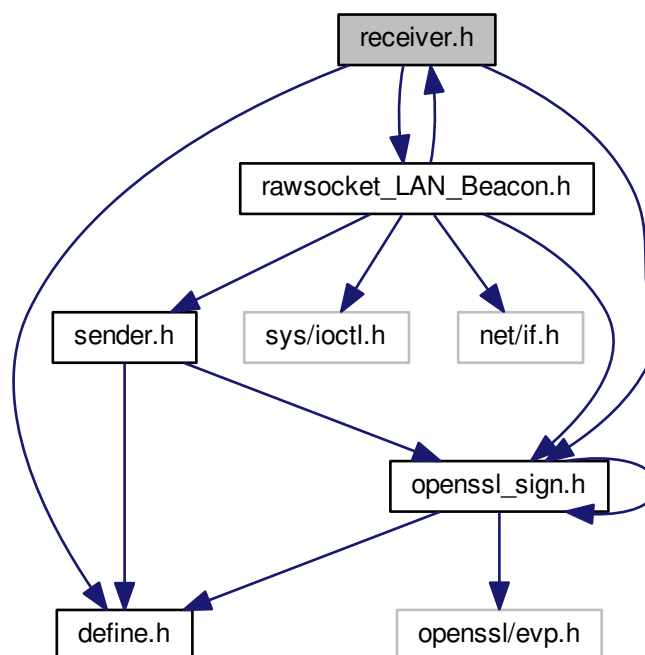
<i>destination_mac</i>	Destination MAC address
<i>payload</i>	Payload that should be sent
<i>payloadLen</i>	Length of payload
<i>etherType</i>	EtherType of payload
<i>lanbeacon_keys</i>	Keys that are used for sending
<i>interface_to_send_on</i>	Interface, that information should be sent on
<i>my_sender_information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

## 4.5 receiver.h File Reference

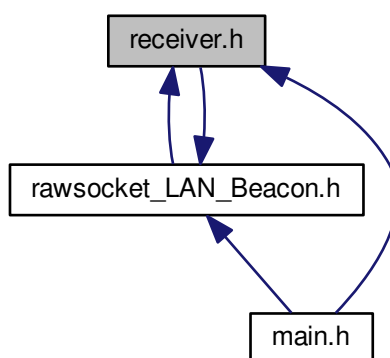
Receiver-specific functions and structures.

```
#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

<i>my_receiver_information</i>	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

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

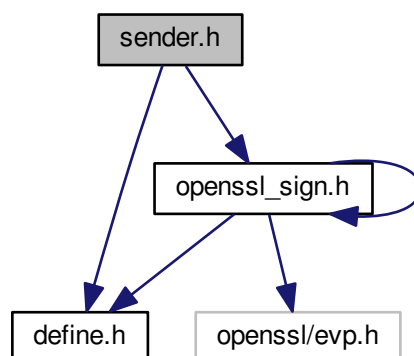
##### Returns

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

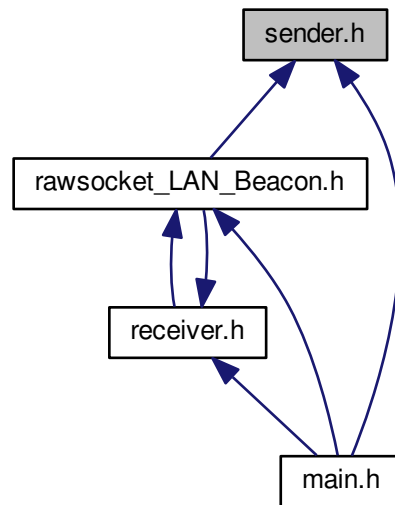
## 4.6 sender.h File Reference

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 \*\*combinedString, 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

<i>ip_V4or6</i>	Switch between IPv4 and IPv6 mode
<i>optarg</i>	String, which should be parsed
<i>combinedString</i>	Pointer to the string, that contains text representation of all contents
<i>combinedBeacon</i>	PDU of beacon, that TLVs should be added to
<i>currentByte</i>	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

<i>argc</i>	Number of arguments.
