Hackable ESP device

Generated by Doxygen 1.9.3

Chapter 1

Hackable ESP8266 device

Firmware for ESP8266 based device (D1 Mini board) with designed vulnerabilities to practice ethical hacking. The software is tested on the following boards:

- `D1 Mini`

1.1 Getting Started

These instructions will get you a copy of the project up and running on your D1 Mini (or other ESP8266 based boards) for development or hacking purposes.

1.1.1 Prerequisites

The software is written, compiled and uploaded using the `Arduino IDE`. Platform.io and Visual Studio Code can be used as well. Use the script to convert the project to a Platform.io.

1.1.2 Dependencies

- ESP for Arduino IDE
- ESP Async WebServer V1.2.3
- Wifimanager V0.16.0

1.1.3 Installing

General Install

1. Install the `driver` for the esp8266.
2. Clone the repository.

There are multiple ways to upload the program files to the board. The two ways listed here are using Arduino IDE and Platformio on Visual Studio Code.

1.1.3.1 <b>Option 1: Arduino IDE</b>

1. Install the `Arduino IDE`
2. `Add the esp8266 libraries to Arduino IDE`.
3. Follow `this` tutorial about the SPIFFS.
4. Navigate to the `hackableEspDevice` folder.
5. Open `hackableEspDevice.ino`.
6. Upload the files in the `data` folder (see the tutorial).
7. Upload the program to the device.
8. Connect to the `Configure Smartlight Wifi` AP to configure the wifi.

   ESP8266 Sketch Data Upload

   (a) The Arduino IDE won't have the option 'ESP8266 Sketch Data Upload'.

   (b) You can download it from this `link`.

   (c) The file should be unpacked at `<home_dir>/Arduino-<version>/tools/ESP8266↩ FS/tools/`.

   - If the directory `tools` does not exist you should create it. You have to create a new file named "tools" if it doesn't exist already inside of `Arduino file`.

1.1.3.2 <b>Option 2: Visual Studio Code + Platformio</b>

1. Install the `Platformio` plugin.
2. Prepare files for platformio.
   - Run the `toPlatformio.ps1` script and select the copy or symbolic option.
     - Symbolic changes the original ideal for editing the files.
     - Copy simply copies the files to a new location for platformio files.
   - Run the `toPlatformio.ps1` script and select fix.
   - Or prepare the files manually see manual prep platformio.
3. Open visual studio code in the `HackableEspDevicePlatformio` directory.
4. In visual studio code open the project in the platformio addon. `(Platoformio > Projects > open HackableEspDevicePlatformio).`
5. Upload the program `(project tasks > General> Upload).`
6. Upload the filesystem Image `(Project tasks > Platform > Upload filesystem Image).`
7. Done. The device should now be ready for use.

1.1.4 Manual Platformio Prep

1. Create the correct hierarchy.

```
|HackableEspDevicePlatformio\
|--- platformio.ini
|--- src\
|--- src\main.cpp\
|--- data\
```

1. The src dir needs to contain all the files from the `hackableEspDevice` directory except the data directory.
2. Rename the `hackableEspDevice.ino` to `main.cpp`.
3. In `main.cpp` add a reference to all functions in main e.g.
```
void setup();
void setup();
void initializeHostname();
void connectWifi();
void initializeServer();
void loop();
String getContentType(String filename);
void handleFileRequest(String path, uint8_t permissionLevel);
void handleFileUpload();
void handleFileDownload();
```
4. Move the `platformio.ini` file from the root dir to the `hackableEspDevicePlatformio` dir.
5. Copy all files from `hackableEspDevice\data` to `hackableEspDevicePlatformio\data`.

1.1.5 <b>Running</b>

1.1.5.1 <b>Wifi Manager First Boot</b>

1. Start up the device.
2. Connect to the `Configure Smartlight Wifi` via a mobile device.
3. Go to the IP address listed in the serial monitor. Most of the time this is `http://192.168.4.1`.
4. Follow the steps on the website to configure a wifi connection.
5. The device should now restart, connect to the selected wifi network and be ready for use.

1.1.6 Customization of Hackable ESP (Contains spoilers) (Look in raw version of readme.md)

1.2 Hardware

- 1x D1 Mini Board
- 1x USB to USB-mini cable
- 1x ESP8266 casing

1.3 Questions or Feedback?

There is technical documentation available if you want to contribute to this project. There is a user manual as well, contact us for information. You can open an issue if you have questions or feedback for this repository.

1.4 Authors

- **Luke de Munk** - *Head author* - `LinkedIn`
- **Thijs Takken** - *Head author* - `LinkedIn`
- **Christina Kostine** - *Head author* - `LinkedIn`
- **Twenne Elffers** - *Head author* - `LinkedIn`

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 AES Class Reference

```
#include <AES.h>
```

Public Member Functions

- AES ()

  AES constructor.
- byte set_key (byte key[ ], int keylen)
- void clean ()
- void copy_n_bytes (byte ∗AESt, byte ∗src, byte n)
- byte encrypt (byte plain[N_BLOCK], byte cipher[N_BLOCK])
- byte cbc_encrypt (byte ∗plain, byte ∗cipher, int n_block, byte iv[N_BLOCK])
- byte cbc_encrypt (byte ∗plain, byte ∗cipher, int n_block)
- byte decrypt (byte cipher[N_BLOCK], byte plain[N_BLOCK])
- byte cbc_decrypt (byte ∗cipher, byte ∗plain, int n_block, byte iv[N_BLOCK])
- byte cbc_decrypt (byte ∗cipher, byte ∗plain, int n_block)
- void set_IV (unsigned long long int IVCl)
- void iv_inc ()
- int get_size ()
- void set_size (int sizel)
- int get_pad ()
- void get_IV (byte ∗out)
- void calc_size_n_pad (int p_size)
- void padPlaintext (void ∗in, byte ∗out)
- bool CheckPad (byte ∗in, int size)
- void printArray (byte output[ ], bool p_pad=true)
- void printArray (byte output[ ], int sizel)
- void do_aes_encrypt (byte ∗plain, int size_p, byte ∗cipher, byte ∗key, int bits, byte ivl[N_BLOCK])
- void do_aes_encrypt (byte ∗plain, int size_p, byte ∗cipher, byte ∗key, int bits)
- void do_aes_decrypt (byte ∗cipher, int size_c, byte ∗plain, byte ∗key, int bits, byte ivl[N_BLOCK])
- void do_aes_decrypt (byte ∗cipher, int size_c, byte ∗plain, byte ∗key, int bits)

4.1.1 Detailed Description

Definition at line 39 of file AES.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 AES()

```
AES::AES ( )
```

AES constructor.

This function initialized an instance of AES.

Definition at line 231 of file AES.cpp.

4.1.3 Member Function Documentation

4.1.3.1 calc_size_n_pad()

```
void AES::calc_size_n_pad (
             int p_size )
```

Calculates the size of the plaintext and the padding.

Calculates the size of theplaintext with the padding and the size of the padding needed. Moreover it stores them in their class variables.

Parameters

p_size	the size of the byte array ex sizeof(plaintext)

Definition at line 492 of file AES.cpp.

4.1.3.2 cbc_decrypt() [1/2]

```
byte AES::cbc_decrypt (
             byte * cipher,
             byte * plain,
             int n_block )
```

CBC decrypt a number of blocks (input and return an IV)

Parameters

∗cipher	Pointer, points to the ciphertext that will be created.
∗plain	Pointer, points to the plaintex.
n_block	integer, indicated the number of blocks to be ciphered. @Return 0 if SUCCESS or -1 if FAILURE

Definition at line 432 of file AES.cpp.

4.1.3.3 cbc_decrypt() [2/2]

```
byte AES::cbc_decrypt (
            byte * cipher,
            byte * plain,
            int n_block,
            byte iv[N_BLOCK] )
```

CBC decrypt a number of blocks (input and return an IV)

Parameters

∗cipher	Pointer, points to the ciphertext that will be created.
∗plain	Pointer, points to the plaintex.
n_block	integer, indicated the number of blocks to be ciphered.
iv[N_BLOCK]	byte Array that holds the IV (initialization vector). @Return 0 if SUCCESS or -1 if FAILURE

Definition at line 414 of file AES.cpp.

4.1.3.4 cbc_encrypt() [1/2]

```
byte AES::cbc_encrypt (
            byte * plain,
            byte * cipher,
            int n_block )
```

CBC encrypt a number of blocks (input and return an IV).

Parameters

∗plain	Pointer, points to the plaintex.
∗cipher	Pointer, points to the ciphertext that will be created.
n_block	integer, indicated the number of blocks to be ciphered. @Return 0 if SUCCESS or -1 if FAILURE

Definition at line 375 of file AES.cpp.

4.1.3.5 cbc_encrypt() [2/2]

```
byte AES::cbc_encrypt (
            byte * plain,
            byte * cipher,
            int n_block,
            byte iv[N_BLOCK] )
```

CBC encrypt a number of blocks (input and return an IV).

Parameters

∗plain	Pointer, points to the plaintex.
∗cipher	Pointer, points to the ciphertext that will be created.
n_block	integer, indicated the number of blocks to be ciphered.
iv[N_BLOCK]	byte Array that holds the IV (initialization vector). @Return 0 if SUCCESS or -1 if FAILURE

Definition at line 359 of file AES.cpp.

4.1.3.6 CheckPad()

```
bool AES::CheckPad (
            byte * in,
            int size )
```

Check the if the padding is correct.

This functions checks the padding of the plaintext.

Parameters

in	the string of the plaintext in a byte array
size	the size of the string

Returns

true if correct / false if not

Definition at line 509 of file AES.cpp.

4.1.3.7 clean()

```
void AES::clean ( )
```

clean up subkeys after use.

Definition at line 311 of file AES.cpp.

4.1.3.8 copy_n_bytes()

```
void AES::copy_n_bytes (
             byte * AESt,
             byte * src,
             byte n )
```

copying and xoring utilities.

Parameters

*AESt	byte pointer of the AEStination array.
*src	byte pointer of the source array.
n	byte, indicating the sizeof the bytes to be copied.

Note

this is an alternative for memcpy(void *s1,const void *s2, site_t n), i have not updated the function in the implementation yet, but it is considered a future plan.

Definition at line 320 of file AES.cpp.

4.1.3.9 decrypt()

```
byte AES::decrypt (
             byte cipher[N_BLOCK],
             byte plain[N_BLOCK] )
```

Decrypt a single block of 16 bytes

Parameters

cipher[N_BLOCK]	Array of the ciphertext.
plain[N_BLOCK]	Array of the plaintext.

Note

The N_BLOCK is defined in AES_config.h as,

```
#define N_ROW                   4
     #define N_COL                   4
     #define N_BLOCK    (N_ROW * N_COL)
```

Changed to that will change the Block_size. @Return 0 if SUCCESS or -1 if FAILURE

Definition at line 391 of file AES.cpp.

4.1.3.10 do_aes_decrypt() [1/2]

```
void AES::do_aes_decrypt (
             byte * cipher,
             int size_c,
             byte * plain,
             byte * key,
             int bits )
```

User friendly implementation of AES-CBC decryption.

Parameters

∗cipher	pointer to the ciphertext
size←↩ _c	size of the ciphertext
∗plain	pointer to the plaintext
∗key	pointer to the key that will be used.
bits	bits of the encryption/decrpytion

Note

The key will be stored in class variable.

Definition at line 585 of file AES.cpp.

4.1.3.11 do_aes_decrypt() [2/2]

```
void AES::do_aes_decrypt (
             byte * cipher,
             int size_c,
             byte * plain,
             byte * key,
             int bits,
             byte ivl[N_BLOCK] )
```

User friendly implementation of AES-CBC decryption.

