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 files with brief descriptions:

Chapter 3

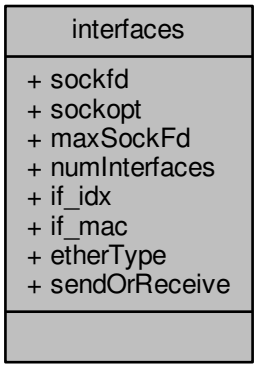
Data Structure Documentation

3.1 interfaces Struct Reference

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

`#include <receiver.h>`

Collaboration diagram for interfaces:



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.

Referenced by getInterfaces(), and sendRawSocket().

3.1.2.2 if_idx

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

Interface IDs.

Referenced by getInterfaces(), and sendRawSocket().

3.1.2.3 if_mac

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

Interface MACs.

Referenced by getInterfaces(), and sendRawSocket().

3.1.2.4 maxSockFd

```
int interfaces::maxSockFd
```

Needed for select function.

Referenced by getInterfaces(), lan_beacon_receiver(), and receiveChallenge().

3.1.2.5 numInterfaces

```
int interfaces::numInterfaces
```

Number of used interfaces.

Referenced by getInterfaces(), lan_beacon_receiver(), receiveChallenge(), and sendRawSocket().

3.1.2.6 sendOrReceive

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

Switch for send or receive mode.

Referenced by getInterfaces().

3.1.2.7 sockfd

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

File descriptors of raw sockets.

Referenced by getInterfaces(), lan_beacon_receiver(), receiveChallenge(), and sendRawSocket().

3.1.2.8 sockopt

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

. Options for each raw socket.

Referenced by getInterfaces().

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

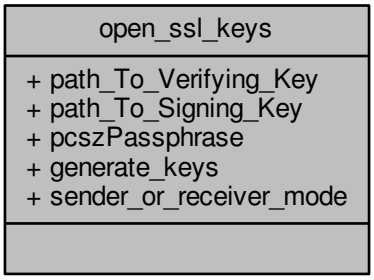
- receiver.h

3.2 open_ssl_keys Struct Reference

Key locations, password and further configurations.

`#include <openssl_sign.h>`

Collaboration diagram for open_ssl_keys:



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.

Referenced by lanbeacon_creator(), and signlanbeacon().

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.

Referenced by lanbeacon_creator(), make_keys(), and read_keys().

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.

Referenced by lanbeacon_creator(), make_keys(), read_keys(), receiver(), signlanbeacon(), and verifylanbeacon().

3.2.2.4 pcszPassphrase

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

Specified password for private key.

Referenced by lanbeacon_creator(), make_keys(), and read_keys().

3.2.2.5 sender_or_receiver_mode

```
int open_ssl_keys::sender_or_receiver_mode
```

Flag for corresponding client mode.

Referenced by read_keys(), and receiver().

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

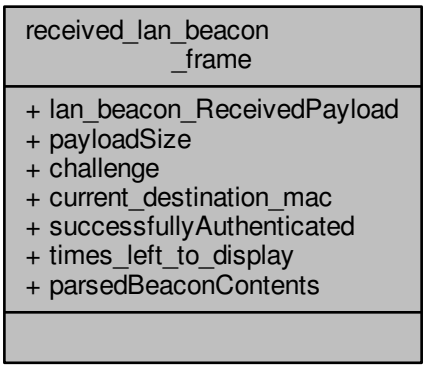
- openssl_sign.h

3.3 received_lan_beacon_frame Struct Reference

Contains all the information related to one received frame.

`#include <receiver.h>`

Collaboration diagram for received_lan_beacon_frame:



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.

Referenced by evaluatelanbeacon(), and lan_beacon_receiver().

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.

Referenced by lan_beacon_receiver().

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.

Referenced by evaluatelanbeacon(), and lan_beacon_receiver().

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.

Referenced by bananaPIprint(), lan_beacon_receiver(), and receiver().

3.3.2.5 payloadSize

```
ssize_t received_lan_beacon_frame::payloadSize
```

The size of the raw payload.

Referenced by evaluatelanbeacon(), and lan_beacon_receiver().

3.3.2.6 successfullyAuthenticated

```
int received_lan_beacon_frame::successfullyAuthenticated
```

Has frame already been authenticated?

Referenced by evaluatelanbeacon(), and lan_beacon_receiver().

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.

Referenced by bananaPIprint(), and lan_beacon_receiver().

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

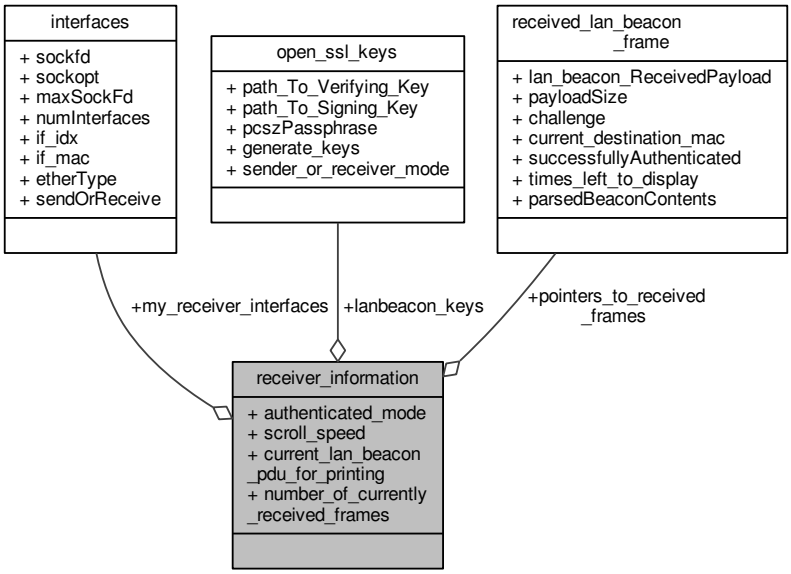
- receiver.h

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?

Referenced by lan_beacon_receiver(), and receiver().

3.4.2.2 current_lan_beacon_pdu_for_printing

```
int receiver_information::current_lan_beacon_pdu_for_printing
```

The currently printed PDU.

Referenced by bananaPIprint(), and receiver().

3.4.2.3 lanbeacon_keys

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

The paths to the keys.

Referenced by lan_beacon_receiver(), and receiver().

3.4.2.4 my_receiver_interfaces

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

Interfaces, that are used for LAN-Beacon reception.

Referenced by lan_beacon_receiver(), and receiver().

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.

Referenced by bananaPIprint(), lan_beacon_receiver(), and receiver().

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.

Referenced by bananaPIprint(), lan_beacon_receiver(), and receiver().

3.4.2.7 scroll_speed

```
int receiver_information::scroll_speed
```

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

Referenced by bananaPIprint(), and receiver().

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

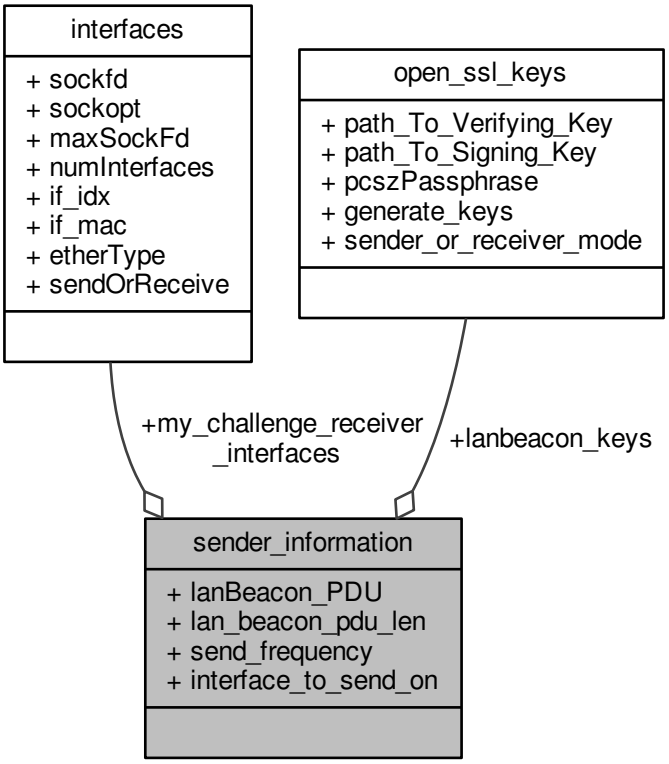
- receiver.h

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 interfaces my_challenge_receiver_interfaces
- 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 by start parameters, interface that is used for sending.

Referenced by lanbeacon_creator(), sender(), and sendRawSocket().

3.5.2.2 lan_beacon_pdu_len

```
int sender_information::lan_beacon_pdu_len
```

Length of the combined PDU.

Referenced by lanbeacon_creator(), and send_lan_beacon_rawSock().

3.5.2.3 lanbeacon_keys

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

Keys configuration.

Referenced by lanbeacon_creator(), and sendRawSocket().

3.5.2.4 lanBeacon_PDU

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

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

Referenced by send_lan_beacon_rawSock(), and sender().

3.5.2.5 my_challenge_receiver_interfaces

```
struct interfaces sender_information::my_challenge_receiver_interfaces
```

Interfaces that are used for receiving challenges.

Referenced by sender(), and sendRawSocket().

3.5.2.6 send_frequency

```
int sender_information::send_frequency
```

Number of seconds between each sent PDU.

Referenced by lanbeacon_creator(), receiveChallenge(), and sendRawSocket().

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

- sender.h

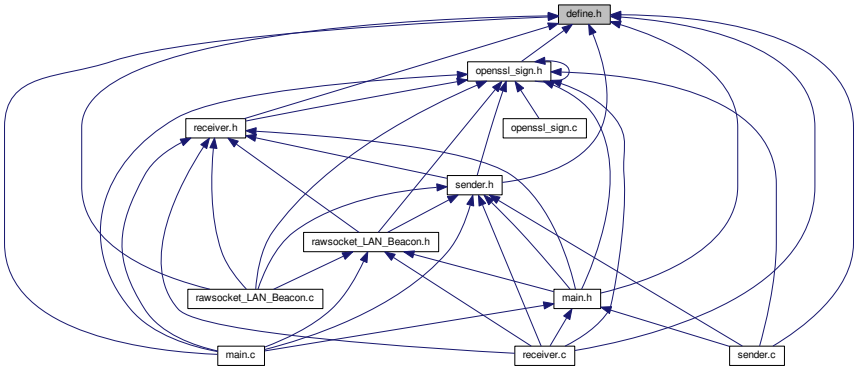
Chapter 4

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

Macro for gettext localization support

- #define _(STRING) gettext(STRING)

Protocol options such send frequency

- #define LAN_BEACON_SEND_FREQUENCY 5

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 5
- #define SHOW_FRAMES_X_TIMES 2
- #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

4.1.2 Macro Definition Documentation

4.1.2.1 _

```
#define _(
              STRING ) gettext(STRING)
```

Referenced by bananaPIprint(), evaluatelanbeacon(), getInterfaces(), ipParser(), lan_beacon_receiver(), lanbeacon_creator(), make_keys(), printHelp(), read_keys(), receiveChallenge(), receiver(), sendRawSocket(), signlanbeacon(), transfer_to_pdu(), transfer_to_string(), and verifylanbeacon().