<i>argv</i>	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

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

<i>subtype</i>	Subtype of the TLV
<i>TLVdescription</i>	Descriptor string of the TLV
<i>combinedString</i>	Pointer to the string, that contains text representation of all contents
<i>source</i>	String contents, that should be included to the PDU
<i>combinedBeacon</i>	PDU of beacon, that TLVs should be added to
<i>currentByte</i>	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

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

# Index

- authenticated\_mode
  - receiver\_information, [10](#)
- bananaPIprint
  - receiver.h, [27](#)
- challenge
  - received\_lan\_beacon\_frame, [8](#)
- current\_destination\_mac
  - received\_lan\_beacon\_frame, [8](#)
- current\_lan\_beacon\_pdu\_for\_printing
  - receiver\_information, [10](#)
- define.h, [13](#)
- etherType
  - interfaces, [5](#)
- evaluatelanbeacon
  - receiver.h, [27](#)
- generate\_keys
  - open\_ssl\_keys, [7](#)
- getInterfaces
  - rawsocket\_LAN\_Beacon.h, [22](#)
- if\_idx
  - interfaces, [5](#)
- if\_mac
  - interfaces, [6](#)
- interface\_to\_send\_on
  - sender\_information, [12](#)
- interfaces, [5](#)
  - etherType, [5](#)
  - if\_idx, [5](#)
  - if\_mac, [6](#)
  - maxSockFd, [6](#)
  - numInterfaces, [6](#)
  - sendOrReceive, [6](#)
  - sockfd, [6](#)
  - sockopt, [6](#)
- ipParser
  - sender.h, [30](#)
- lan\_beacon\_ReceivedPayload
  - received\_lan\_beacon\_frame, [8](#)
- lan\_beacon\_pdu\_len
  - sender\_information, [12](#)
- lanBeacon\_PDU
  - sender\_information, [12](#)
- lanbeacon\_keys
  - receiver\_information, [10](#)
- sender\_information, [12](#)
- main
  - main.h, [16](#)
- main.h, [15](#)
  - main, [16](#)
- make\_keys
  - openssl\_sign.h, [18](#)
- maxSockFd
  - interfaces, [6](#)
- mergedlanbeaconCreator
  - sender.h, [30](#)
- my\_receiver\_interfaces
  - receiver\_information, [10](#)
- new\_lan\_beacon\_receiver
  - rawsocket\_LAN\_Beacon.h, [22](#)
- numInterfaces
  - interfaces, [6](#)
- number\_of\_currently\_received\_frames
  - receiver\_information, [10](#)
- open\_ssl\_keys, [7](#)
  - generate\_keys, [7](#)
  - path\_To\_Signing\_Key, [7](#)
  - path\_To\_Verifying\_Key, [7](#)
  - pcszPassphrase, [7](#)
  - sender\_or\_receiver\_mode, [7](#)
- openssl\_sign.h, [16](#)
  - make\_keys, [18](#)
  - print\_it, [18](#)
  - read\_keys, [19](#)
  - signlanbeacon, [19](#)
  - verifylanbeacon, [20](#)
- parsedBeaconContents
  - received\_lan\_beacon\_frame, [9](#)
- path\_To\_Signing\_Key
  - open\_ssl\_keys, [7](#)
- path\_To\_Verifying\_Key
  - open\_ssl\_keys, [7](#)
- payloadSize
  - received\_lan\_beacon\_frame, [9](#)
- pcszPassphrase
  - open\_ssl\_keys, [7](#)
- pointers\_to\_received\_frames
  - receiver\_information, [10](#)
- print\_it
  - openssl\_sign.h, [18](#)
- rawsocket\_LAN\_Beacon.h, [20](#)

- getInterfaces, 22
- new\_lan\_beacon\_receiver, 22
- receiveChallenge, 24
- send\_lan\_beacon\_rawSock, 24
- sendRawSocket, 24
- read\_keys
  - openssl\_sign.h, 19
- receiveChallenge
  - rawsocket\_LAN\_Beacon.h, 24
- received\_lan\_beacon\_frame, 8
  - challenge, 8
  - current\_destination\_mac, 8
  - lan\_beacon\_ReceivedPayload, 8
  - parsedBeaconContents, 9
  - payloadSize, 9
  - successfullyAuthenticated, 9
  - times\_left\_to\_display, 9
- receiver.h, 25
  - bananaPIprint, 27
  - evaluatelanbeacon, 27
- receiver\_information, 9
  - authenticated\_mode, 10
  - current\_lan\_beacon\_pdu\_for\_printing, 10
  - lanbeacon\_keys, 10
  - my\_receiver\_interfaces, 10
  - number\_of\_currently\_received\_frames, 10
  - pointers\_to\_received\_frames, 10
  - scroll\_speed, 11
- scroll\_speed
  - receiver\_information, 11
- send\_frequency
  - sender\_information, 12
- send\_lan\_beacon\_rawSock
  - rawsocket\_LAN\_Beacon.h, 24
- sendOrReceive
  - interfaces, 6
- sendRawSocket
  - rawsocket\_LAN\_Beacon.h, 24
- sender.h, 28
  - ipParser, 30
  - mergedlanbeaconCreator, 30
  - transferToCombinedBeacon, 31
  - transferToCombinedBeaconAndString, 31
  - transferToCombinedString, 31
- sender\_information, 11
  - interface\_to\_send\_on, 12
  - lan\_beacon\_pdu\_len, 12
  - lanBeacon\_PDU, 12
  - lanbeacon\_keys, 12
  - send\_frequency, 12
- sender\_or\_receiver\_mode
  - open\_ssl\_keys, 7
- signlanbeacon
  - openssl\_sign.h, 19
- sockfd
  - interfaces, 6
- sockopt
  - interfaces, 6
- successfullyAuthenticated
  - received\_lan\_beacon\_frame, 9
- times\_left\_to\_display
  - received\_lan\_beacon\_frame, 9
- transferToCombinedBeacon
  - sender.h, 31
- transferToCombinedBeaconAndString
  - sender.h, 31
- transferToCombinedString
  - sender.h, 31
- verifylanbeacon
  - openssl\_sign.h, 20