Parameters

∗cipher	pointer to the ciphertext
size_c	size of the ciphertext
∗plain	pointer to the plaintext
∗key	pointer to the key that will be used.
bits	bits of the encryption/decrpytion
ivl[N_BLOCK]	the initialization vector IV that will be used for decryption.

Note

The key will be stored in class variable.

Definition at line 576 of file AES.cpp.

4.1.3.12 do_aes_encrypt() [1/2]

```
void AES::do_aes_encrypt (
             byte * plain,
             int size_p,
             byte * cipher,
             byte * key,
             int bits )
```

User friendly implementation of AES-CBC encryption.

Parameters

∗plain	pointer to the plaintext
size↩ _p	size of the plaintext
∗cipher	pointer to the ciphertext
∗key	pointer to the key that will be used.
bits	bits of the encryption/decrpytion

Note

The key will be stored in class variable.

Definition at line 565 of file AES.cpp.

4.1.3.13 do_aes_encrypt() [2/2]

```
void AES::do_aes_encrypt (
             byte * plain,
             int size_p,
             byte * cipher,
             byte * key,
             int bits,
             byte ivl[N_BLOCK] )
```

User friendly implementation of AES-CBC encryption.

Parameters

∗plain	pointer to the plaintext
size_p	size of the plaintext
∗cipher	pointer to the ciphertext
∗key	pointer to the key that will be used.
bits	bits of the encryption/decrpytion
ivl[N_BLOCK]	the initialization vector IV that will be used for encryption.

Note

The key will be stored in class variable.

Definition at line 554 of file AES.cpp.

4.1.3.14 encrypt()

```
byte AES::encrypt (
            byte plain[N_BLOCK],
            byte cipher[N_BLOCK] )
```

Encrypt a single block of 16 bytes .

Parameters

plain[N_BLOCK]	Array of the plaintext.
cipher[N_BLOCK]	Array of the ciphertext.

Note

The N_BLOCK is defined in AES_config.h as,

```
#define N_ROW                   4
     #define N_COL                   4
     #define N_BLOCK   (N_ROW * N_COL)
```

Changed to that will change the Block_size. @Return 0 if SUCCESS or -1 if FAILURE

Definition at line 336 of file AES.cpp.

4.1.3.15 get_IV()

```
void AES::get_IV (
            byte * out )
```

Getter method for IV

This function return the IV

Parameters

out	byte pointer that gets the IV.

Returns

none, the IV is writed to the out pointer.

Definition at line 485 of file AES.cpp.

4.1.3.16 get_pad()

```
int AES::get_pad ( )
```

Definition at line 472 of file AES.cpp.

4.1.3.17 get_size()

```
int AES::get_size ( )
```

Getter method for size

This function return the size

Returns

an integer, that is the size of the of the padded plaintext, thus, the size of the ciphertext.

Definition at line 466 of file AES.cpp.

4.1.3.18 iv_inc()

```
void AES::iv_inc ( )
```

increase the iv (initialization vector) and IVC (IV counter) by 1

This function increased the VI by one step in order to have a different IV each time

Definition at line 458 of file AES.cpp.

4.1.3.19 padPlaintext()

```
void AES::padPlaintext (
             void * in,
             byte * out )
```

Pads the plaintext

This function pads the plaintext and returns an char array with the plaintext and the padding in order for the plaintext to be compatible with 16bit size blocks required by AES

Parameters

in	the string of the plaintext in a byte array
out	The string of the out array.

Returns

no return, The padded plaintext is stored in the out pointer.

Definition at line 499 of file AES.cpp.

4.1.3.20 printArray() [1/2]

```
void AES::printArray (
            byte output[],
            bool p_pad = true )
```

Prints the array given.

This function prints the given array and pad, It is mainlly used for debugging purpuses or to output the string.

Parameters

output[ ]	the string of the text in a byte array
p_pad	optional, used to print with out the padding characters

Definition at line 525 of file AES.cpp.

4.1.3.21 printArray() [2/2]

```
void AES::printArray (
            byte output[],
            int sizel )
```

Prints the array given.

This function prints the given array in Hexadecimal.

Parameters

output[ ]	the string of the text in a byte array
sizel	the size of the array.

Definition at line 542 of file AES.cpp.

4.1.3.22 set_IV()

```
void AES::set_IV (
            unsigned long long int IVCl )
```

Sets IV (initialization vector) and IVC (IV counter). This function changes the ivc and iv variables needed for AES.

Parameters

IVCl	int or hex value of iv , ex. 0x0000000000000001

Note

example:

```
unsigned long long int my_iv = 01234567;
```

Definition at line 450 of file AES.cpp.

4.1.3.23 set_key()

```
byte AES::set_key (
            byte key[],
            int keylen )
```

Set the cipher key for the pre-keyed version.

Parameters

key[]	pointer to the key string.
keylen	Integer that indicates the length of the key.

Note

NOTE: If the length_type used for the key length is an unsigned 8-bit character, a key length of 256 bits must be entered as a length in bytes (valid inputs are hence 128, 192, 16, 24 and 32).

Definition at line 255 of file AES.cpp.

4.1.3.24 set_size()

```
void AES::set_size (
            int sizel )
```

Setter method for size

This function sets the size of the plaintext+pad

Definition at line 478 of file AES.cpp.

The documentation for this class was generated from the following files:

- hackableEspDevice/AES.h
- hackableEspDevice/AES.cpp

4.2 BufferOverflow Class Reference

```
#include <BufferOverflow.h>
```

Public Member Functions

- BufferOverflow ()

  Constructor.
- void ls ()

  Prints the fake list of files.
- void vi ()

  Prints the vulnerable testprogram.
- void objectDump ()

  Prints the disassembled code of the vulnerable testprogram.
- bool runCProgram (String arg)

  Simulates the vulnerable testprogram.

4.2.1 Detailed Description

Definition at line 21 of file BufferOverflow.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 BufferOverflow()

```
BufferOverflow::BufferOverflow ( )
```

Constructor.

Definition at line 17 of file BufferOverflow.cpp.

4.2.3 Member Function Documentation

4.2.3.1 ls()

```
void BufferOverflow::ls ( )
```

Prints the fake list of files.

Definition at line 26 of file BufferOverflow.cpp.

4.2.3.2 objectDump()

```
void BufferOverflow::objectDump ( )
```

Prints the disassembled code of the vulnerable testprogram.

Definition at line 72 of file BufferOverflow.cpp.

4.2.3.3 runCProgram()

```
bool BufferOverflow::runCProgram (
             String arg )
```

Simulates the vulnerable testprogram.

Parameters

arg	Given argument

Returns

bool True if the buffer overflow attack is done correctly

Definition at line 129 of file BufferOverflow.cpp.

4.2.3.4 vi()

```
void BufferOverflow::vi ( )
```

Prints the vulnerable testprogram.

Definition at line 36 of file BufferOverflow.cpp.

The documentation for this class was generated from the following files:

- hackableEspDevice/BufferOverflow.h
- hackableEspDevice/BufferOverflow.cpp

4.3 CbcEncryptor Class Reference

```
#include <CbcEncryptor.h>
```

Public Member Functions

- CbcEncryptor ()

    Constructor.
- bool encryptFile (String filename)

    Encrypts the SPIFFS file if it exists.
- bool decryptFile (String filename)

    Decrypts the SPIFFS file if it exists.
- String encryptLine (String line)

    Encrypts a line of text.
- String decryptLine (String line)

    Decrypts a line of text.
- bool setKey (String key)

    Sets the encryption key.

4.3.1 Detailed Description

Definition at line 15 of file CbcEncryptor.h.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 CbcEncryptor()

```
CbcEncryptor::CbcEncryptor ( )
```

Constructor.

Definition at line 16 of file CbcEncryptor.cpp.

4.3.3 Member Function Documentation

4.3.3.1 decryptFile()

```
bool CbcEncryptor::decryptFile (
              String filename )
```

Decrypts the SPIFFS file if it exists.

Parameters

filename	Name of the file to decrypt

Returns

bool True if decryption is successfull

Definition at line 75 of file CbcEncryptor.cpp.

4.3.3.2 decryptLine()

```
String CbcEncryptor::decryptLine (
             String line )
```

Decrypts a line of text.

Parameters

line	Line to decrypt

Returns

String Decrypted string

Definition at line 168 of file CbcEncryptor.cpp.

4.3.3.3 encryptFile()

```
bool CbcEncryptor::encryptFile (
             String filename )
```

Encrypts the SPIFFS file if it exists.

Parameters

filename	Name of the file to encrypt

Returns

bool True if encryption is successfull

Definition at line 33 of file CbcEncryptor.cpp.

4.3.3.4 encryptLine()

```
String CbcEncryptor::encryptLine (
             String line )
```

Encrypts a line of text.

Parameters

line	Line to encrypt

Returns

String Encrypted string

Definition at line 137 of file CbcEncryptor.cpp.

4.3.3.5 setKey()

```
bool CbcEncryptor::setKey (
            String key )
```

Sets the encryption key.

Parameters

key	Encryption key

Returns

bool True if is successfull

Definition at line 115 of file CbcEncryptor.cpp.

The documentation for this class was generated from the following files:

- hackableEspDevice/CbcEncryptor.h
- hackableEspDevice/CbcEncryptor.cpp

4.4 SerialCommandExecuter Class Reference

```
#include <SerialCommandExecuter.h>
```

Public Member Functions

- SerialCommandExecuter ()

  Constructor.
- void executeCommand ()

  Reads the commands and sends them to the parser.
- void setUsers (String ∗users, uint8_t numUsers)

  Sets the users for user list.

4.4.1 Detailed Description

Definition at line 66 of file SerialCommandExecuter.h.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 SerialCommandExecuter()

```
SerialCommandExecuter::SerialCommandExecuter ( )
```

Constructor.

Definition at line 16 of file SerialCommandExecuter.cpp.

4.4.3 Member Function Documentation

4.4.3.1 executeCommand()

```
void SerialCommandExecuter::executeCommand ( )
```

Reads the commands and sends them to the parser.

Definition at line 40 of file SerialCommandExecuter.cpp.

4.4.3.2 setUsers()

```
void SerialCommandExecuter::setUsers (
             String * users,
             uint8_t numUsers )
```

Sets the users for user list.

Parameters

users	Array of the users
numUsers	Number of users

Definition at line 27 of file SerialCommandExecuter.cpp.

The documentation for this class was generated from the following files:

- hackableEspDevice/SerialCommandExecuter.h
- hackableEspDevice/SerialCommandExecuter.cpp

4.5 UserHandler Class Reference

```
#include <userHandler.h>
```

Public Member Functions

- UserHandler (ESP8266WebServer ∗server)

  Constructor.
- void updateUsers ()

  Updates the users from the config file in RAM.
- String ∗ getUsers ()

  Gets users.
- uint8_t getNumberOfUsers ()

  Gets number of users.
- bool checkPermission (uint8_t permissionLevel, ESP8266WebServer ∗server)

  Checks if user has permission.

4.5.1 Detailed Description

Definition at line 17 of file userHandler.h.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 UserHandler()

```
UserHandler::UserHandler (
            ESP8266WebServer ∗ server )
```

Constructor.

Parameters

server	Webserver object

Definition at line 17 of file userHandler.cpp.

4.5.3 Member Function Documentation

4.5.3.1 checkPermission()

```
bool UserHandler::checkPermission (
            uint8_t permissionLevel,
            ESP8266WebServer * server )
```

Checks if user has permission.