4.1.2.2 CHALLENGE_ETHTYPE

```
#define CHALLENGE_ETHTYPE 0x88B6
```

Referenced by lan_beacon_receiver(), sender(), and sendRawSocket().

4.1.2.3 DEFAULT_SCROLLSPEED

```
#define DEFAULT_SCROLLSPEED 5
```

Referenced by receiver().

4.1.2.4 DESCRIPTOR_COMBINED_STRING

```
#define DESCRIPTOR_COMBINED_STRING gettext("Combined String:")
```

Referenced by evaluatelanbeacon().

4.1.2.5 DESCRIPTOR_CUSTOM

```
#define DESCRIPTOR_CUSTOM gettext("Freetext:")
```

Referenced by evaluatelanbeacon(), and lanbeacon_creator().

4.1.2.6 DESCRIPTOR_DHCP

```
#define DESCRIPTOR_DHCP gettext("DHCP:")
```

Referenced by evaluatelanbeacon(), and lanbeacon_creator().

4.1.2.7 DESCRIPTOR_EMAIL

```
#define DESCRIPTOR_EMAIL gettext("Email:")
```

Referenced by evaluatelanbeacon(), and lanbeacon_creator().

4.1.2.8 DESCRIPTOR_IPV4

```
#define DESCRIPTOR_IPV4 gettext("IPv4:")
```

Referenced by evaluatelanbeacon(), and ipParser().

4.1.2.9 DESCRIPTOR_IPV6

```
#define DESCRIPTOR_IPV6 gettext("IPv6:")
```

Referenced by evaluatelanbeacon(), and ipParser().

4.1.2.10 DESCRIPTOR_NAME

```
#define DESCRIPTOR_NAME gettext("VLAN-Name:")
```

Referenced by evaluatelanbeacon(), and lanbeacon_creator().

4.1.2.11 DESCRIPTOR_ROUTER

```
#define DESCRIPTOR_ROUTER gettext("Router:")
```

Referenced by evaluatelanbeacon(), and lanbeacon_creator().

4.1.2.12 DESCRIPTOR_SIGNATURE

```
#define DESCRIPTOR_SIGNATURE gettext("Authentication:")
```

Referenced by evaluatelanbeacon().

4.1.2.13 DESCRIPTOR_VLAN_ID

```
#define DESCRIPTOR_VLAN_ID gettext("VLAN-ID:")
```

Referenced by evaluatelanbeacon(), and lanbeacon_creator().

4.1.2.14 DESCRIPTOR_WIDTH

```
#define DESCRIPTOR_WIDTH 10
```

Referenced by bananaPIprint().

4.1.2.15 KEY_PATHLENGTH_MAX

```
#define KEY_PATHLENGTH_MAX 800
```

Referenced by lanbeacon_creator(), and receiver().

4.1.2.16 LAN_BEACON_BUF_SIZ

```
#define LAN_BEACON_BUF_SIZ 2000
```

Referenced by lan_beacon_receiver(), and sendRawSocket().

4.1.2.17 LAN_BEACON_DEST_MAC

```
#define LAN_BEACON_DEST_MAC 0xff, 0xff, 0xff, 0xff, 0xff, 0xff
```

Referenced by lan_beacon_receiver(), and send_lan_beacon_rawSock().

4.1.2.18 LAN_BEACON_ETHER_TYPE

```
#define LAN_BEACON_ETHER_TYPE 0x88B5
```

Referenced by receiver(), send_lan_beacon_rawSock(), and sendRawSocket().

4.1.2.19 LAN_BEACON_SEND_FREQUENCY

```
#define LAN_BEACON_SEND_FREQUENCY 5
```

Referenced by sender().

4.1.2.20 PARSED_TLVS_MAX_LENGTH

```
#define PARSED_TLVS_MAX_LENGTH 510
```

Referenced by evaluatelanbeacon().

4.1.2.21 PARSED_TLVS_MAX_NUMBER

```
#define PARSED_TLVS_MAX_NUMBER 25
```

Referenced by bananaPIprint(), evaluatelanbeacon(), and receiver().

4.1.2.22 PRIVATE_KEY_STANDARD_PATH

```
#define PRIVATE_KEY_STANDARD_PATH "private_key.pem"
```

Referenced by sender().

4.1.2.23 PUBLIC_KEY_STANDARD_PATH

```
#define PUBLIC_KEY_STANDARD_PATH "public_key.pem"
```

Referenced by receiver(), and sender().

4.1.2.24 SHOW_FRAMES_X_TIMES

```
#define SHOW_FRAMES_X_TIMES 2
```

Referenced by lan_beacon_receiver().

4.1.2.25 SUBTYPE_COMBINED_STRING

```
#define SUBTYPE_COMBINED_STRING 217
```

Referenced by evaluatelanbeacon(), and lanbeacon_creator().

4.1.2.26 SUBTYPE_CUSTOM

```
#define SUBTYPE_CUSTOM 202
```

Referenced by evaluatelanbeacon(), and lanbeacon_creator().

4.1.2.27 SUBTYPE_DHCP

```
#define SUBTYPE_DHCP 206
```

Referenced by evaluatelanbeacon(), and lanbeacon_creator().

4.1.2.28 SUBTYPE_EMAIL

```
#define SUBTYPE_EMAIL 205
```

Referenced by evaluatelanbeacon(), and lanbeacon_creator().

4.1.2.29 SUBTYPE_IPV4

```
#define SUBTYPE_IPV4 203
```

Referenced by evaluatelanbeacon(), and ipParser().

4.1.2.30 SUBTYPE_IPV6

```
#define SUBTYPE_IPV6 204
```

Referenced by evaluatelanbeacon(), and ipParser().

4.1.2.31 SUBTYPE_NAME

```
#define SUBTYPE_NAME 201
```

Referenced by evaluatelanbeacon(), and lanbeacon_creator().

4.1.2.32 SUBTYPE_ROUTER

```
#define SUBTYPE_ROUTER 207
```

Referenced by evaluatelanbeacon(), and lanbeacon_creator().

4.1.2.33 SUBTYPE_SIGNATURE

```
#define SUBTYPE_SIGNATURE 216
```

Referenced by evaluatelanbeacon(), and sendRawSocket().

4.1.2.34 SUBTYPE_VLAN_ID

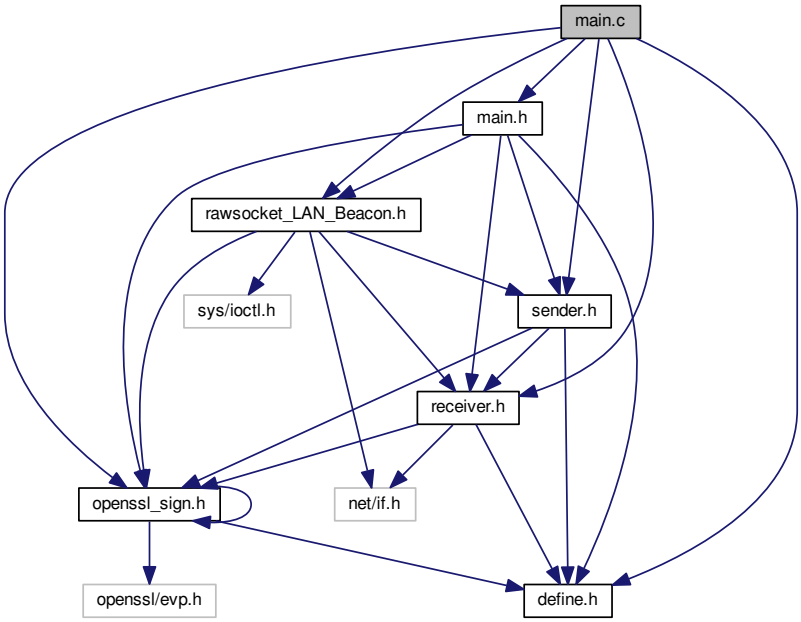
```
#define SUBTYPE_VLAN_ID 200
```

Referenced by evaluatelanbeacon(), and lanbeacon_creator().

4.2 main.c File Reference

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

Include dependency graph for main.c:



Functions

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

  Separates receiver from sender mode and some setup.
- void printHelp ()

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

4.2.1 Function Documentation

4.2.1.1 main()

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

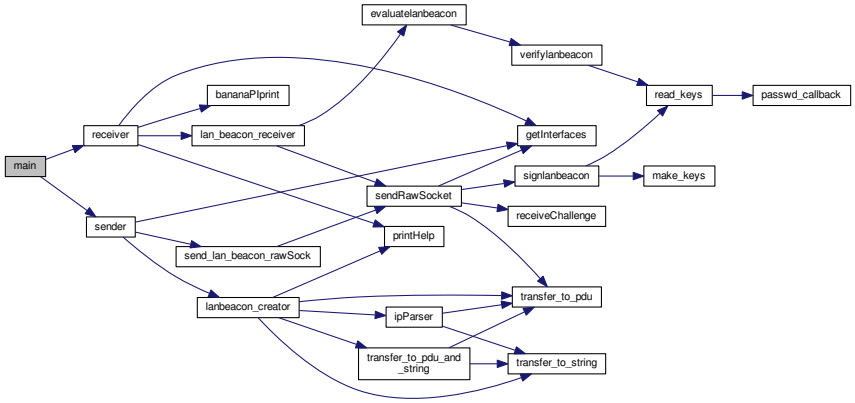
Separates receiver from sender mode and some setup.

Returns

Success or failure code.

References receiver(), and sender().

Here is the call graph for this function:



4.2.1.2 printHelp()

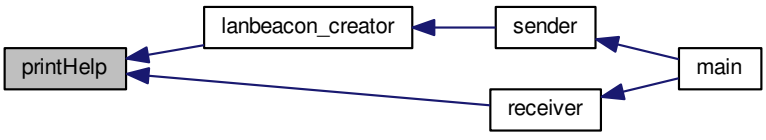
```
void printHelp ( )
```

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

References _.

Referenced by lanbeacon_creator(), and receiver().

Here is the caller graph for this function:

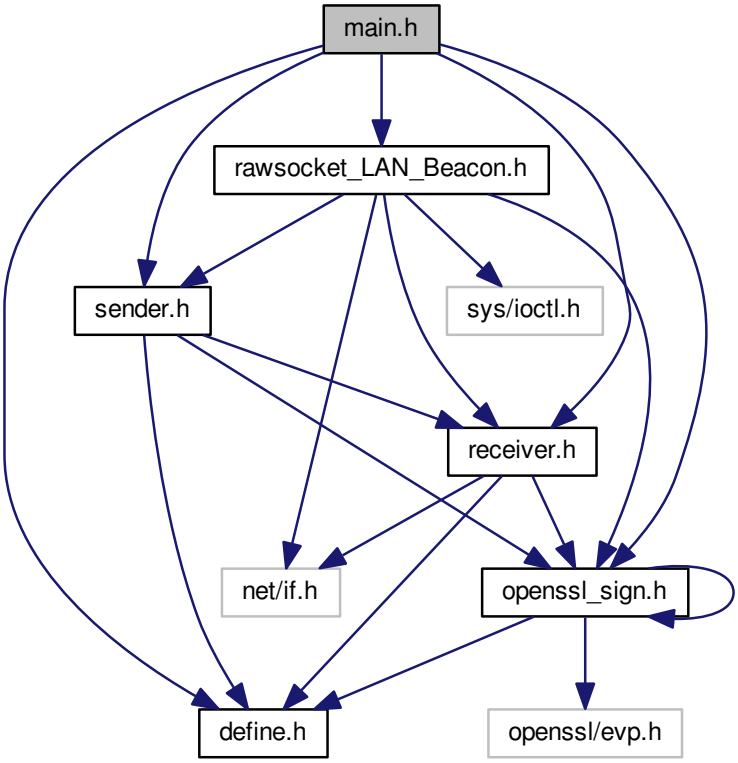


4.3 main.h File Reference

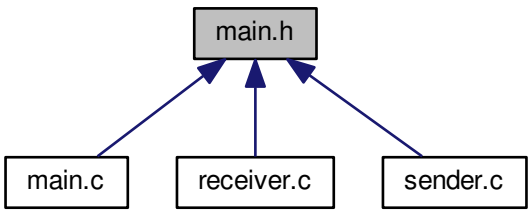
Main function and help function.