Parameters

permissionLevel	0 = not logged in, 1 = user, 2 = admin
server	Webserver object

Returns

bool True if user has permission

Definition at line 94 of file userHandler.cpp.

4.5.3.2 getNumberOfUsers()

```
uint8_t UserHandler::getNumberOfUsers ( )
```

Gets number of users.

Returns

Uint8_t Number of users

Definition at line 82 of file userHandler.cpp.

4.5.3.3 getUsers()

```
String * UserHandler::getUsers ( )
```

Gets users.

Returns

String* String array of users

Definition at line 72 of file userHandler.cpp.

4.5.3.4 updateUsers()

```
void UserHandler::updateUsers ( )
```

Updates the users from the config file in RAM.

Definition at line 26 of file userHandler.cpp.

The documentation for this class was generated from the following files:

- hackableEspDevice/userHandler.h
- hackableEspDevice/userHandler.cpp

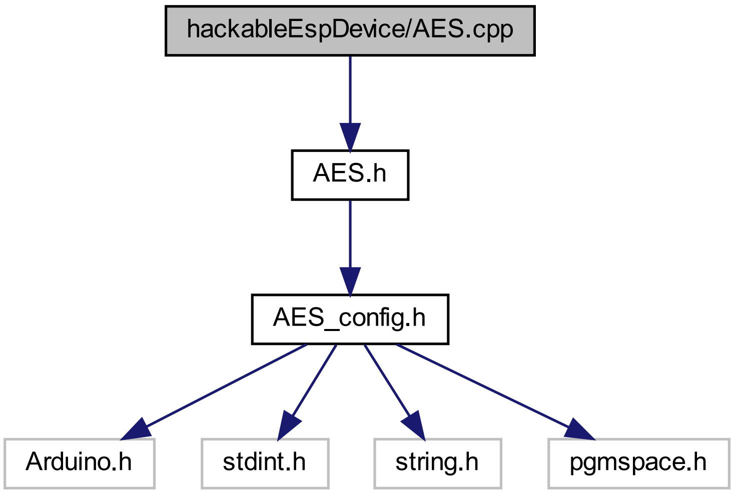
Chapter 5

File Documentation

5.1 hackableEspDevice/AES.cpp File Reference

```
#include "AES.h"
```

Include dependency graph for AES.cpp:



Macros

- #define WPOLY 0x011B
- #define DPOLY 0x008D
- #define f2(x) ((x) & 0x80 ? (x << 1) ^ WPOLY : x << 1)
- #define d2(x) (((x) >> 1) ^ ((x) & 1 ? DPOLY : 0))

5.1.1 Macro Definition Documentation

5.1.1.1 d2

```
#define d2(
             x ) (((x) >> 1) ^ ((x) & 1 ? DPOLY : 0))
```

Definition at line 110 of file AES.cpp.

5.1.1.2 DPOLY

```
#define DPOLY 0x008D
```

Definition at line 66 of file AES.cpp.

5.1.1.3 f2

```
#define f2(
             x ) ((x) & 0x80 ? (x << 1) ^ WPOLY : x << 1)
```

Definition at line 109 of file AES.cpp.

5.1.1.4 WPOLY

```
#define WPOLY 0x011B
```

Definition at line 65 of file AES.cpp.

5.2 AES.cpp

Go to the documentation of this file.

```
#include "AES.h"

/*
 ---------------------------------------------------------------------------
 Copyright (c) 1998-2008, Brian Gladman, Worcester, UK. All rights reserved.

 LICENSE TERMS

 The redistribution and use of this software (with or without changes)
 is allowed without the payment of fees or royalties provided that:

  1. source code distributions include the above copyright notice, this
     list of conditions and the following disclaimer;

  2. binary distributions include the above copyright notice, this list
     of conditions and the following disclaimer in their documentation;

  3. the name of the copyright holder is not used to endorse products
     built using this software without specific written permission.

 DISCLAIMER

 This software is provided 'as is' with no explicit or implied warranties
 in respect of its properties, including, but not limited to, correctness
 and/or fitness for purpose.
 ---------------------------------------------------------------------------
 Issue 09/09/2006

 This is an AES implementation that uses only 8-bit byte operations on the
 cipher state (there are options to use 32-bit types if available).

 The combination of mix columns and byte substitution used here is based on
 that developed by Karl Malbrain. His contribution is acknowledged.
 */

/* This version derived by Mark Tillotson 2012-01-23, tidied up, slimmed down
   and tailored to 8-bit microcontroller abilities and Arduino datatypes.

   The s-box and inverse s-box were retained as tables (0.5kB PROGMEM) but all
   the other transformations are coded to save table space.  Many efficiency
   improvments to the routines mix_sub_columns() and inv_mix_sub_columns()
   (mainly common sub-expression elimination).

   Only the routines with precalculated subkey schedule are retained (together
   with set_key() - this does however mean each AES object takes 240 bytes of
   RAM, alas)

   The CBC routines side-effect the iv argument (so that successive calls work
   together correctly).

   All the encryption and decryption routines work with plain == cipher for
   in-place encryption, note.

*/


/* functions for finite field multiplication in the AES Galois field    */

/* code was modified by george spanos <spaniakos@gmail.com>
 * 16/12/14
 */

// GF(2^8) stuff

#define WPOLY   0x011B
#define DPOLY   0x008D

static const byte s_fwd [0x100] PROGMEM =
{
  0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76,
  0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0,
  0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15,
  0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75,
  0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84,
  0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf,
  0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8,
  0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2,
  0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73,
  0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb,
  0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79,
  0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08,
  0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a,
  0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e,
  0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf,
  0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16,
} ;

static const byte s_inv [0x100] PROGMEM =
{
  0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38, 0xbf, 0x40, 0xa3, 0x9e, 0x81, 0xf3, 0xd7, 0xfb,
  0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87, 0x34, 0x8e, 0x43, 0x44, 0xc4, 0xde, 0xe9, 0xcb,
  0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d, 0xee, 0x4c, 0x95, 0x0b, 0x42, 0xfa, 0xc3, 0x4e,
  0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2, 0x76, 0x5b, 0xa2, 0x49, 0x6d, 0x8b, 0xd1, 0x25,
  0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xd4, 0xa4, 0x5c, 0xcc, 0x5d, 0x65, 0xb6, 0x92,
  0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda, 0x5e, 0x15, 0x46, 0x57, 0xa7, 0x8d, 0x9d, 0x84,
  0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a, 0xf7, 0xe4, 0x58, 0x05, 0xb8, 0xb3, 0x45, 0x06,
  0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02, 0xc1, 0xaf, 0xbd, 0x03, 0x01, 0x13, 0x8a, 0x6b,
  0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea, 0x97, 0xf2, 0xcf, 0xce, 0xf0, 0xb4, 0xe6, 0x73,
  0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85, 0xe2, 0xf9, 0x37, 0xe8, 0x1c, 0x75, 0xdf, 0x6e,
  0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89, 0x6f, 0xb7, 0x62, 0x0e, 0xaa, 0x18, 0xbe, 0x1b,
  0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20, 0x9a, 0xdb, 0xc0, 0xfe, 0x78, 0xcd, 0x5a, 0xf4,
  0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31, 0xb1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xec, 0x5f,
  0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d, 0x2d, 0xe5, 0x7a, 0x9f, 0x93, 0xc9, 0x9c, 0xef,
  0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0, 0xc8, 0xeb, 0xbb, 0x3c, 0x83, 0x53, 0x99, 0x61,
  0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26, 0xe1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0c, 0x7d,
} ;

// times 2 in the GF(2^8)
#define f2(x)   ((x) & 0x80 ? (x « 1) ^ WPOLY : x « 1)
#define d2(x)  (((x) » 1) ^ ((x) & 1 ? DPOLY : 0))

static byte s_box (byte x)
{
  //  return fwd_affine (pgm_read_byte (&inv [x])) ;
  return pgm_read_byte (& s_fwd [x]) ;
}

// Inverse Sbox
static byte is_box (byte x)
{
  // return pgm_read_byte (&inv [inv_affine (x)]) ;
  return pgm_read_byte (& s_inv [x]) ;
}


static void xor_block (byte * d, byte * s)
{
  for (byte i = 0 ; i < N_BLOCK ; i += 4)
    {
      *d++ ^= *s++ ;  // some unrolling
      *d++ ^= *s++ ;
      *d++ ^= *s++ ;
      *d++ ^= *s++ ;
    }
}

static void copy_and_key (byte * d, byte * s, byte * k)
{
  for (byte i = 0 ; i < N_BLOCK ; i += 4)
    {
      *d++ = *s++ ^ *k++ ;  // some unrolling
      *d++ = *s++ ^ *k++ ;
      *d++ = *s++ ^ *k++ ;
      *d++ = *s++ ^ *k++ ;
    }
}

// #define add_round_key(d, k) xor_block (d, k)

/* SUB ROW PHASE */

static void shift_sub_rows (byte st [N_BLOCK])
{
  st [0] = s_box (st [0]) ; st [4]  = s_box (st [4]) ;
  st [8] = s_box (st [8]) ; st [12] = s_box (st [12]) ;

  byte tt = st [1] ;
  st [1] = s_box (st [5]) ;  st [5]  = s_box (st [9]) ;
  st [9] = s_box (st [13]) ; st [13] = s_box (tt) ;

  tt = st[2] ; st [2] = s_box (st [10]) ; st [10] = s_box (tt) ;
  tt = st[6] ; st [6] = s_box (st [14]) ; st [14] = s_box (tt) ;

  tt = st[15] ;
  st [15] = s_box (st [11]) ; st [11] = s_box (st [7]) ;
  st [7]  = s_box (st [3]) ;  st [3]  = s_box (tt) ;
}

static void inv_shift_sub_rows (byte st[N_BLOCK])
```

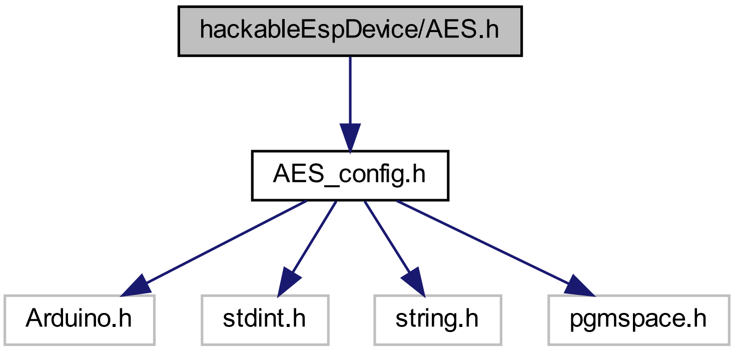
```
{
  st [0] = is_box (st[0]) ; st [4] = is_box (st [4]);
  st [8] = is_box (st[8]) ; st [12] = is_box (st [12]);

  byte tt = st[13] ;
  st [13] = is_box (st [9]) ; st [9] = is_box (st [5]) ;
  st [5]  = is_box (st [1]) ; st [1] = is_box (tt) ;

  tt = st [2] ; st [2] = is_box (st [10]) ; st [10] = is_box (tt) ;
  tt = st [6] ; st [6] = is_box (st [14]) ; st [14] = is_box (tt) ;

  tt = st [3] ;
  st [3]  = is_box (st [7])  ; st [7]  = is_box (st [11]) ;
  st [11] = is_box (st [15]) ; st [15] = is_box (tt) ;
}

/* SUB COLUMNS PHASE */

static void mix_sub_columns (byte dt[N_BLOCK], byte st[N_BLOCK])
{
  byte j = 5 ;
  byte k = 10 ;
  byte l = 15 ;
  for (byte i = 0 ; i < N_BLOCK ; i += N_COL)
    {
      byte a = st [i] ;
      byte b = st [j] ;  j = (j+N_COL) & 15 ;
      byte c = st [k] ;  k = (k+N_COL) & 15 ;
      byte d = st [l] ;  l = (l+N_COL) & 15 ;
      byte a1 = s_box (a), b1 = s_box (b), c1 = s_box (c), d1 = s_box (d) ;
      byte a2 = f2(a1),    b2 = f2(b1),    c2 = f2(c1),    d2 = f2(d1) ;
      dt[i]   = a2     ^  b2^b1  ^  c1     ^  d1 ;
      dt[i+1] = a1     ^  b2     ^  c2^c1  ^  d1 ;
      dt[i+2] = a1     ^  b1     ^  c2     ^  d2^d1 ;
      dt[i+3] = a2^a1  ^  b1     ^  c1     ^  d2 ;
    }
}

static void inv_mix_sub_columns (byte dt[N_BLOCK], byte st[N_BLOCK])
{
  for (byte i = 0 ; i < N_BLOCK ; i += N_COL)
    {
      byte a1 = st [i] ;
      byte b1 = st [i+1] ;
      byte c1 = st [i+2] ;
      byte d1 = st [i+3] ;
      byte a2 = f2(a1), b2 = f2(b1), c2 = f2(c1), d2 = f2(d1) ;
      byte a4 = f2(a2), b4 = f2(b2), c4 = f2(c2), d4 = f2(d2) ;
      byte a8 = f2(a4), b8 = f2(b4), c8 = f2(c4), d8 = f2(d4) ;
      byte a9 = a8 ^ a1,b9 = b8 ^ b1,c9 = c8 ^ c1,d9 = d8 ^ d1 ;
      byte ac = a8 ^ a4,bc = b8 ^ b4,cc = c8 ^ c4,dc = d8 ^ d4 ;

      dt[i]          = is_box (ac^a2  ^  b9^b2  ^  cc^c1  ^  d9) ;
      dt[(i+5)&15]   = is_box (a9     ^  bc^b2  ^  c9^c2  ^  dc^d1) ;
      dt[(i+10)&15]  = is_box (ac^a1  ^  b9     ^  cc^c2  ^  d9^d2) ;
      dt[(i+15)&15]  = is_box (a9^a2  ^  bc^b1  ^  c9     ^  dc^d2) ;
    }
}

/******************************************************************************/

AES::AES(){
    byte ar_iv[8] = { 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01 };
    memcpy(iv,ar_iv,8);
    memcpy(iv+8,ar_iv,8);
    arr_pad[0] = 0x01;
    arr_pad[1] = 0x02;
    arr_pad[2] = 0x03;
    arr_pad[3] = 0x04;
    arr_pad[4] = 0x05;
    arr_pad[5] = 0x06;
    arr_pad[6] = 0x07;
    arr_pad[7] = 0x08;
    arr_pad[8] = 0x09;
    arr_pad[9] = 0x0a;
    arr_pad[10] = 0x0b;
    arr_pad[11] = 0x0c;
    arr_pad[12] = 0x0d;
    arr_pad[13] = 0x0e;
    arr_pad[14] = 0x0f;
    arr_pad[15] = 0x10;
}

/******************************************************************************/

byte AES::set_key (byte key [], int keylen)
{
  byte hi ;
  switch (keylen)
    {
    case 16:
    case 128:
      keylen = 16; // 10 rounds
      round = 10 ;
      break;
    case 24:
    case 192:
      keylen = 24; // 12 rounds
      round = 12 ;
      break;
    case 32:
    case 256:
      keylen = 32; // 14 rounds
      round = 14 ;
      break;
    default:
      round = 0;
      return FAILURE;
    }
  hi = (round + 1) « 4 ;
  copy_n_bytes (key_sched, key, keylen) ;
  byte t[4] ;
  byte next = keylen ;
  for (byte cc = keylen, rc = 1 ; cc < hi ; cc += N_COL)
    {
      for (byte i = 0 ; i < N_COL ; i++)
        t[i] = key_sched [cc-4+i] ;
      if (cc == next)
        {
          next += keylen ;
          byte ttt = t[0] ;
          t[0] = s_box (t[1]) ^ rc ;
          t[1] = s_box (t[2]) ;
          t[2] = s_box (t[3]) ;
          t[3] = s_box (ttt) ;
          rc = f2 (rc) ;
        }
      else if (keylen == 32 && (cc & 31) == 16)
        {
          for (byte i = 0 ; i < 4 ; i++)
            t[i] = s_box (t[i]) ;
        }
      byte tt = cc - keylen ;
      for (byte i = 0 ; i < N_COL ; i++)
        key_sched [cc + i] = key_sched [tt + i] ^ t[i] ;
    }
  return SUCCESS ;
}

/******************************************************************************/

void AES::clean ()
{
  for (byte i = 0 ; i < KEY_SCHEDULE_BYTES ; i++)
    key_sched [i] = 0 ;
  round = 0 ;
}

/******************************************************************************/

void AES::copy_n_bytes (byte * d, byte * s, byte nn)
{
  while (nn >= 4)
    {
      *d++ = *s++ ;  // some unrolling
      *d++ = *s++ ;
      *d++ = *s++ ;
      *d++ = *s++ ;
      nn -= 4 ;
    }
  while (nn--)
    *d++ = *s++ ;
}

/******************************************************************************/

byte AES::encrypt (byte plain [N_BLOCK], byte cipher [N_BLOCK])
{
  if (round)
    {
      byte s1 [N_BLOCK], r ;
      copy_and_key (s1, plain, (byte*) (key_sched)) ;

      for (r = 1 ; r < round ; r++)
```

```
        {
          byte s2 [N_BLOCK] ;
          mix_sub_columns (s2, s1) ;
          copy_and_key (s1, s2, (byte*) (key_sched + r * N_BLOCK)) ;
        }
      shift_sub_rows (s1) ;
      copy_and_key (cipher, s1, (byte*) (key_sched + r * N_BLOCK)) ;
    }
  else
    return FAILURE ;
  return SUCCESS ;
}

/******************************************************************************/

byte AES::cbc_encrypt (byte * plain, byte * cipher, int n_block, byte iv [N_BLOCK])
{
  while (n_block--)
    {
      xor_block (iv, plain) ;
      if (encrypt (iv, iv) != SUCCESS)
        return FAILURE ;
      copy_n_bytes (cipher, iv, N_BLOCK) ;
      plain  += N_BLOCK ;
      cipher += N_BLOCK ;
    }
  return SUCCESS ;
}

/******************************************************************************/

byte AES::cbc_encrypt (byte * plain, byte * cipher, int n_block)
{
  while (n_block--)
    {
      xor_block (iv, plain) ;
      if (encrypt (iv, iv) != SUCCESS)
        return FAILURE ;
      copy_n_bytes (cipher, iv, N_BLOCK) ;
      plain  += N_BLOCK ;
      cipher += N_BLOCK ;
    }
  return SUCCESS ;
}

/******************************************************************************/

byte AES::decrypt (byte plain [N_BLOCK], byte cipher [N_BLOCK])
{
  if (round)
    {
      byte s1 [N_BLOCK] ;
      copy_and_key (s1, plain, (byte*) (key_sched + round * N_BLOCK)) ;
      inv_shift_sub_rows (s1) ;

      for (byte r = round ; --r ; )
       {
         byte s2 [N_BLOCK] ;
         copy_and_key (s2, s1, (byte*) (key_sched + r * N_BLOCK)) ;
         inv_mix_sub_columns (s1, s2) ;
       }
      copy_and_key (cipher, s1, (byte*) (key_sched)) ;
    }
  else
    return FAILURE ;
  return SUCCESS ;
}

/******************************************************************************/

byte AES::cbc_decrypt (byte * cipher, byte * plain, int n_block, byte iv [N_BLOCK])
{
  while (n_block--)
    {
      byte tmp [N_BLOCK] ;
      copy_n_bytes (tmp, cipher, N_BLOCK) ;
      if (decrypt (cipher, plain) != SUCCESS)
        return FAILURE ;
      xor_block (plain, iv) ;
      copy_n_bytes (iv, tmp, N_BLOCK) ;
      plain  += N_BLOCK ;
      cipher += N_BLOCK;
    }
  return SUCCESS ;
}

/******************************************************************************/
```

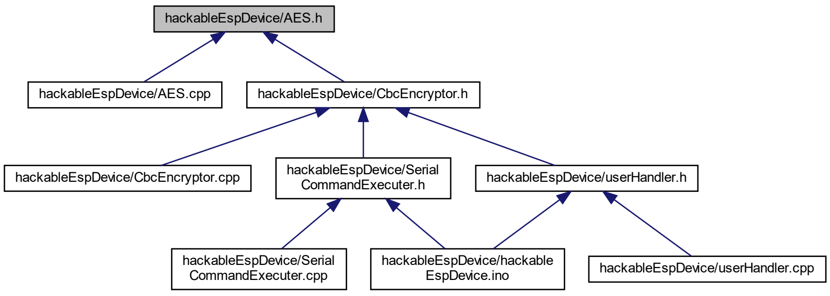
```

byte AES::cbc_decrypt (byte * cipher, byte * plain, int n_block)
{
  while (n_block--)
    {
      byte tmp [N_BLOCK] ;
      copy_n_bytes (tmp, cipher, N_BLOCK) ;
      if (decrypt (cipher, plain) != SUCCESS)
        return FAILURE ;
      xor_block (plain, iv) ;
      copy_n_bytes (iv, tmp, N_BLOCK) ;
      plain  += N_BLOCK ;
      cipher += N_BLOCK;
    }
  return SUCCESS ;
}

/******************************************************************************/

void AES::set_IV(unsigned long long int IVCl){
    memcpy(iv,&IVCl,8);
    memcpy(iv+8,&IVCl,8);
    IVC = IVCl;
}

/******************************************************************************/

void AES::iv_inc(){
    IVC += 1;
    memcpy(iv,&IVC,8);
    memcpy(iv+8,&IVC,8);
}

/******************************************************************************/

int AES::get_size(){
    return size;
}

/******************************************************************************/

int AES::get_pad(){
    return pad;
}

/******************************************************************************/

void AES::set_size(int sizel){
    size = sizel;
}


/******************************************************************************/

void AES::get_IV(byte *out){
    memcpy(out,&IVC,8);
    memcpy(out+8,&IVC,8);
}

/******************************************************************************/

void AES::calc_size_n_pad(int p_size){
    pad = N_BLOCK - p_size % N_BLOCK;
    size = p_size + pad;
}

/******************************************************************************/

void AES::padPlaintext(void* in,byte* out)
{
    memcpy(out,in,size);
    for (int i = size-pad; i < size; i++){;
        out[i] = arr_pad[pad - 1];
    }
}

/******************************************************************************/

bool AES::CheckPad(byte* in,int lsize){
    if (in[lsize-1] <= 0x10){
        int lpad = (int)in[lsize-1];
        for (int i = lsize - 1; i >= lsize-lpad; i--){
            if (arr_pad[lpad - 1] != in[i]){
                return false;
            }
        }
    }else{
        return true;
    }
return true;
}

/******************************************************************************/

void AES::printArray(byte output[],bool p_pad)
{
uint8_t i,j;
uint8_t loops = size/N_BLOCK;
uint8_t outp = N_BLOCK;
for (j = 0; j < loops; j += 1){
  if (p_pad && (j == (loops  - 1)) ) { outp = N_BLOCK - pad; }
  for (i = 0; i < outp; i++)
  {
    printf_P(PSTR("%c"),output[j*N_BLOCK + i]);
  }
}
  printf_P(PSTR("\n"));
}

/******************************************************************************/

void AES::printArray(byte output[],int sizel)
{
  for (int i = 0; i < sizel; i++)
  {
    printf_P(PSTR("%02x"),output[i]); // print hex in fixed 2-cgar format
  }
  printf_P(PSTR("\n"));
}


/******************************************************************************/

void AES::do_aes_encrypt(byte *plain,int size_p,byte *cipher,byte *key, int bits, byte ivl [N_BLOCK]){
    calc_size_n_pad(size_p);
    byte plain_p[get_size()];
    padPlaintext(plain,plain_p);
    int blocks = get_size() / N_BLOCK;
    set_key (key, bits) ;
    cbc_encrypt (plain_p, cipher, blocks, ivl);
}

/******************************************************************************/

void AES::do_aes_encrypt(byte *plain,int size_p,byte *cipher,byte *key, int bits){
    calc_size_n_pad(size_p);
    byte plain_p[get_size()];
    padPlaintext(plain,plain_p);
    int blocks = get_size() / N_BLOCK;
    set_key (key, bits) ;
    cbc_encrypt (plain_p, cipher, blocks);
}

/******************************************************************************/

void AES::do_aes_decrypt(byte *cipher,int size_c,byte *plain,byte *key, int bits, byte ivl [N_BLOCK]){
    set_size(size_c);
    int blocks = size_c / N_BLOCK;
    set_key (key, bits);
    cbc_decrypt (cipher,plain, blocks, ivl);
}

/******************************************************************************/

void AES::do_aes_decrypt(byte *cipher,int size_c,byte *plain,byte *key, int bits){
    set_size(size_c);
    int blocks = size_c / N_BLOCK;
    set_key (key, bits);
    cbc_decrypt (cipher,plain, blocks);
}


/******************************************************************************/

#if defined(AES_LINUX)
double AES::millis(){
    gettimeofday(&tv, NULL);
    return (tv.tv_sec + 0.000001 * tv.tv_usec);
}
#endif
```