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

Include dependency graph for main.h:



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



Functions

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

  Separates receiver from sender mode and some setup.
- void printHelp ()

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

4.3.1 Detailed Description

Main function and help function.

Author

Dominik Bitzer

Date

2017

4.3.2 Function Documentation

4.3.2.1 main()

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

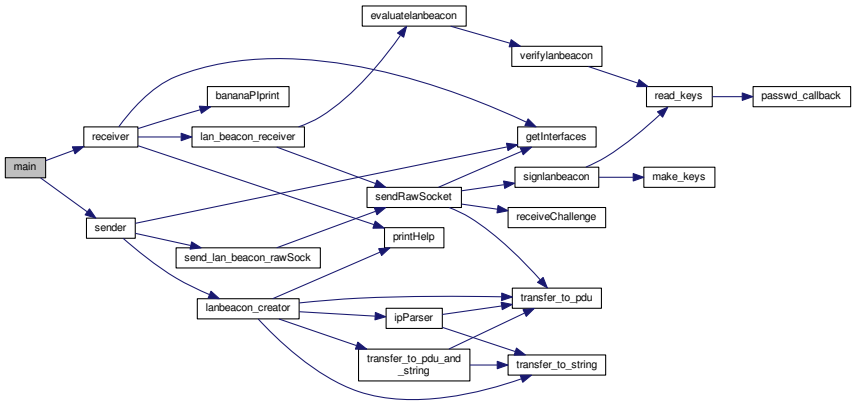
Separates receiver from sender mode and some setup.

Returns

Success or failure code.

References receiver(), and sender().

Here is the call graph for this function:



4.3.2.2 printHelp()

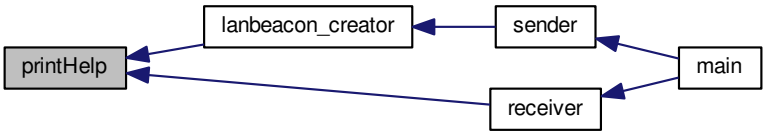
```
void printHelp ( )
```

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

References _.

Referenced by lanbeacon_creator(), and receiver().

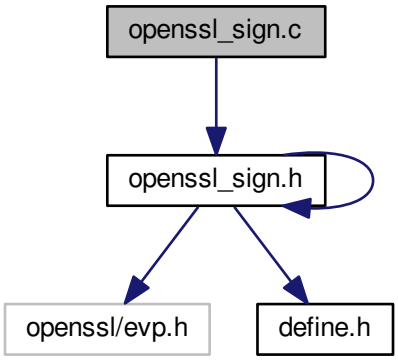
Here is the caller graph for this function:



4.4 openssl_sign.c File Reference

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

Include dependency graph for openssl_sign.c:



Macros

- #define KEY_READ_PROBLEM 0b1
- #define VERFIY_PROBLEM 0b01
- #define NO_PRIVATE_KEY 0b001
- #define NO_PUBLIC_KEY 0b0001
- #define PROBLEM_IN_SIGN_CALL 0b00001
- #define PROBLEM_IN_VERIFY_CALL 0b000001
- #define SIG_LEN 256

Functions

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

  Verify the signature for LAN-Beacon PDUs.
- int signlanbeacon (unsigned char **sig, size_t *slen, const unsigned char *msg, size_t mlen, struct open_ssl_keys *lanbeacon_keys)

  Create signature for LAN-Beacon PDU.
- void print_it (const char *label, const unsigned char *buff, size_t len)

  Prints a buffer to stdout. Label is optional.
- int read_keys (EVP_PKEY **skey, EVP_PKEY **vkey, struct open_ssl_keys *lanbeacon_keys)

  Read stored pem files into memory.
- int make_keys (EVP_PKEY **skey, EVP_PKEY **vkey, struct open_ssl_keys *lanbeacon_keys)

  Generate and save keys to specified paths.
- int passwd_callback (char *pcszBuff, int size, int rwflag, void *pPass)

  Password callback function to retrieve password from configuration.

Variables

- const char hn [ ] = "SHA256"

4.4.1 Macro Definition Documentation

4.4.1.1 KEY_READ_PROBLEM

```
#define KEY_READ_PROBLEM 0b1
```

4.4.1.2 NO_PRIVATE_KEY

```
#define NO_PRIVATE_KEY 0b001
```

Referenced by read_keys(), and signlanbeacon().

4.4.1.3 NO_PUBLIC_KEY

```
#define NO_PUBLIC_KEY 0b0001
```

Referenced by read_keys(), and verifylanbeacon().

4.4.1.4 PROBLEM_IN_SIGN_CALL

```
#define PROBLEM_IN_SIGN_CALL 0b00001
```

Referenced by signlanbeacon().

4.4.1.5 PROBLEM_IN_VERIFY_CALL

```
#define PROBLEM_IN_VERIFY_CALL 0b000001
```

Referenced by verifylanbeacon().

4.4.1.6 SIG_LEN

```
#define SIG_LEN 256
```

4.4.1.7 VERFIY_PROBLEM

```
#define VERFIY_PROBLEM 0b01
```

4.4.2 Function Documentation

4.4.2.1 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

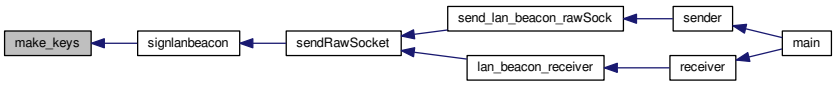
Returns

Returns 0 for success, non-0 otherwise

References _, open_ssl_keys::path_To_Signing_Key, open_ssl_keys::path_To_Verifying_Key, and open_ssl_↩ keys::pcszPassphrase.

Referenced by signlanbeacon().

Here is the caller graph for this function:



4.4.2.2 passwd_callback()

```
int passwd_callback (
            char * pcszBuff,
            int size,
            int rwflag,
            void * pPass )
```

Password callback function to retrieve password from configuration.

Parameters

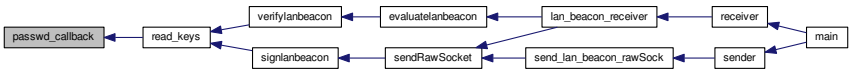
pcszBuff	Buffer for password
size	Size of buffer
rwflag	Read/write flag
pPass	Password

Returns

Success or error codes

Referenced by read_keys().

Here is the caller graph for this function:



4.4.2.3 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

4.4.2.4 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

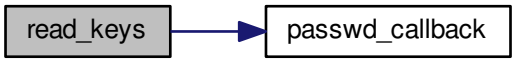
Returns

Success or error codes

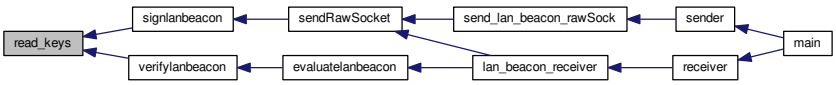
References _, NO_PRIVATE_KEY, NO_PUBLIC_KEY, passwd_callback(), open_ssl_keys::path_To_Signing_Key, open_ssl_keys::path_To_Verifying_Key, open_ssl_keys::pcszPassphrase, RECEIVER_MODE, SENDER_MODE, and open_ssl_keys::sender_or_receiver_mode.

Referenced by signlanbeacon(), and verifylanbeacon().

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.2.5 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

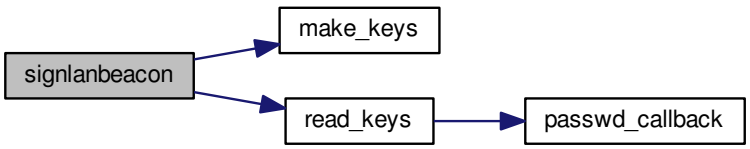
Returns

Success or error codes

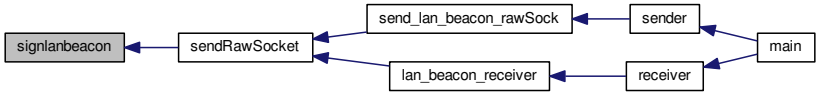
References _, open_ssl_keys::generate_keys, hn, make_keys(), NO_PRIVATE_KEY, open_ssl_keys::path_To_↩ Verifying_Key, PROBLEM_IN_SIGN_CALL, and read_keys().

Referenced by sendRawSocket().

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.2.6 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

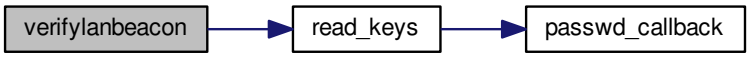
Returns

Success or error codes

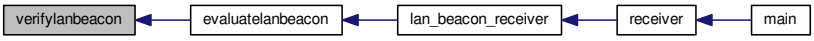
References _, hn, NO_PUBLIC_KEY, open_ssl_keys::path_To_Verifying_Key, PROBLEM_IN_VERIFY_CALL, and read_keys().

Referenced by evaluatelanbeacon().

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.3 Variable Documentation

4.4.3.1 hn

```
const char hn[] = "SHA256"
```

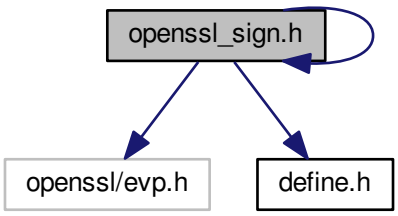
Referenced by signlanbeacon(), and verifylanbeacon().