5.3 hackableEspDevice/AES.h File Reference

`#include "AES_config.h"`
Include dependency graph for AES.h:



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


hackableEspDevice/AES.h
hackableEspDevice/AES.cpp
hackableEspDevice/CbcEncryptor.h
hackableEspDevice/CbcEncryptor.cpp
hackableEspDevice/Serial CommandExecuter.h
hackableEspDevice/userHandler.h
hackableEspDevice/Serial CommandExecuter.cpp
hackableEspDevice/hackable EspDevice.ino
hackableEspDevice/userHandler.cpp

Classes

- class AES

5.4 AES.h

Go to the documentation of this file.

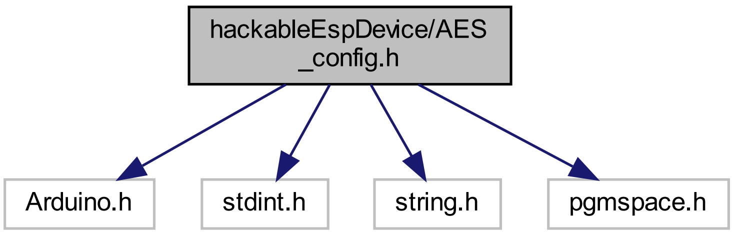
```
#ifndef __AES_H__
#define __AES_H__

#include "AES_config.h"
/*
 ---------------------------------------------------------------------------
 Copyright (c) 1998-2008, Brian Gladman, Worcester, UK. All rights reserved.

```

```
LICENSE TERMS

The redistribution and use of this software (with or without changes)
is allowed without the payment of fees or royalties provided that:

 1. source code distributions include the above copyright notice, this
    list of conditions and the following disclaimer;

 2. binary distributions include the above copyright notice, this list
    of conditions and the following disclaimer in their documentation;

 3. the name of the copyright holder is not used to endorse products
    built using this software without specific written permission.

DISCLAIMER

This software is provided 'as is' with no explicit or implied warranties
in respect of its properties, including, but not limited to, correctness
and/or fitness for purpose.
---------------------------------------------------------------------------
Issue 09/09/2006

This is an AES implementation that uses only 8-bit byte operations on the
cipher state.
*/

/* code was modified by george spanos <spaniakos@gmail.com>
* 16/12/14
*/

class AES
{
 public:

/*  The following calls are for a precomputed key schedule

    NOTE: If the length_type used for the key length is an
    unsigned 8-bit character, a key length of 256 bits must
    be entered as a length in bytes (valid inputs are hence
    128, 192, 16, 24 and 32).
*/
    AES();

    byte set_key (byte key[], int keylen) ;

    void clean () ;  // delete key schedule after use

    void copy_n_bytes (byte * AESt, byte * src, byte n) ;

    byte encrypt (byte plain [N_BLOCK], byte cipher [N_BLOCK]) ;

    byte cbc_encrypt (byte * plain, byte * cipher, int n_block, byte iv [N_BLOCK]) ;

    byte cbc_encrypt (byte * plain, byte * cipher, int n_block) ;


    byte decrypt (byte cipher [N_BLOCK], byte plain [N_BLOCK]) ;

    byte cbc_decrypt (byte * cipher, byte * plain, int n_block, byte iv [N_BLOCK]) ;

    byte cbc_decrypt (byte * cipher, byte * plain, int n_block) ;

    void set_IV(unsigned long long int IVCl);

    void iv_inc();

    int get_size();

    void set_size(int sizel);

  int get_pad();

    void get_IV(byte *out);

    void calc_size_n_pad(int p_size);

    void padPlaintext(void* in,byte* out);

    bool CheckPad(byte* in,int size);

    void printArray(byte output[],bool p_pad = true);

    void printArray(byte output[],int sizel);

    void do_aes_encrypt(byte *plain,int size_p,byte *cipher,byte *key, int bits, byte ivl [N_BLOCK]);

    void do_aes_encrypt(byte *plain,int size_p,byte *cipher,byte *key, int bits);

    void do_aes_decrypt(byte *cipher,int size_c,byte *plain,byte *key, int bits, byte ivl [N_BLOCK]);

    void do_aes_decrypt(byte *cipher,int size_c,byte *plain,byte *key, int bits);

    #if defined(AES_LINUX)
        double millis();
    #endif
 private:
  int round ;
  byte key_sched [KEY_SCHEDULE_BYTES] ;
  unsigned long long int IVC;
  byte iv[16];
  int pad;
  int size;
  #if defined(AES_LINUX)
    timeval tv;
    byte arr_pad[16];
  #else
    byte arr_pad[16] = {
 0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09,0x0a,0x0b,0x0c,0x0d,0x0e,0x0f,0x10 };
  #endif
} ;


#endif

```

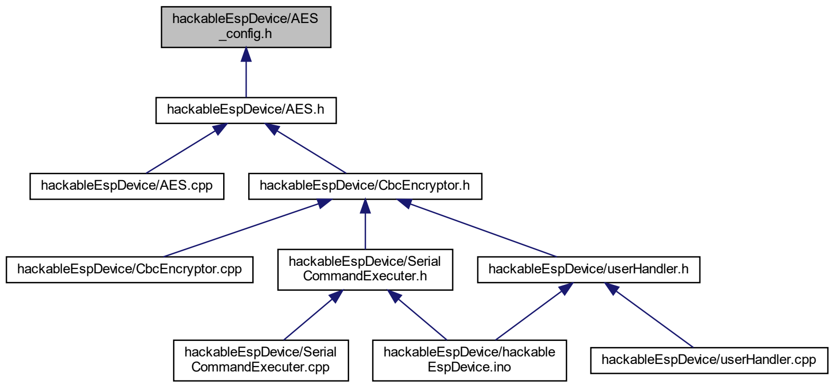
5.5 hackableEspDevice/AES_config.h File Reference

```
#include <Arduino.h>
#include <stdint.h>
#include <string.h>
#include <pgmspace.h>
```

Include dependency graph for AES_config.h:



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


hackableEspDevice/AES_config.h
hackableEspDevice/AES.h
hackableEspDevice/AES.cpp
hackableEspDevice/CbcEncryptor.h
hackableEspDevice/CbcEncryptor.cpp
hackableEspDevice/SerialCommandExecuter.h
hackableEspDevice/userHandler.h
hackableEspDevice/SerialCommandExecuter.cpp
hackableEspDevice/hackableEspDevice.ino
hackableEspDevice/userHandler.cpp

Macros

- #define N_ROW 4
- #define N_COL 4
- #define N_BLOCK (N_ROW ∗ N_COL)
- #define N_MAX_ROUNDS 14
- #define KEY_SCHEDULE_BYTES ((N_MAX_ROUNDS + 1) ∗ N_BLOCK)
- #define SUCCESS (0)
- #define FAILURE (-1)

5.5.1 Macro Definition Documentation

5.5.1.1 FAILURE

```
#define FAILURE (-1)
```

Definition at line 53 of file AES_config.h.

5.5.1.2 KEY_SCHEDULE_BYTES

```
#define KEY_SCHEDULE_BYTES ((N_MAX_ROUNDS + 1) * N_BLOCK)
```

Definition at line 51 of file AES_config.h.

5.5.1.3 N_BLOCK

```
#define N_BLOCK (N_ROW * N_COL)
```

Definition at line 49 of file AES_config.h.

5.5.1.4 N_COL

```
#define N_COL 4
```

Definition at line 48 of file AES_config.h.

5.5.1.5 N_MAX_ROUNDS

```
#define N_MAX_ROUNDS 14
```

Definition at line 50 of file AES_config.h.

5.5.1.6 N_ROW

```
#define N_ROW 4
```

Definition at line 47 of file AES_config.h.

5.5.1.7 SUCCESS

```
#define SUCCESS (0)
```

Definition at line 52 of file AES_config.h.