4.5 openssl_sign.h File Reference

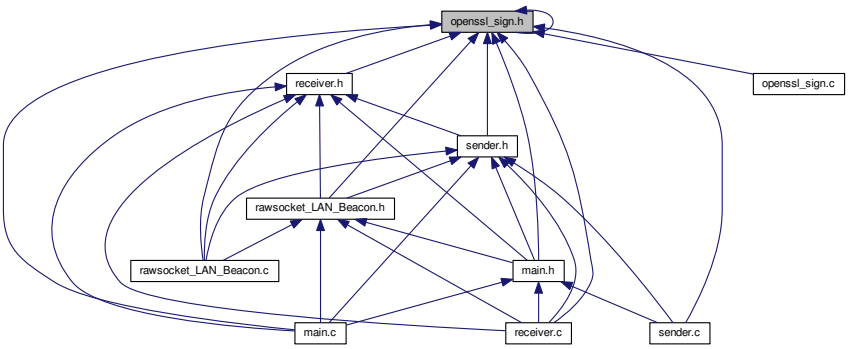
signing, verifying and key I/O

```
#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)

  Password callback function to retrieve password from configuration.
- 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.

4.5.1 Detailed Description

signing, verifying and key I/O

Author

Dominik Bitzer

Date

2017

4.5.2 Macro Definition Documentation

4.5.2.1 RECEIVER_MODE

```
#define RECEIVER_MODE 1
```

Referenced by read_keys(), and receiver().

4.5.2.2 SENDER_MODE

```
#define SENDER_MODE 0
```

Referenced by read_keys(), and sender().

4.5.3 Function Documentation

4.5.3.1 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

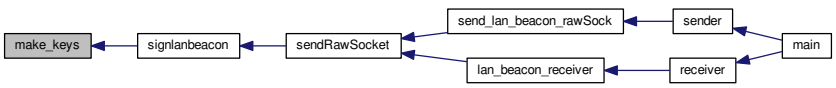
Returns

Returns 0 for success, non-0 otherwise

References _, open_ssl_keys::path_To_Signing_Key, open_ssl_keys::path_To_Verifying_Key, and open_ssl_keys::pcszPassphrase.

Referenced by signlanbeacon().

Here is the caller graph for this function:



4.5.3.2 passwd_callback()

```
int passwd_callback (
             char * pcszBuff,
             int size,
             int rwflag,
             void * pPass )
```

Password callback function to retrieve password from configuration.

Parameters

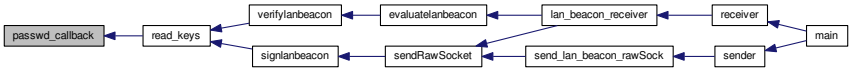
pcszBuff	Buffer for password
size	Size of buffer
rwflag	Read/write flag
pPass	Password

Returns

Success or error codes

Referenced by read_keys().

Here is the caller graph for this function:



4.5.3.3 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

4.5.3.4 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

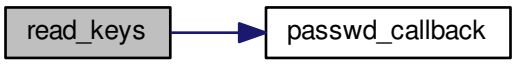
Returns

Success or error codes

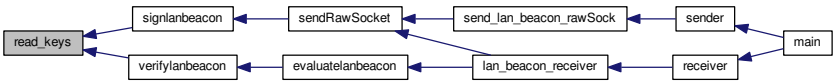
References _, NO_PRIVATE_KEY, NO_PUBLIC_KEY, passwd_callback(), open_ssl_keys::path_To_Signing_Key, open_ssl_keys::path_To_Verifying_Key, open_ssl_keys::pcszPassphrase, RECEIVER_MODE, SENDER_MODE, and open_ssl_keys::sender_or_receiver_mode.

Referenced by signlanbeacon(), and verifylanbeacon().

Here is the call graph for this function:



Here is the caller graph for this function:



4.5.3.5 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

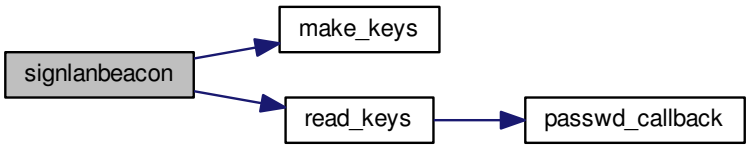
Returns

Success or error codes

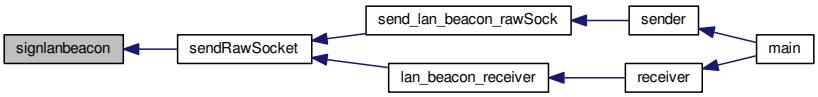
References _, open_ssl_keys::generate_keys, hn, make_keys(), NO_PRIVATE_KEY, open_ssl_keys::path_To_↩ Verifying_Key, PROBLEM_IN_SIGN_CALL, and read_keys().

Referenced by sendRawSocket().

Here is the call graph for this function:



Here is the caller graph for this function:



4.5.3.6 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

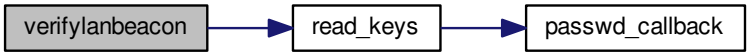
Returns

Success or error codes

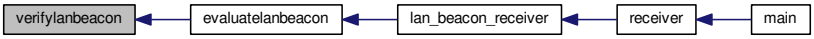
References _, hn, NO_PUBLIC_KEY, open_ssl_keys::path_To_Verifying_Key, PROBLEM_IN_VERIFY_CALL, and read_keys().

Referenced by evaluatelanbeacon().

Here is the call graph for this function:



Here is the caller graph for this function:

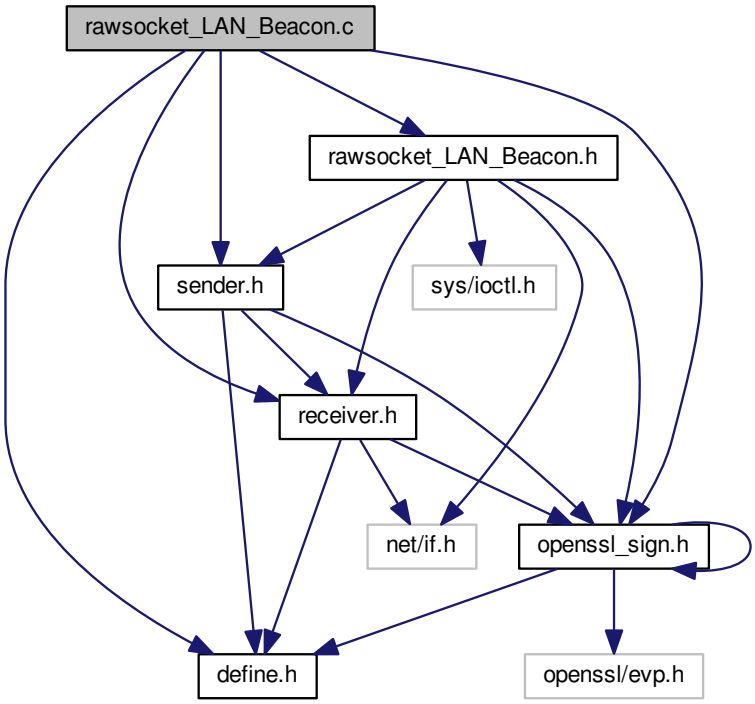


4.6 rawsocket_LAN_Beacon.c File Reference

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

```
#include "sender.h"
```

Include dependency graph for rawsocket_LAN_Beacon.c:



Macros

- #define _GNU_SOURCE

Functions

- void sendRawSocket (unsigned char *destination_mac, void *payload, int payloadLen, unsigned short etherType, struct sender_information *my_sender_information)

  Generic function to send on raw sockets, both handles sending of LAN-Beacon and challenges.
- int send_lan_beacon_rawSock (struct sender_information *my_sender_information)

  Shortcut that can be used for sending LAN-Beacons, provides some configuration already.
- void lan_beacon_receiver (struct receiver_information *my_receiver_information)

  Receives LAN-Beacons and adds them to the structure of received beacons.
- unsigned long receiveChallenge (struct interfaces *my_challenge_receiver_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.

4.6.1 Macro Definition Documentation

4.6.1.1 _GNU_SOURCE