5.6 AES_config.h

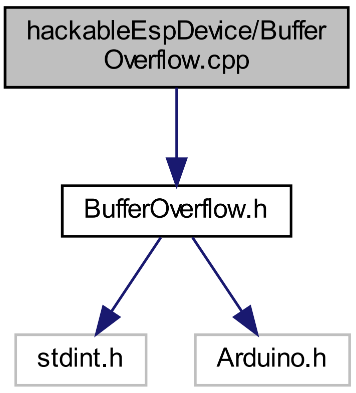
Go to the documentation of this file.

```
/* code was modified by george spanos <spaniakos@gmail.com>
 * 16/12/14
 */

#ifndef __AES_CONFIG_H__
#define __AES_CONFIG_H__

#if  (defined(__linux) || defined(linux)) && !(defined(__ARDUINO_X86__) || defined(__arm__))

  #define AES_LINUX

  #include <stdint.h>
  #include <stdio.h>
  #include <stdlib.h>
  #include <string.h>
  #include <sys/time.h>
  #include <unistd.h>
#else
  #include <Arduino.h>
#endif

#include <stdint.h>
#include <string.h>

#if defined(__ARDUINO_X86__) || defined(__arm__) || (defined (__linux) || defined (linux))
    #undef PROGMEM
    #define PROGMEM __attribute__(( section(".progmem.data") ))
    #define pgm_read_byte(p) (*(p))
    typedef unsigned char byte;
    #define printf_P printf
        #ifndef PSTR
      #define PSTR(x) (x)
        #endif
#elif defined ( ESP8266 )
    #include <pgmspace.h>
        #ifndef PSTR
          #define PSTR(x) (x)
        #endif
#else
    #if (defined(__AVR__))
        #include <avr/pgmspace.h>
    #else
        #include <pgmspace.h>
    #endif
#endif

#define N_ROW                   4
#define N_COL                   4
#define N_BLOCK   (N_ROW * N_COL)
#define N_MAX_ROUNDS           14
#define KEY_SCHEDULE_BYTES ((N_MAX_ROUNDS + 1) * N_BLOCK)
#define SUCCESS (0)
#define FAILURE (-1)

#endif
```

5.7 hackableEspDevice/BufferOverflow.cpp File Reference

`#include "BufferOverflow.h"`

Include dependency graph for BufferOverflow.cpp:



5.8 BufferOverflow.cpp

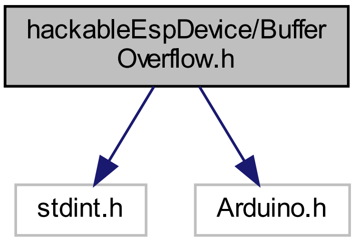
Go to the documentation of this file.

```
/*
 * File:       BufferOverflow.h
 * Author:     Luke de Munk
 * Class:      BufferOverflow
 * Version:    1.0
 *
 * Buffer overflow simulator based on a Linux cli.
 * All elements of the bufferflow are in this class.
 */
#include "BufferOverflow.h"

/******************************************************************************/
/******************************************************************************/
BufferOverflow::BufferOverflow() {
    _clearInput();                                                     //First time call is the declaration of the array.
}

/******************************************************************************/
/******************************************************************************/
void BufferOverflow::ls() {
    Serial.println(F("testprogram.c"));
    Serial.println(F("testprogram"));
}

/******************************************************************************/
/******************************************************************************/
void BufferOverflow::vi() {
    Serial.println(F("|-----------FILENAME-----------|-----------TYPE-----------|-----------AUTOR-----------|"));
    Serial.println(F("|---------testprogram.c--------|---------READONLY---------|-----------admin-----------|"));
    Serial.println(F("|--------------------------------------------------------------------------------------|"));
    Serial.println(F("1      /*                                                                               |"));
    Serial.println(F("2       * File:   testprogram.c                                                         |"));
    Serial.println(F("3       * Author: admin                                                                 |"));
    Serial.println(F("4      *
      |"));
    Serial.println(F("5      * To test superuser login. DELETE WHEN FINISHING DEVELOPMENT!!!
      |"));
    Serial.println(F("6      */
      |"));
    Serial.println(F("7     #include <stdio.h>
      |"));
    Serial.println(F("8     #include <string.h>
      |"));
    Serial.println(F("9
      |"));
    Serial.println(F("10
      |"));
    Serial.println(F("11
 /**************************************************************************/    |"));
    Serial.println(F("12    /*!
      |"));
    Serial.println(F("13      @brief    Logs given user name in as superuser and logs out again.
      |"));
    Serial.println(F("14    */
      |"));
    Serial.println(F("15
 /**************************************************************************/    |"));
    Serial.println(F("16    int main(int argc, char** argv) {
      |"));
    Serial.println(F("17        char username[10];
      |"));
    Serial.println(F("18        strcpy(username, argv[1]);
      |"));
    Serial.println(F("19        login(*username);
      |"));
    Serial.println(F("20        logout();
      |"));
    Serial.println(F("21
      |"));
    Serial.println(F("22        return 0;
      |"));
    Serial.println(F("23    }
      |"));

 Serial.println(F("|--------------------------------------------------------------------------------------|"));
    Serial.println(F(""));
}

/**************************************************************************/
/**************************************************************************/
void BufferOverflow::objectDump() {
    Serial.println(F("testprogram:     file format elf32-littlearm"));
    Serial.println(F(""));
    Serial.println(F("Disassembly of section .init:"));
    Serial.println(F(""));
    Serial.println(F("00010438 <main>:"));
    Serial.println(F("   10438: e92d4800  push  {fp, lr}"));
    Serial.println(F("   1043c: e28db004  add fp, sp, #4"));
    Serial.println(F("   10440: e24dd018  sub sp, sp, #24"));
    Serial.println(F("   10444: e50b0018  str r0, [fp, #-24]  ; 0xffffffe8"));
    Serial.println(F("   10448: e50b101c  str r1, [fp, #-28]  ; 0xffffffe4"));
    Serial.println(F("   1044c: e51b301c  ldr r3, [fp, #-28]  ; 0xffffffe4"));
    Serial.println(F("   10450: e2833004  add r3, r3, #4"));
    Serial.println(F("   10454: e5932000  ldr r2, [r3]"));
    Serial.println(F("   10458: e24b3010  sub r3, fp, #16"));
    Serial.println(F("   1045c: e1a01002  mov r1, r2"));
    Serial.println(F("   10460: e1a00003  mov r0, r3"));
    Serial.println(F("   10464: ebfffffab  bl  10318 <strcpy@plt>"));
    Serial.println(F("   10468: e55b3010  ldrb  r3, [fp, #-16]"));
    Serial.println(F("   1046c: e1a00003  mov r0, r3"));
    Serial.println(F("   10470: eb000004  bl  10488 <login>"));
    Serial.println(F("   10474: eb00000d  bl  104b0 <logout>"));
    Serial.println(F("   10478: e3a03000  mov r3, #0"));
    Serial.println(F("   1047c: e1a00003  mov r0, r3"));
    Serial.println(F("   10480: e24bd004  sub sp, fp, #4"));
    Serial.println(F("   10484: e8bd8800  pop {fp, pc}"));
    Serial.println(F(""));
    Serial.println(F("00010488 <login>:"));
    Serial.println(F("   10488: e92d4800  push  {fp, lr}"));
    Serial.println(F("   1048c: e28db004  add fp, sp, #4"));
    Serial.println(F("   10490: e24dd008  sub sp, sp, #8"));
    Serial.println(F("   10494: e50b0008  str r0, [fp, #-8]"));
    Serial.println(F("   10498: e59f000c  ldr r0, [pc, #12] ; 104ac <login+0x24>"));
    Serial.println(F("   1049c: ebffff9a  bl  1030c <printf@plt>"));
    Serial.println(F("   104a0: e1a00000  nop     ; (mov r0, r0)"));
    Serial.println(F("   104a4: e24bd004  sub sp, fp, #4"));
    Serial.println(F("   104a8: e8bd8800  pop {fp, pc}"));
    Serial.println(F("   104ac: 0001053c  .word 0x0001053c"));
    Serial.println(F(""));
    Serial.println(F("000104b0 <logout>:"));
    Serial.println(F("   104b0: e92d4800  push  {fp, lr}"));
    Serial.println(F("   104b4: e28db004  add fp, sp, #4"));
    Serial.println(F("   104b8: e59f0008  ldr r0, [pc, #8]  ; 104c8 <logout+0x18>"));
    Serial.println(F("   104bc: ebffff92  bl  1030c <printf@plt>"));
    Serial.println(F("   104c0: e1a00000  nop     ; (mov r0, r0)"));
    Serial.println(F("   104c4: e8bd8800  pop {fp, pc}"));
    Serial.println(F("   104c8: 00010548  .word 0x00010548"));
    Serial.println(F(""));
}

/**************************************************************************/
/**************************************************************************/
bool BufferOverflow::runCProgram(String arg) {
    _formatInput(arg);

    if (_numChars < OVERFLOW_BEGIN) {
        Serial.println("You are now super user.");
        Serial.print("Hello ");
        Serial.println(arg);
        Serial.println("You are not longer super user.");
    } else {
        if (_checkBufferOverflow()) {
            return true;
        }
    }
    return false;
}

/**************************************************************************/
/**************************************************************************/
bool BufferOverflow::_checkBufferOverflow() {
    if(_getOverflowPortion() == RETURN_ADDRESS) {
        return true;
    }

    _printOverflowError();                                              //If the overflow is not correctly, print value of the return address pointer
    return false;
}

/**************************************************************************/
/**************************************************************************/
void BufferOverflow::_printOverflowError() {
    Serial.println("Program received signal SIGSEGV, Segmentation fault.");
    Serial.print("0x");
    _getOverflowPortion(true);
    Serial.println(" in ?? ()");
}

/**************************************************************************/
/**************************************************************************/
void BufferOverflow::_formatInput(String input) {
    String tmp = "";

    _clearInput();

    /* Set every character in an element */
    for (uint16_t i = 0; i < input.length(); i++) {
        if (input[i] == '\\') {
            _formattedInput[_numChars] = "\\x";                          //Move all hex chars in one element (for ex.: '\x90')
            _formattedInput[_numChars] += input[i+2];
            _formattedInput[_numChars] += input[i+3];
            i += 3;                                                      //Increase with 3, because the number of chars taken for a hex is 4 ('\x90')
        } else {
            _formattedInput[_numChars] = input[i];
        }
        _numChars++;
    }

    /* Turn the whole array, to simulate little endian systems */
    for (uint8_t i = 0; i < _numChars/2-1; i++) {
        tmp = _formattedInput[i];
        _formattedInput[i] = _formattedInput[_numChars-i-1];
        _formattedInput[_numChars-i-1] = tmp;
        tmp = "";
    }
}

/**************************************************************************/
/**************************************************************************/
String BufferOverflow::_getOverflowPortion(bool print) {
    String overflowPortion = "";

    if (_numChars < OVERFLOW_LENGTH) {
        uint8_t numMissingBytes = OVERFLOW_LENGTH - _numChars;
        overflowPortion += _generateRandomBytes(numMissingBytes);

        if (print) {
            Serial.print(overflowPortion);
        }

        /* To determine and print the overflow portion */
        for (uint8_t i = 0; i < ADDRESS_LENGTH - numMissingBytes; i++) {
            /* Check if is hex number, else print as hex */
            if (_formattedInput[i][0] == '\\') {
                overflowPortion += _formattedInput[i][2];
                overflowPortion += _formattedInput[i][3];
                if (print) {
                    Serial.print(_formattedInput[i][2]);
                    Serial.print(_formattedInput[i][3]);
                }
            } else {
                overflowPortion += _formattedInput[i];
                if (print) {
                    Serial.print(char(_formattedInput[i][0]), HEX);
                }
            }
        }
    } else {
        /* To print the overflow portion */
        uint8_t delta = abs(_numChars - OVERFLOW_LENGTH);
        for (uint8_t i = delta; i < delta + ADDRESS_LENGTH; i++) {
            /* Check if is hex number, else print as hex */
            if (_formattedInput[i][0] == '\\') {
                overflowPortion += _formattedInput[i][2];
                overflowPortion += _formattedInput[i][3];
                if (print) {
                    Serial.print(_formattedInput[i][2]);
                    Serial.print(_formattedInput[i][3]);
                }
            } else {
                if (print) {
                    Serial.print(char(_formattedInput[i][0]), HEX);
                }
                overflowPortion += _formattedInput[i];
            }
        }
    }
    return overflowPortion;
}

/*****************************************************************************/
/*****************************************************************************/
void BufferOverflow::_clearInput() {
    for (uint16_t i = 0; i < MAX_NUM_CHARS; i++) {
        _formattedInput[i] = "";
    }
    _numChars = 0;
}

/*****************************************************************************/
/*****************************************************************************/
String BufferOverflow::_generateRandomBytes(uint8_t numBytes) {
    String bytes = "";
    randomSeed(numBytes);

    for (uint8_t i = 0; i < numBytes; i++) {
        bytes += String(random(127), HEX);
    }
    return bytes;
}
```

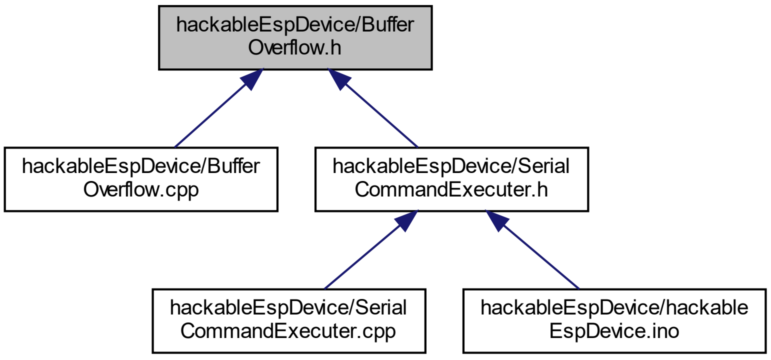
5.9 hackableEspDevice/BufferOverflow.h File Reference

```
#include <stdint.h>
#include "Arduino.h"
```

Include dependency graph for BufferOverflow.h:



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



Classes

- class BufferOverflow

Macros

- #define OVERFLOW_BEGIN 16
- #define ADDRESS_LENGTH 4
- #define OVERFLOW_LENGTH 20
- #define RETURN_ADDRESS "00010488"
- #define MAX_NUM_CHARS 256

5.9.1 Macro Definition Documentation

5.9.1.1 ADDRESS_LENGTH

```
#define ADDRESS_LENGTH 4
```

Definition at line 16 of file BufferOverflow.h.

5.9.1.2 MAX_NUM_CHARS

```
#define MAX_NUM_CHARS 256
```

Definition at line 19 of file BufferOverflow.h.

5.9.1.3 OVERFLOW_BEGIN

```
#define OVERFLOW_BEGIN 16
```

Definition at line 15 of file BufferOverflow.h.

5.9.1.4 OVERFLOW_LENGTH

```
#define OVERFLOW_LENGTH 20
```

Definition at line 17 of file BufferOverflow.h.

5.9.1.5 RETURN_ADDRESS

```
#define RETURN_ADDRESS "00010488"
```

Definition at line 18 of file BufferOverflow.h.

5.10 BufferOverflow.h

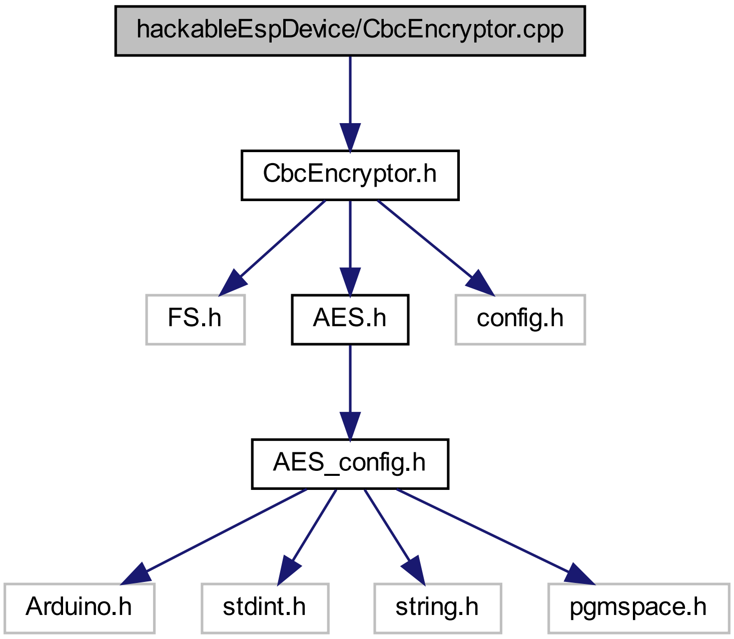
Go to the documentation of this file.

```
/*
 * File:      BufferOverflow.h
 * Author:    Luke de Munk
 * Class:     BufferOverflow
 * Version:   1.0
 *
 * Buffer overflow simulator based on a Linux cli.
 * All elements of the bufferflow are in this class.
 */
#ifndef BUFFER_OVERFLOW_H
#define BUFFER_OVERFLOW_H
#include <stdint.h>                                                   //For defining bits per integer
#include "Arduino.h"

#define OVERFLOW_BEGIN    16                                          //Because array is in byte resolution, 10 becomes 16. Then the return address pointer starts
#define ADDRESS_LENGTH    4                                           //Address is 32 bits long, so 4 bytes
#define OVERFLOW_LENGTH   20                                          //OVERFLOW_BEGIN + ADDRESS_LENGTH
#define RETURN_ADDRESS    "00010488"                                  //0x00010488 == address of login function.
#define MAX_NUM_CHARS     256

class BufferOverflow
{
    public:
        BufferOverflow();
        void ls();
        void vi();
        void objectDump();
        bool runCProgram(String arg);

    private:
        bool _checkBufferOverflow();
        void _printOverflowError();
        void _formatInput(String input);
        String _getOverflowPortion(bool print = false);
        void _clearInput();
        String _generateRandomBytes(uint8_t numberOfBytes);

        String _formattedInput[256];
        uint8_t _numChars;
};
#endif
```

5.11 hackableEspDevice/CbcEncryptor.cpp File Reference

`#include "CbcEncryptor.h"`

Include dependency graph for CbcEncryptor.cpp:



5.12 CbcEncryptor.cpp

Go to the documentation of this file.

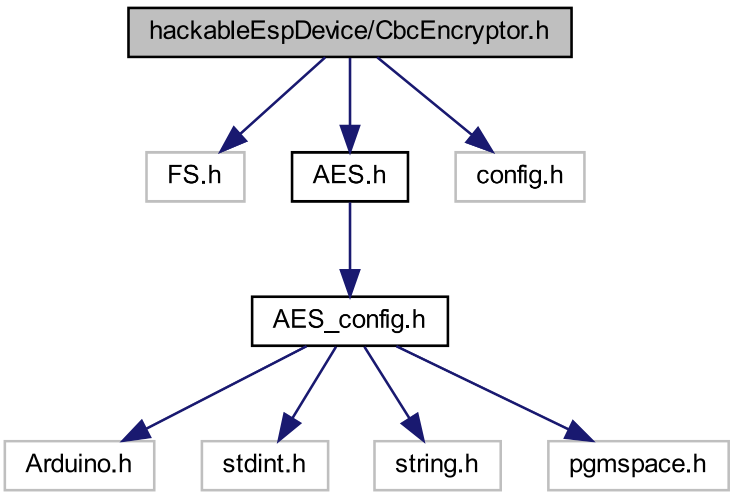
```
/*
 * File:      CbcEncryptor.cpp
 * Author:    Luke de Munk
 * Class:     CbcEncryptor
 * Version:   1.0
 *
 * Handles encryption and decryption of files. Uses AES encryption.
 */
#include "CbcEncryptor.h"

/**************************************************************************/
/**************************************************************************/
CbcEncryptor::CbcEncryptor() {
    _aesInitVecInt = CBC_INIT_VECTOR;
    _aesKeyString = AES_KEY;

    /* Convert key string to bytes */
    for (uint8_t i = 0; i < AES_KEY_SIZE; i++) {
        _aesKey[i] = (byte) _aesKeyString[i];
    }
}

/**************************************************************************/
/**************************************************************************/
bool CbcEncryptor::encryptFile(String filename) {
    if (!SPIFFS.exists(filename)) {
        return false;
    }

    setKey(AES_KEY);                                                        //Reset key
    _aesKeyString = AES_KEY;

    File inputFile = SPIFFS.open(filename, "r");                            //Plaintext file
    File tmpFile = SPIFFS.open("tmp_" + filename, "w");                     //Temporary file to store encrypted part

    /* If already is plaintext, don't do anything */
    if (inputFile.readStringUntil('\n') == "format: encrypted") {
        return false;
    }

    tmpFile.println("format: encrypted");                                   //Save format
    String line = inputFile.readStringUntil('\n');

    _aes.iv_inc();                                                          //Unique initialization vector every encryption

    /* Encrypt the file line by line */
    while (line != "") {
        tmpFile.println(encryptLine(line));
        line = inputFile.readStringUntil('\n');                             //Read a line from the file
    }

    inputFile.close();
    tmpFile.close();

    SPIFFS.remove(filename);                                                //Remove plain text file
    SPIFFS.rename("tmp_" + filename, filename);                             //Rename tmp file
    return true;
}

/**************************************************************************/
/**************************************************************************/
bool CbcEncryptor::decryptFile(String filename) {
    if (!SPIFFS.exists(filename)) {
        return false;
    }

    setKey(AES_KEY);                                                        //Reset key
    _aesKeyString = AES_KEY;

    File inputFile = SPIFFS.open(filename, "r");
    File tmpFile = SPIFFS.open("tmp_" + filename, "w");

    /* If already is plaintext, don't do anything */
    if (inputFile.readStringUntil('\n') == "format: plaintext") {
        return false;
    }

    tmpFile.println("format: plaintext");                                   //Save format
    String line = inputFile.readStringUntil('\n');

    /* Encrypt the file line by line */
    while (line != "") {
        tmpFile.println(decryptLine(line));
        line = inputFile.readStringUntil('\n');                             //Read a line from the file
    }

    inputFile.close();
    tmpFile.close();

    SPIFFS.remove(filename);                                                //Remove plain text file
    SPIFFS.rename("tmp_" + filename, filename);                             //Rename tmp file
    return true;
}

/**************************************************************************/
/**************************************************************************/
bool CbcEncryptor::setKey(String key) {
    _aesKeyString = key;

    /* Check if key is the right length */
    if (key.length() != AES_KEY_SIZE) {
        return false;
    }

    /* Convert key string to bytes */
    for (uint8_t i = 0; i < AES_KEY_SIZE; i++) {
        _aesKey[i] = (byte) key[i];
    }
    return true;
}

```

```
/****************************************************************************/
/****************************************************************************/
String CbcEncryptor::encryptLine(String line) {
    uint16_t len = line.length();
    byte plain[len];                                                        //To store plain bytes
    uint8_t paddedLength = len + N_BLOCK - len % N_BLOCK;                   //Calculate total length when padded
    byte encrypted[paddedLength];                                           //To store encrypted bytes
    char encryptedString[paddedLength*2+1];                                 //To store return value

    /* Convert string to bytes */
    for (uint8_t i = 0; i < len; i++) {
        plain[i] = (byte) line[i];
    }

    _aes.set_IV(_aesInitVecInt);
    _aes.get_IV(_aesInitVector);
    _aes.do_aes_encrypt(plain, len, encrypted, _aesKey, 128, _aesInitVector);

    /* Convert bytes to string */
    for (uint8_t i = 0; i < paddedLength; i++) {
        sprintf(encryptedString+2*i,"%02x",*(encrypted+i));
    }

    return encryptedString;
}

/****************************************************************************/
/****************************************************************************/
String CbcEncryptor::decryptLine(String line) {
    uint16_t len = line.length()/2;
    byte encrypted[len];                                                    //To store encrypted bytes
    byte decrypted[len];                                                    //To store decrypted bytes
    uint8_t outputLen = 0;                                                  //To store plaintext length
    char decryptedString[len];                                              //To store return value

    /* Convert hex string to bytes */
    for (uint8_t i = 0; i < len; i++) {
        encrypted[i] = _hexCharToByte(line[i*2])«4 | _hexCharToByte(line[i*2+1]);
    }

    _aes.set_IV(_aesInitVecInt);
    _aes.get_IV(_aesInitVector);
    _aes.do_aes_decrypt(encrypted, len, decrypted, _aesKey, 128, _aesInitVector);

    /* Convert bytes to string */
    for (uint8_t i = 0; i < len; i++) {
        /* If decrypted is printable character, save */
        if (decrypted[i] < 127 && decrypted[i] > 32) {
            decryptedString[i] = (char) decrypted[i];
        } else {
            outputLen = i;
            break;
        }
    }

    if (_aesKeyString != AES_KEY) {
        return String(decryptedString);                                     //If another key is used, just throw the whole string back
    }
    return String(decryptedString).substring(0, outputLen);                 //Convert to string to trim the string with substring()
}

/****************************************************************************/
/****************************************************************************/
byte CbcEncryptor::_hexCharToByte(char hexChar) {
    uint8_t c = hexChar;

    if (c <= '9' && c >= '0') {
        c -= '0';
    } else if (c <= 'f' && c >= 'a') {
        c -= ('a' - 0x0a);
    } else if (c <= 'F' && c >= 'A') {
        c -= ('A' - 0x0a);
    } else {
        return(1);
    }

   return (c);
}
```