```
#define _GNU_SOURCE
```

4.6.2 Function Documentation

4.6.2.1 getInterfaces()

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

Get raw sockets for interfaces.

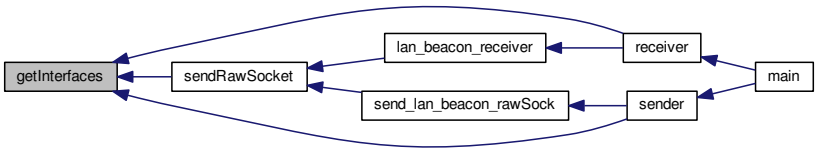
Parameters

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

References _, interfaces::etherType, interfaces::if_idx, interfaces::if_mac, interfaces::maxSockFd, interfaces::numInterfaces, REC_SOCKET, SEND_SOCKET, interfaces::sendOrReceive, interfaces::sockfd, and interfaces::sockopt.

Referenced by receiver(), sender(), and sendRawSocket().

Here is the caller graph for this function:



4.6.2.2 lan_beacon_receiver()

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

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

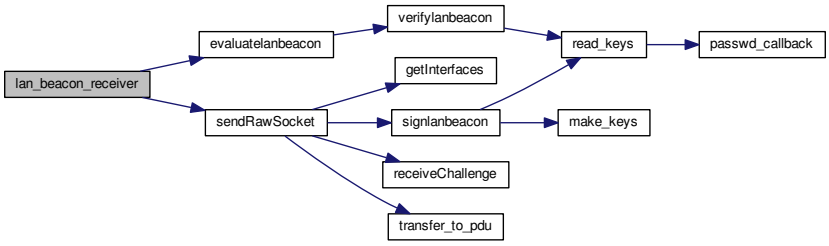
Parameters

my_receiver_information	Receiver configuration and structs for storing the received beacons

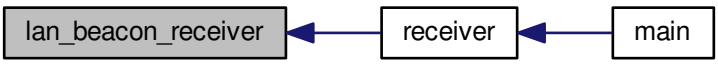
References _, receiver_information::authenticated_mode, received_lan_beacon_frame::challenge, CHALLENGE_ETHTYPE, received_lan_beacon_frame::current_destination_mac, evaluatelanbeacon(), LAN_BEACON_BUF_SIZ, LAN_BEACON_DEST_MAC, received_lan_beacon_frame::lan_beacon_ReceivedPayload, receiver_information::lanbeacon_keys, interfaces::maxSockFd, receiver_information::my_receiver_interfaces, receiver_information::number_of_currently_received_frames, interfaces::numInterfaces, received_lan_beacon_frame::parsedBeaconContents, received_lan_beacon_frame::payloadSize, receiver_information::pointers_to_received_frames, sendRawSocket(), SHOW_FRAMES_X_TIMES, interfaces::sockfd, received_lan_beacon_frame::successfullyAuthenticated, and received_lan_beacon_frame::times_left_to_display.

Referenced by receiver().

Here is the call graph for this function:



Here is the caller graph for this function:



4.6.2.3 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

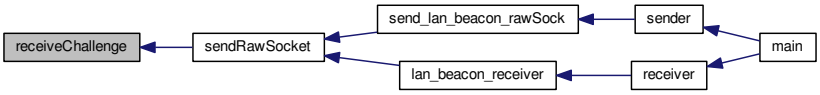
Returns

Returnes the value of the received challenge

References _, interfaces::maxSockFd, interfaces::numInterfaces, sender_information::send_frequency, and interfaces::sockfd.

Referenced by sendRawSocket().

Here is the caller graph for this function:



4.6.2.4 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

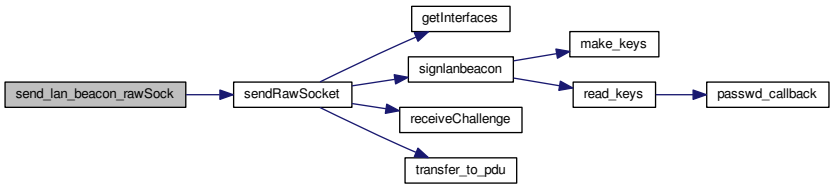
Returns

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

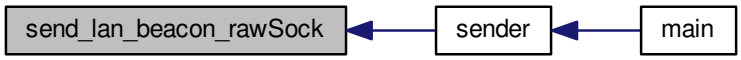
References LAN_BEACON_DEST_MAC, LAN_BEACON_ETHER_TYPE, sender_information::lan_beacon_pdu↩_len, sender_information::lanBeacon_PDU, and sendRawSocket().

Referenced by sender().

Here is the call graph for this function:



Here is the caller graph for this function:



4.6.2.5 sendRawSocket()

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

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

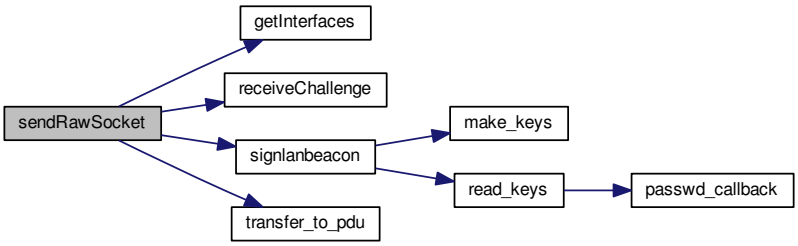
Parameters

destination_mac	Destination MAC address
payload	Payload that should be sent
payloadLen	Length of payload
etherType	EtherType of payload
my_sender_information	Sender configurations

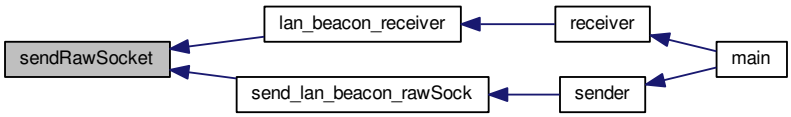
References _, CHALLENGE_ETHTYPE, interfaces::etherType, getInterfaces(), interfaces::if_idx, interfaces::if_mac, sender_information::interface_to_send_on, LAN_BEACON_BUF_SIZ, LAN_BEACON_ETHER_TYPE, sender↩_information::lanbeacon_keys, sender_information::my_challenge_receiver_interfaces, interfaces::numInterfaces, receiveChallenge(), sender_information::send_frequency, SEND_SOCKET, signlanbeacon(), interfaces::sockfd, SUBTYPE_SIGNATURE, and transfer_to_pdu().

Referenced by lan_beacon_receiver(), and send_lan_beacon_rawSock().

Here is the call graph for this function:



Here is the caller graph for this function:

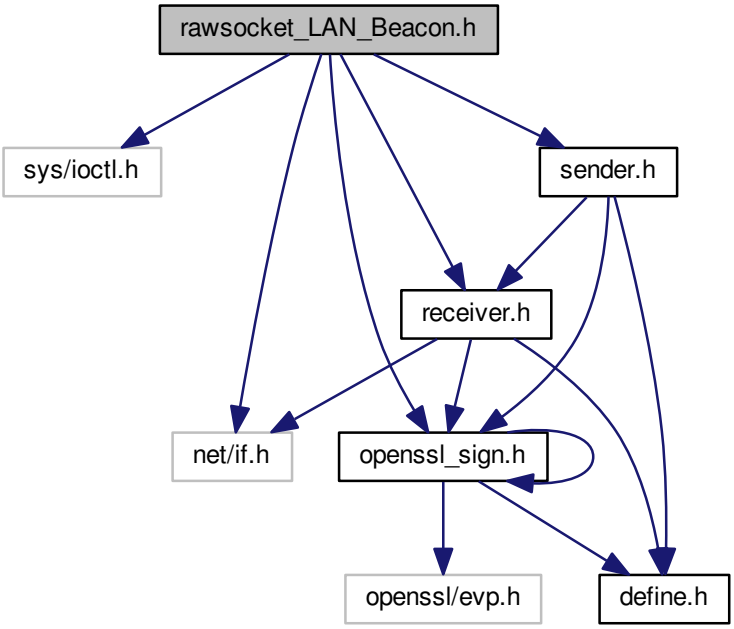


4.7 rawsocket_LAN_Beacon.h File Reference

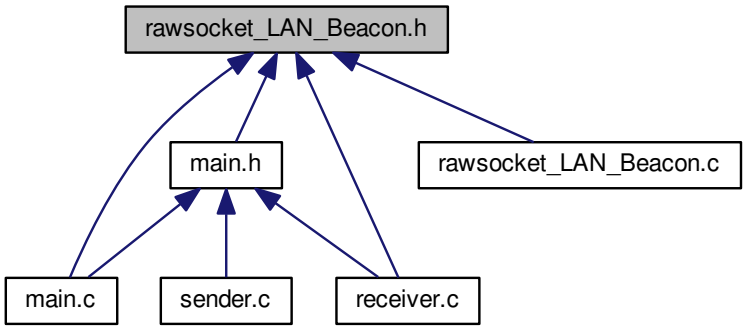
raw-socket sending and receiving

```
#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 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 sender_information ∗my_sender_information)

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

4.7.1 Detailed Description

raw-socket sending and receiving

Author

Dominik Bitzer

Date

2017

4.7.2 Macro Definition Documentation

4.7.2.1 REC_SOCKET

```
#define REC_SOCKET 1
```

Referenced by getInterfaces(), receiver(), and sender().

4.7.2.2 SEND_SOCKET

```
#define SEND_SOCKET 0
```

Referenced by getInterfaces(), and sendRawSocket().

4.7.3 Function Documentation

4.7.3.1 getInterfaces()

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

Get raw sockets for interfaces.

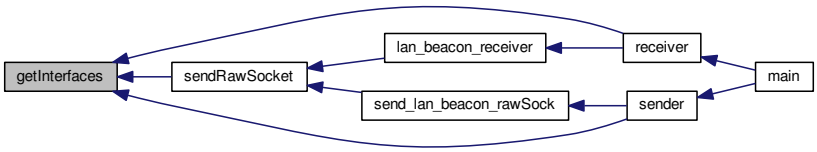
Parameters

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

References _, interfaces::etherType, interfaces::if_idx, interfaces::if_mac, interfaces::maxSockFd, interfaces::numInterfaces, REC_SOCKET, SEND_SOCKET, interfaces::sendOrReceive, interfaces::sockfd, and interfaces::sockopt.

Referenced by receiver(), sender(), and sendRawSocket().

Here is the caller graph for this function:



4.7.3.2 lan_beacon_receiver()

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

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

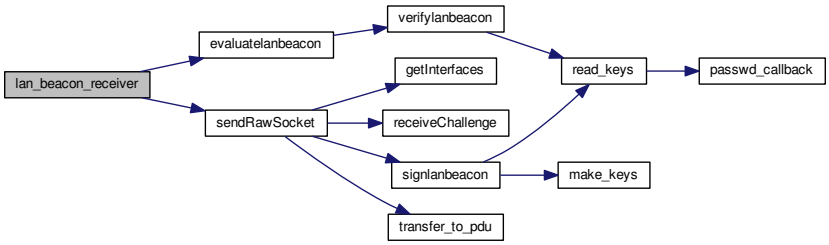
Parameters

my_receiver_information	Receiver configuration and structs for storing the received beacons

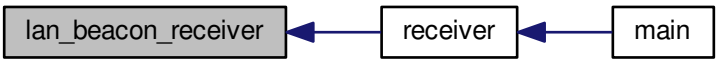
References _, receiver_information::authenticated_mode, received_lan_beacon_frame::challenge, CHALLENGE_ETHTYPE, received_lan_beacon_frame::current_destination_mac, evaluatelanbeacon(), LAN_BEACON_BUF_SIZ, LAN_BEACON_DEST_MAC, received_lan_beacon_frame::lan_beacon_ReceivedPayload, receiver_information::lanbeacon_keys, interfaces::maxSockFd, receiver_information::my_receiver_interfaces, receiver_information::number_of_currently_received_frames, interfaces::numInterfaces, received_lan_beacon_frame::parsedBeaconContents, received_lan_beacon_frame::payloadSize, receiver_information::pointers_to_received_frames, sendRawSocket(), SHOW_FRAMES_X_TIMES, interfaces::sockfd, received_lan_beacon_frame::successfullyAuthenticated, and received_lan_beacon_frame::times_left_to_display.

Referenced by receiver().

Here is the call graph for this function:



Here is the caller graph for this function:



4.7.3.3 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

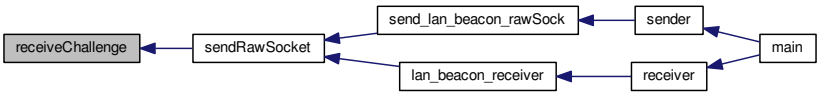
Returns

Returnes the value of the received challenge

References _, interfaces::maxSockFd, interfaces::numInterfaces, sender_information::send_frequency, and interfaces::sockfd.

Referenced by sendRawSocket().

Here is the caller graph for this function:



4.7.3.4 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

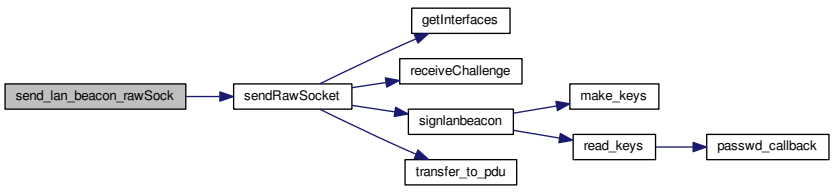
Returns

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

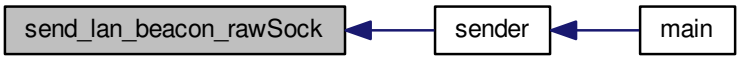
References LAN_BEACON_DEST_MAC, LAN_BEACON_ETHER_TYPE, sender_information::lan_beacon_pdu↩_len, sender_information::lanBeacon_PDU, and sendRawSocket().

Referenced by sender().

Here is the call graph for this function:



Here is the caller graph for this function:



4.7.3.5 sendRawSocket()

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

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

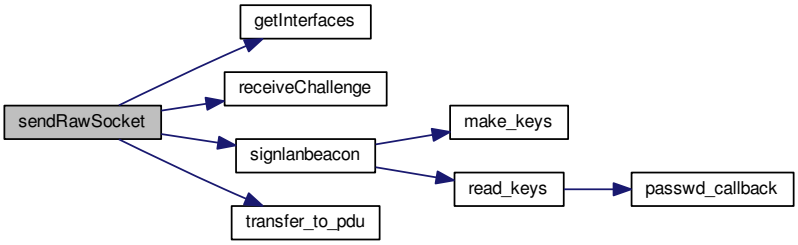
Parameters

destination_mac	Destination MAC address
payload	Payload that should be sent
payloadLen	Length of payload
etherType	EtherType of payload
my_sender_information	Sender configurations

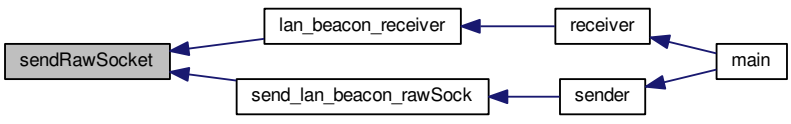
References _, CHALLENGE_ETHTYPE, interfaces::etherType, getInterfaces(), interfaces::if_idx, interfaces::if_mac, sender_information::interface_to_send_on, LAN_BEACON_BUF_SIZ, LAN_BEACON_ETHER_TYPE, sender_information::lanbeacon_keys, sender_information::my_challenge_receiver_interfaces, interfaces::numInterfaces, receiveChallenge(), sender_information::send_frequency, SEND_SOCKET, signlanbeacon(), interfaces::sockfd, SUBTYPE_SIGNATURE, and transfer_to_pdu().

Referenced by lan_beacon_receiver(), and send_lan_beacon_rawSock().

Here is the call graph for this function:



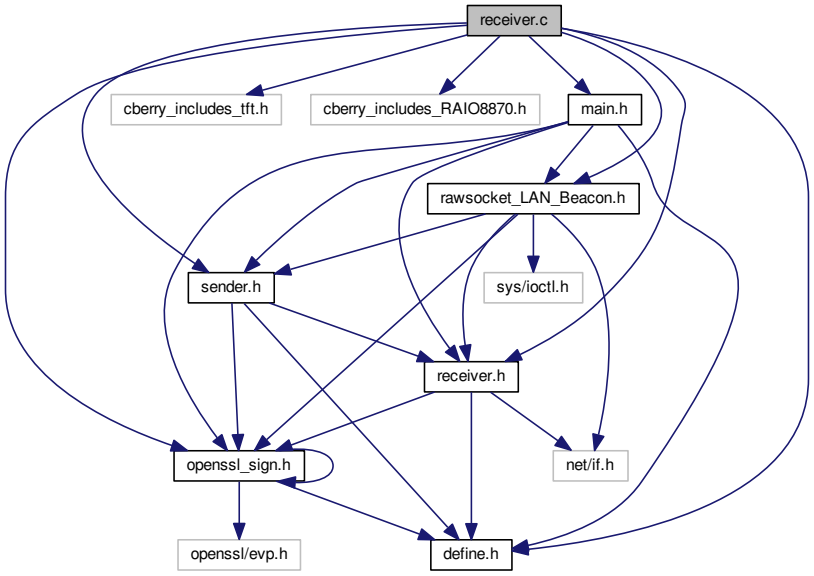
Here is the caller graph for this function:



4.8 receiver.c File Reference

```
#include "openssl_sign.h"
#include "cberry_includes_tft.h"
#include "cberry_includes_RAIO8870.h"
#include "rawsocket_LAN_Beacon.h"
#include "receiver.h"
#include "sender.h"
#include "define.h"
#include "main.h"
```

Include dependency graph for receiver.c:



Macros

- #define TLV_CUSTOM_COPY(descriptor, TLV_parsed_content, makro_currentTLVcontentSize)
- #define TLV_STRING_COPY(descriptor)

Functions

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

  This function has the main receiver logic and starts all other receiver 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.

4.8.1 Macro Definition Documentation

4.8.1.1 TLV_CUSTOM_COPY

```
#define TLV_CUSTOM_COPY(
              descriptor,
              TLV_parsed_content,
              makro_currentTLVcontentSize )
```

Value:

```
snprintf(parsedTLVs [numberParsedTLVs++], PARSED_TLVS_MAX_LENGTH, "%-10s%.*s",  \
        descriptor, (int) makro_currentTLVcontentSize, TLV_parsed_content); \
    break;
```

Referenced by evaluatelanbeacon().

4.8.1.2 TLV_STRING_COPY

```
#define TLV_STRING_COPY(
              descriptor )
```

Value:

```
TLV_CUSTOM_COPY(descriptor,  \
        (char*) &my_received_lan_beacon_frame->lan_beacon_ReceivedPayload[currentPayloadByte+6], \
        currentTLVsize-4);
```

Referenced by evaluatelanbeacon().

4.8.2 Function Documentation

4.8.2.1 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.

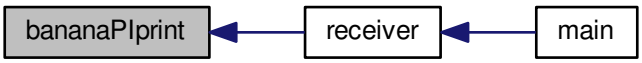
Parameters

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

References _, receiver_information::current_lan_beacon_pdu_for_printing, DESCRIPTOR_WIDTH, receiver_information::number_of_currently_received_frames, PARSED_TLVS_MAX_NUMBER, received_lan_beacon_frame::parsedBeaconContents, receiver_information::pointers_to_received_frames, receiver_information::scroll_speed, and received_lan_beacon_frame::times_left_to_display.

Referenced by receiver().

Here is the caller graph for this function:



4.8.2.2 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

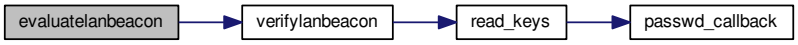
Returns

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

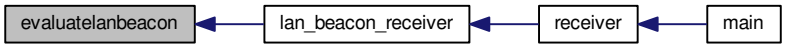
References _, received_lan_beacon_frame::challenge, DESCRIPTOR_COMBINED_STRING, DESCRIPTOR_CUSTOM, DESCRIPTOR_DHCP, DESCRIPTOR_EMAIL, DESCRIPTOR_IPV4, DESCRIPTOR_IPV6, DESCRIPTOR_NAME, DESCRIPTOR_ROUTER, DESCRIPTOR_SIGNATURE, DESCRIPTOR_VLAN_ID, received_lan_beacon_frame::lan_beacon_ReceivedPayload, PARSED_TLVS_MAX_LENGTH, PARSED_TLVS_MAX_NUMBER, received_lan_beacon_frame::payloadSize, SUBTYPE_COMBINED_STRING, SUBTYPE_CUSTOM, SUBTYPE_DHCP, SUBTYPE_EMAIL, SUBTYPE_IPV4, SUBTYPE_IPV6, SUBTYPE_NAME, SUBTYPE_ROUTER, SUBTYPE_SIGNATURE, SUBTYPE_VLAN_ID, received_lan_beacon_frame::successfullyAuthenticated, TLV_CUSTOM_COPY, TLV_STRING_COPY, and verifylanbeacon().

Referenced by lan_beacon_receiver().

Here is the call graph for this function:



Here is the caller graph for this function:



4.8.2.3 receiver()

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

This function has the main receiver logic and starts all other receiver functions.

Parameters

argc	Number of command line arguments.
argv	Contents of command line arguments.

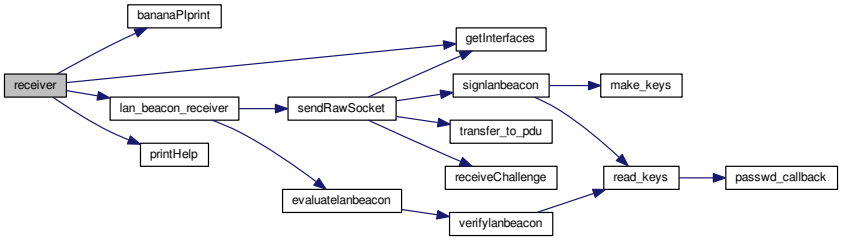
Returns

Error or failure code

References _, receiver_information::authenticated_mode, bananaPIprint(), receiver_information::current_lan_↩ beacon_pdu_for_printing, DEFAULT_SCROLLSPEED, getInterfaces(), KEY_PATHLENGTH_MAX, LAN_BEA↩ CON_ETHER_TYPE, lan_beacon_receiver(), receiver_information::lanbeacon_keys, receiver_information::my↩ _receiver_interfaces, receiver_information::number_of_currently_received_frames, PARSED_TLVS_MAX_NUM↩ BER, received_lan_beacon_frame::parsedBeaconContents, open_ssl_keys::path_To_Verifying_Key, receiver_↩ information::pointers_to_received_frames, printHelp(), PUBLIC_KEY_STANDARD_PATH, REC_SOCKET, REC↩ EIVER_MODE, receiver_information::scroll_speed, and open_ssl_keys::sender_or_receiver_mode.

Referenced by main().

Here is the call graph for this function:



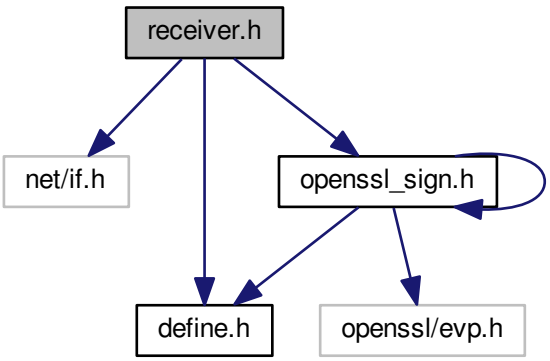
Here is the caller graph for this function:

4.9 receiver.h File Reference

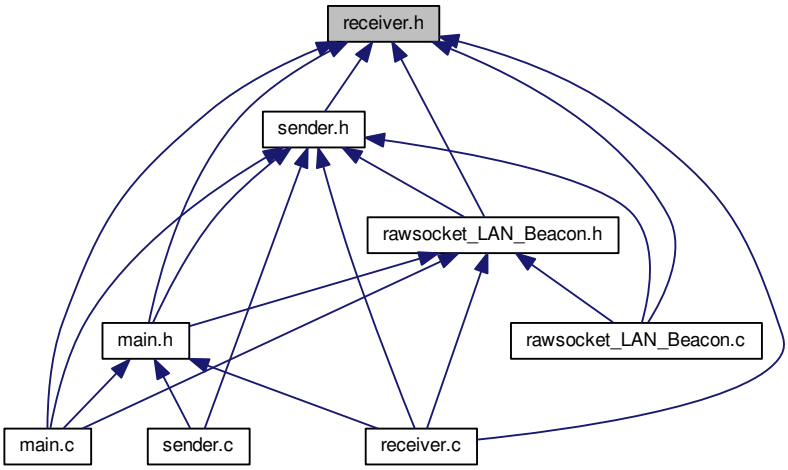
Receiver-specific functions and structures.

```
#include <net/if.h>
#include "define.h"
#include "openssl_sign.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

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

  This function has the main receiver logic and starts all other receiver 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.

4.9.1 Detailed Description

Receiver-specific functions and structures.

Author

Dominik Bitzer

Date

2017

4.9.2 Function Documentation

4.9.2.1 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.

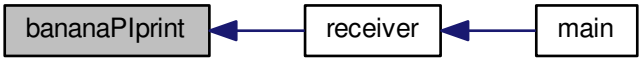
Parameters

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

References _, receiver_information::current_lan_beacon_pdu_for_printing, DESCRIPTOR_WIDTH, receiver_information::number_of_currently_received_frames, PARSED_TLVS_MAX_NUMBER, received_lan_beacon_frame::parsedBeaconContents, receiver_information::pointers_to_received_frames, receiver_information::scroll_speed, and received_lan_beacon_frame::times_left_to_display.

Referenced by receiver().

Here is the caller graph for this function:



4.9.2.2 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

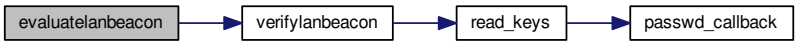
Returns

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

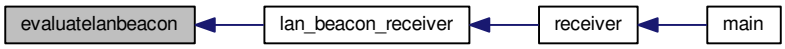
References _, received_lan_beacon_frame::challenge, DESCRIPTOR_COMBINED_STRING, DESCRIPTOR_CUSTOM, DESCRIPTOR_DHCP, DESCRIPTOR_EMAIL, DESCRIPTOR_IPV4, DESCRIPTOR_IPV6, DESCRIPTOR_NAME, DESCRIPTOR_ROUTER, DESCRIPTOR_SIGNATURE, DESCRIPTOR_VLAN_ID, received_lan_beacon_frame::lan_beacon_ReceivedPayload, PARSED_TLVS_MAX_LENGTH, PARSED_TLVS_MAX_NUMBER, received_lan_beacon_frame::payloadSize, SUBTYPE_COMBINED_STRING, SUBTYPE_CUSTOM, SUBTYPE_DHCP, SUBTYPE_EMAIL, SUBTYPE_IPV4, SUBTYPE_IPV6, SUBTYPE_NAME, SUBTYPE_ROUTER, SUBTYPE_SIGNATURE, SUBTYPE_VLAN_ID, received_lan_beacon_frame::successfullyAuthenticated, TLV_CUSTOM_COPY, TLV_STRING_COPY, and verifylanbeacon().

Referenced by lan_beacon_receiver().

Here is the call graph for this function:



Here is the caller graph for this function:



4.9.2.3 receiver()

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

This function has the main receiver logic and starts all other receiver functions.

Parameters

argc	Number of command line arguments.
argv	Contents of command line arguments.

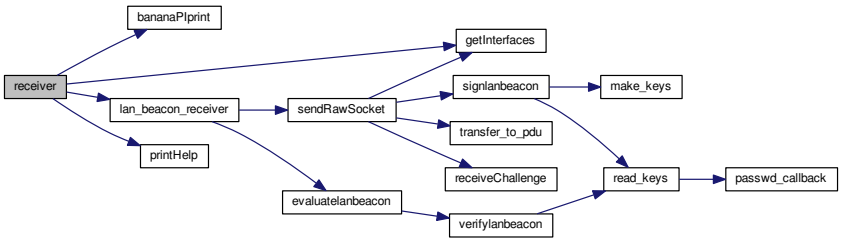
Returns

Error or failure code

References _, receiver_information::authenticated_mode, bananaPIprint(), receiver_information::current_lan_↩ beacon_pdu_for_printing, DEFAULT_SCROLLSPEED, getInterfaces(), KEY_PATHLENGTH_MAX, LAN_BEA↩ CON_ETHER_TYPE, lan_beacon_receiver(), receiver_information::lanbeacon_keys, receiver_information::my↩ _receiver_interfaces, receiver_information::number_of_currently_received_frames, PARSED_TLVS_MAX_NUM↩ BER, received_lan_beacon_frame::parsedBeaconContents, open_ssl_keys::path_To_Verifying_Key, receiver_↩ information::pointers_to_received_frames, printHelp(), PUBLIC_KEY_STANDARD_PATH, REC_SOCKET, REC↩ EIVER_MODE, receiver_information::scroll_speed, and open_ssl_keys::sender_or_receiver_mode.

Referenced by main().

Here is the call graph for this function:

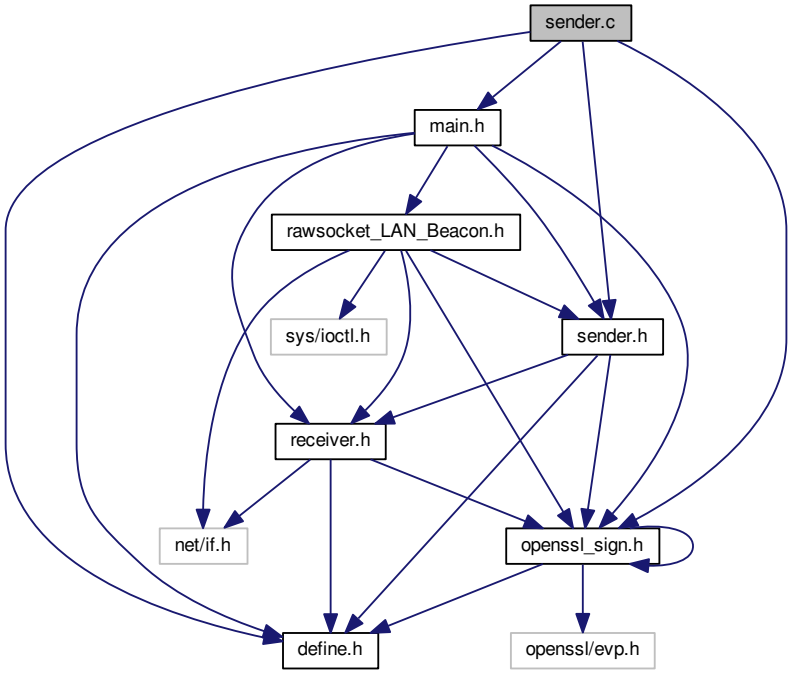


Here is the caller graph for this function:

4.10 sender.c File Reference

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

Include dependency graph for sender.c:



Functions

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

  This function has the main receiver logic and starts all other receiver functions.
- char * lanbeacon_creator (int *argc, char **argv, struct sender_information *my_sender_information)

  Creates a LAN-Beacon PDU from the command line arguments.
- void transfer_to_pdu_and_string (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 transfer_to_pdu (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 transfer_to_string (char *TLVdescription, char **combinedString, char *TLVcontents)

  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.

4.10.1 Function Documentation

4.10.1.1 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

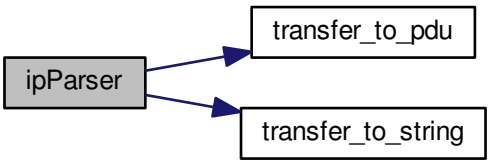
Parameters

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

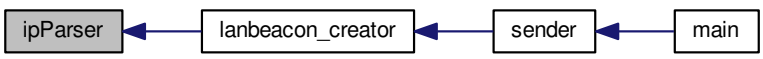
References _, DESCRIPTOR_IPV4, DESCRIPTOR_IPV6, SUBTYPE_IPV4, SUBTYPE_IPV6, transfer_to_pdu(), and transfer_to_string().

Referenced by lanbeacon_creator().

Here is the call graph for this function:



Here is the caller graph for this function:



4.10.1.2 lanbeacon_creator()

```
char* lanbeacon_creator (
            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 lanbeacon_creator()

Parameters

argc	Number of command line arguments.
argv	Contents of command line arguments.

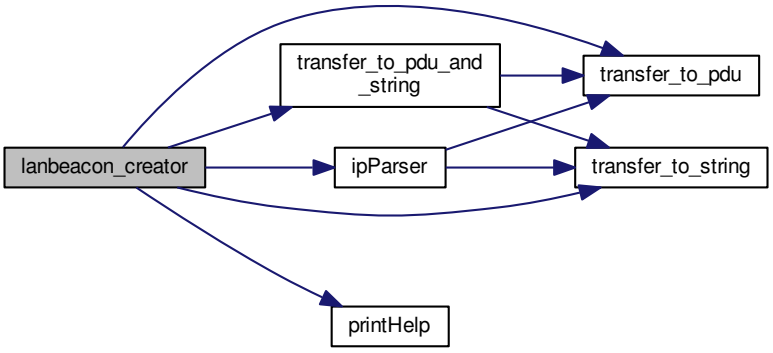
Returns

Returns an array, that contains the payload of a lanBeacon_PDU

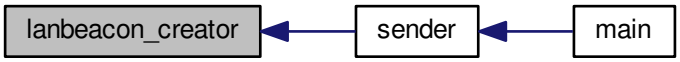
References _, DESCRIPTOR_CUSTOM, DESCRIPTOR_DHCP, DESCRIPTOR_EMAIL, DESCRIPTOR_NAME, DESCRIPTOR_ROUTER, DESCRIPTOR_VLAN_ID, open_ssl_keys::generate_keys, sender_information::interface_to_send_on, ipParser(), KEY_PATHLENGTH_MAX, sender_information::lan_beacon_pdu_len, sender_information::lanbeacon_keys, open_ssl_keys::path_To_Signing_Key, open_ssl_keys::path_To_Verifying_Key, open_ssl_keys::pcszPassphrase, printHelp(), sender_information::send_frequency, SUBTYPE_COMBINED_STRING, SUBTYPE_CUSTOM, SUBTYPE_DHCP, SUBTYPE_EMAIL, SUBTYPE_NAME, SUBTYPE_ROUTER, SUBTYPE_VLAN_ID, transfer_to_pdu(), transfer_to_pdu_and_string(), and transfer_to_string().

Referenced by sender().

Here is the call graph for this function:



Here is the caller graph for this function:



4.10.1.3 sender()

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

This function has the main receiver logic and starts all other receiver functions.

Parameters

argc	Number of command line arguments.
argv	Contents of command line arguments.

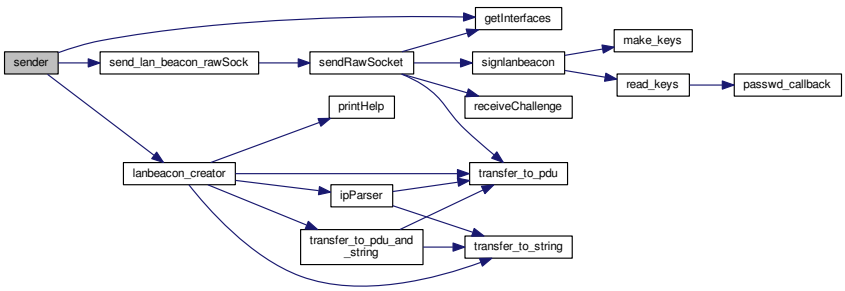
Returns

Error or failure code

References CHALLENGE_ETHTYPE, getInterfaces(), sender_information::interface_to_send_on, LAN_BEAC↩ ON_SEND_FREQUENCY, lanbeacon_creator(), sender_information::lanBeacon_PDU, sender_information::my_↩ challenge_receiver_interfaces, PRIVATE_KEY_STANDARD_PATH, PUBLIC_KEY_STANDARD_PATH, REC_↩ SOCKET, send_lan_beacon_rawSock(), and SENDER_MODE.

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:

4.10.1.4 transfer_to_pdu()

```
void transfer_to_pdu (
            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.

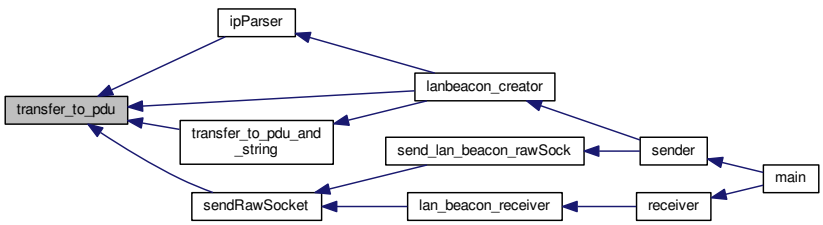
Parameters

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

References _.

Referenced by ipParser(), lanbeacon_creator(), sendRawSocket(), and transfer_to_pdu_and_string().

Here is the caller graph for this function:



4.10.1.5 transfer_to_pdu_and_string()

```
void transfer_to_pdu_and_string (
             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.

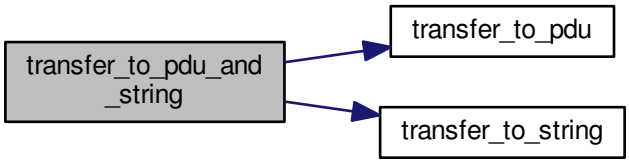
Parameters

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

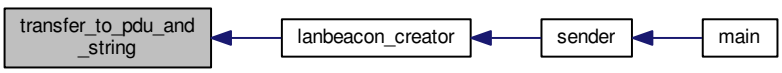
References transfer_to_pdu(), and transfer_to_string().

Referenced by lanbeacon_creator().

Here is the call graph for this function:



Here is the caller graph for this function:



4.10.1.6 transfer_to_string()

```
void transfer_to_string (
            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

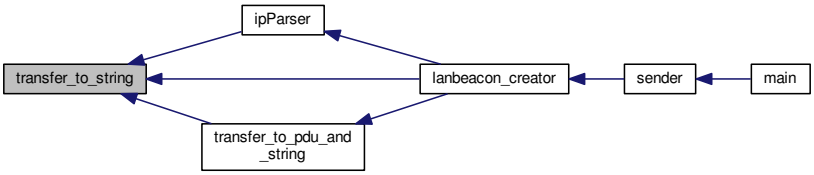
Parameters

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

References _.

Referenced by ipParser(), lanbeacon_creator(), and transfer_to_pdu_and_string().

Here is the caller graph for this function:

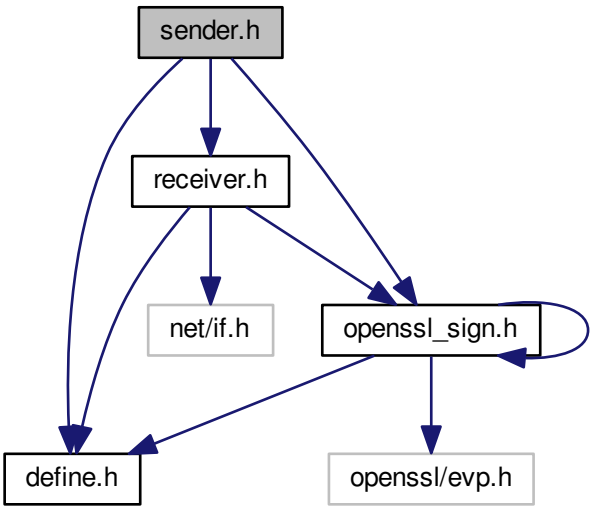


4.11 sender.h File Reference

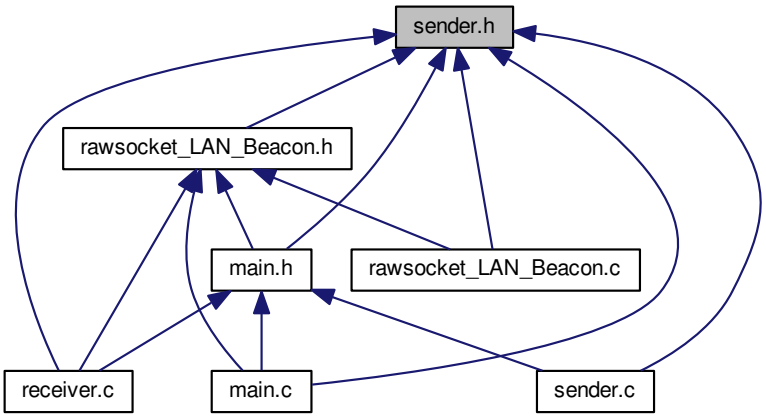
Sender-specific functions and structures.

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

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

  This function has the main receiver logic and starts all other receiver functions.
- char * lanbeacon_creator (int *argc, char **argv, struct sender_information *my_sender_information)

  Creates a LAN-Beacon PDU from the command line arguments.
- void transfer_to_pdu_and_string (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 transfer_to_pdu (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 transfer_to_string (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.

4.11.1 Detailed Description

Sender-specific functions and structures.

Author

Dominik Bitzer

Date

2017

4.11.2 Function Documentation

4.11.2.1 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

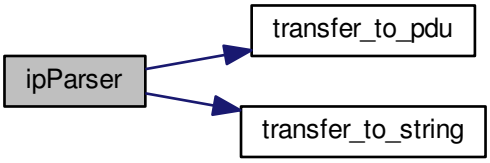
Parameters

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

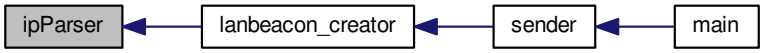
References _, DESCRIPTOR_IPV4, DESCRIPTOR_IPV6, SUBTYPE_IPV4, SUBTYPE_IPV6, transfer_to_pdu(), and transfer_to_string().

Referenced by lanbeacon_creator().

Here is the call graph for this function:



Here is the caller graph for this function:



4.11.2.2 lanbeacon_creator()

```
char* lanbeacon_creator (
            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 lanbeacon_creator()

Parameters

argc	Number of command line arguments.
argv	Contents of command line arguments.

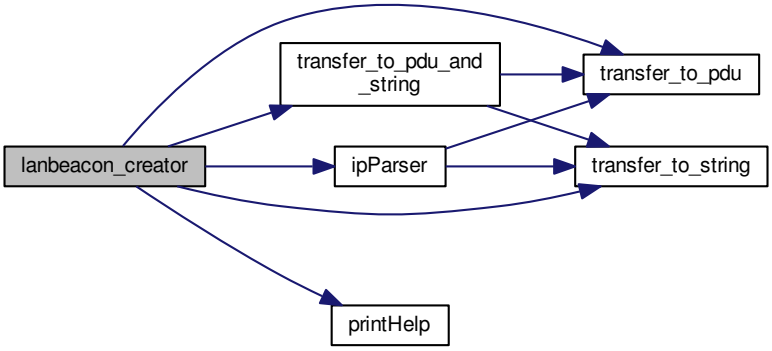
Returns

Returns an array, that contains the payload of a lanBeacon_PDU

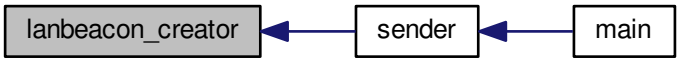
References _, DESCRIPTOR_CUSTOM, DESCRIPTOR_DHCP, DESCRIPTOR_EMAIL, DESCRIPTOR_NAME, DESCRIPTOR_ROUTER, DESCRIPTOR_VLAN_ID, open_ssl_keys::generate_keys, sender_information::interface_to_send_on, ipParser(), KEY_PATHLENGTH_MAX, sender_information::lan_beacon_pdu_len, sender_information::lanbeacon_keys, open_ssl_keys::path_To_Signing_Key, open_ssl_keys::path_To_Verifying_Key, open_ssl_keys::pcszPassphrase, printHelp(), sender_information::send_frequency, SUBTYPE_COMBINED_STRING, SUBTYPE_CUSTOM, SUBTYPE_DHCP, SUBTYPE_EMAIL, SUBTYPE_NAME, SUBTYPE_ROUTER, SUBTYPE_VLAN_ID, transfer_to_pdu(), transfer_to_pdu_and_string(), and transfer_to_string().

Referenced by sender().

Here is the call graph for this function:



Here is the caller graph for this function:



4.11.2.3 sender()

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

This function has the main receiver logic and starts all other receiver functions.

Parameters

argc	Number of command line arguments.
argv	Contents of command line arguments.

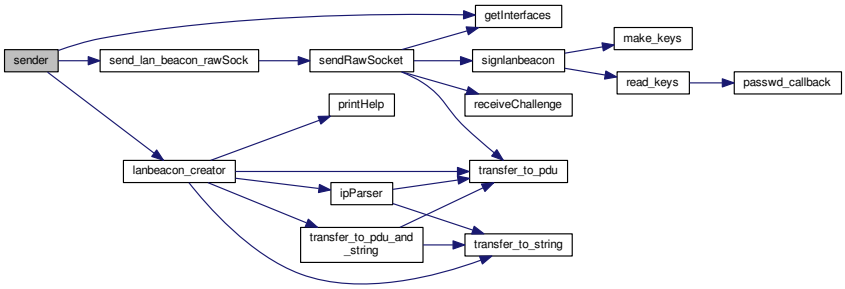
Returns

Error or failure code

References CHALLENGE_ETHTYPE, getInterfaces(), sender_information::interface_to_send_on, LAN_BEAC↩ ON_SEND_FREQUENCY, lanbeacon_creator(), sender_information::lanBeacon_PDU, sender_information::my_↩ challenge_receiver_interfaces, PRIVATE_KEY_STANDARD_PATH, PUBLIC_KEY_STANDARD_PATH, REC_↩ SOCKET, send_lan_beacon_rawSock(), and SENDER_MODE.

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:

4.11.2.4 transfer_to_pdu()

```
void transfer_to_pdu (
            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.

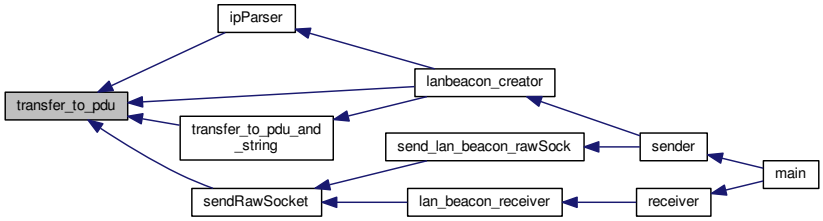
Parameters

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

References _.

Referenced by ipParser(), lanbeacon_creator(), sendRawSocket(), and transfer_to_pdu_and_string().

Here is the caller graph for this function:



4.11.2.5 transfer_to_pdu_and_string()

```
void transfer_to_pdu_and_string (
            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.

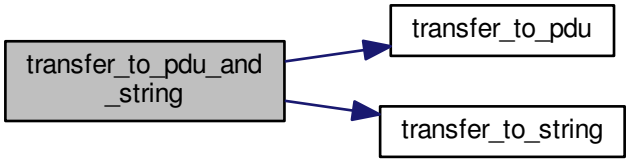
Parameters

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

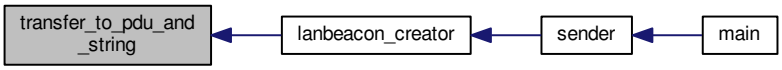
References transfer_to_pdu(), and transfer_to_string().

Referenced by lanbeacon_creator().

Here is the call graph for this function:



Here is the caller graph for this function:



4.11.2.6 transfer_to_string()

```
void transfer_to_string (
            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

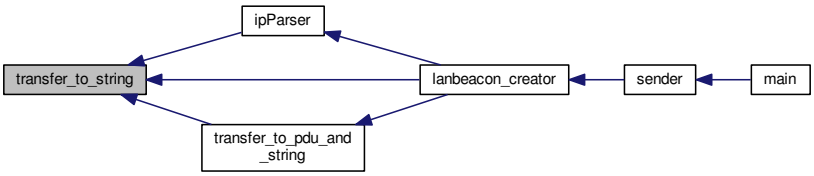
Parameters

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

References _.

Referenced by ipParser(), lanbeacon_creator(), and transfer_to_pdu_and_string().

Here is the caller graph for this function:



Index