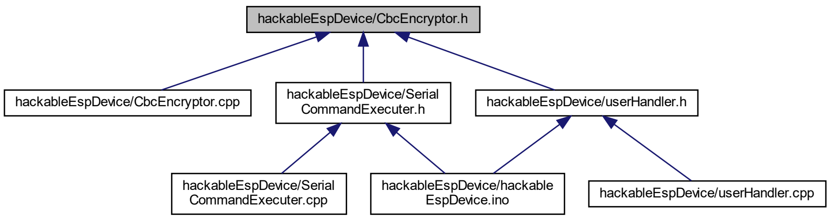
5.13 hackableEspDevice/CbcEncryptor.h File Reference

```
#include <FS.h>
#include "AES.h"
#include "config.h"
```

Include dependency graph for CbcEncryptor.h:



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


hackableEspDevice/CbcEncryptor.h
hackableEspDevice/CbcEncryptor.cpp
hackableEspDevice/Serial CommandExecuter.h
hackableEspDevice/userHandler.h
hackableEspDevice/Serial CommandExecuter.cpp
hackableEspDevice/hackable EspDevice.ino
hackableEspDevice/userHandler.cpp

Classes

- class CbcEncryptor

5.14 CbcEncryptor.h

Go to the documentation of this file.

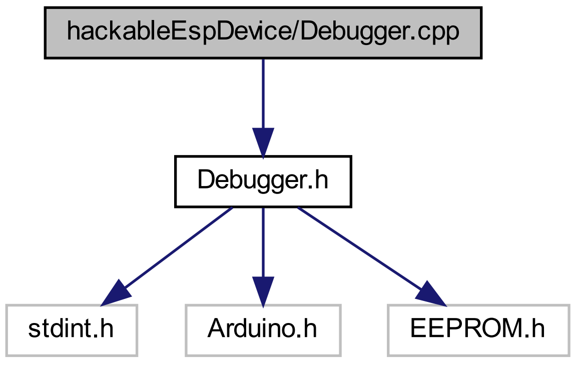
```
00001 /*
```

```
 * File:      CbcEncryptor.h
 * Author:    Luke de Munk
 * Class:     CbcEncryptor
 * Version:   1.0
 *
 * Handles encryption and decryption of files. Uses AES encryption.
 */
#ifndef ENCRYPTOR_H
#define ENCRYPTOR_H
#include <FS.h>                                                        //For SPIFFS
#include "AES.h"                                                       //For the encryption
#include "config.h"                                                    //For the configuration

class CbcEncryptor
{
    public:
        CbcEncryptor();
        bool encryptFile(String filename);
        bool decryptFile(String filename);
        String encryptLine(String line);
        String decryptLine(String line);
        bool setKey(String key);

    private:
        byte _hexCharToByte(char hexChar);

        String _aesKeyString;
        byte _aesKey[AES_KEY_SIZE];
        byte _aesInitVector[N_BLOCK];
        unsigned long long int _aesInitVecInt;
        AES _aes;
};
#endif
```

5.15 hackableEspDevice/Debugger.cpp File Reference

`#include "Debugger.h"`

Include dependency graph for Debugger.cpp:



Functions

- void debug (String text)

  Prints text if debug is enabled.
- void debugln (String text)

  Prints text (+'
  ') if debug is enabled.
- bool getDebugEnabled ()

  Gets if debug is enabled.
- void setDebugEnabled (bool isEnabled)

  Sets if debug is enabled.

5.15.1 Function Documentation

5.15.1.1 debug()

```
void debug (
             String text )
```

Prints text if debug is enabled.

Parameters

text	String of text that needs to be printed

Definition at line 16 of file Debugger.cpp.

5.15.1.2 debugln()

```
void debugln (
             String text )
```

Prints text (+'
') if debug is enabled.

Parameters

text	String of text that needs to be printed

Definition at line 32 of file Debugger.cpp.

5.15.1.3 getDebugEnabled()

```
bool getDebugEnabled ( )
```

Gets if debug is enabled.

Returns

bool True if debug is enabled

Definition at line 47 of file Debugger.cpp.

5.15.1.4 setDebugEnabled()

```
void setDebugEnabled (
            bool isEnabled )
```

Sets if debug is enabled.

Parameters

isEnabled	If debug is enabled (true == enabled)

Definition at line 60 of file Debugger.cpp.

5.16 Debugger.cpp

Go to the documentation of this file.

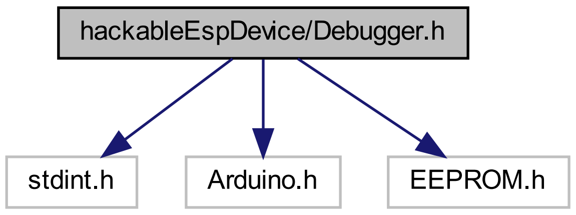
```
00001 /*
00002  * File:      Debugger.h
00003  * Author:    Luke de Munk
00004  * Version:   1.0
00005  *
00006  * Class for handling the debug prints.
00007  */
00008 #include "Debugger.h"
00009
00010 /**************************************************************************/
00015 /**************************************************************************/
00016 void debug(String text) {
00017     EEPROM.begin(1);
00018     bool isEnabled = EEPROM.read(ENABLE_DEBUG_FLAG_ADDRESS);
00019
00020     if(isEnabled) {
00021         Serial.print(text);
00022     }
00023     EEPROM.end();
00024 }
00025
00026 /**************************************************************************/
00031 /**************************************************************************/
00032 void debugln(String text) {
00033     EEPROM.begin(1);
00034     bool isEnabled = EEPROM.read(ENABLE_DEBUG_FLAG_ADDRESS);
00035     if(isEnabled) {
00036         Serial.println(text);
00037     }
00038     EEPROM.end();
00039 }
00040
00041 /**************************************************************************/
00046 /**************************************************************************/
00047 bool getDebugEnabled() {
00048     EEPROM.begin(1);
00049     bool isEnabled = EEPROM.read(ENABLE_DEBUG_FLAG_ADDRESS);
00050     EEPROM.end();
00051     return isEnabled;
00052 }
00053
00054 /**************************************************************************/
```

```
/****************************************************************************/
void setDebugEnabled(bool isEnabled) {
    EEPROM.begin(1);
    EEPROM.write(ENABLE_DEBUG_FLAG_ADDRESS, (uint8_t) isEnabled);            //Set the debug flag
    EEPROM.commit();                                                           //Write to EEPROM
    EEPROM.end();
}
```

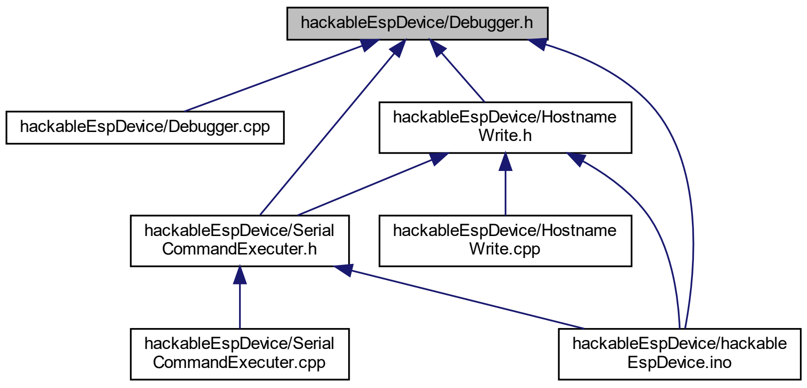
5.17 hackableEspDevice/Debugger.h File Reference

```
#include <stdint.h>
#include "Arduino.h"
#include <EEPROM.h>
```

Include dependency graph for Debugger.h:



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


hackableEspDevice/Debugger.h
hackableEspDevice/Debugger.cpp
hackableEspDevice/Hostname Write.h
hackableEspDevice/Serial CommandExecuter.h
hackableEspDevice/Hostname Write.cpp
hackableEspDevice/Serial CommandExecuter.cpp
hackableEspDevice/hackable EspDevice.ino

Macros

- #define ENABLE_DEBUG_FLAG_ADDRESS 0

Functions

- void debug (String text)

  Prints text if debug is enabled.
- void debugln (String text)

  Prints text (+'
  ') if debug is enabled.
- bool getDebugEnabled ()

  Gets if debug is enabled.
- void setDebugEnabled (bool isEnabled)

  Sets if debug is enabled.

5.17.1 Macro Definition Documentation

5.17.1.1 ENABLE_DEBUG_FLAG_ADDRESS

```
#define ENABLE_DEBUG_FLAG_ADDRESS 0
```

Definition at line 21 of file Debugger.h.

5.17.2 Function Documentation

5.17.2.1 debug()

```
void debug (
             String text )
```

Prints text if debug is enabled.

Parameters

text	String of text that needs to be printed

Definition at line 16 of file Debugger.cpp.

5.17.2.2 debugln()

```
void debugln (
             String text )
```

Prints text (+'
') if debug is enabled.

Parameters

text	String of text that needs to be printed

Definition at line 32 of file Debugger.cpp.

5.17.2.3 getDebugEnabled()

```
bool getDebugEnabled ( )
```

Gets if debug is enabled.

Returns

bool True if debug is enabled

Definition at line 47 of file Debugger.cpp.

5.17.2.4 setDebugEnabled()

```
void setDebugEnabled (
            bool isEnabled )
```

Sets if debug is enabled.

Parameters

isEnabled	If debug is enabled (true == enabled)

Definition at line 60 of file Debugger.cpp.

5.18 Debugger.h

Go to the documentation of this file.

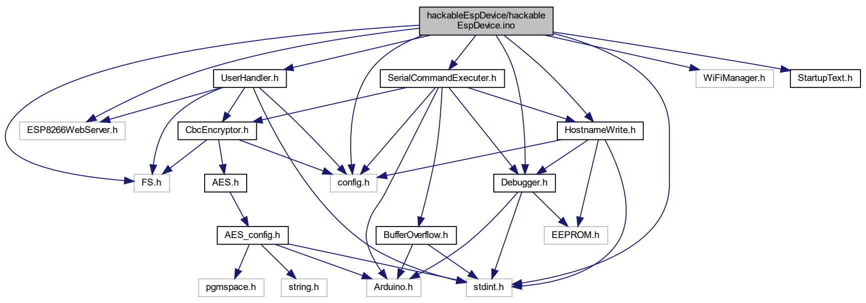
```
/*
 * File:      Debugger.h
 * Author:    Luke de Munk
 * Version:   1.0
 *
 * Class for handling the debug prints.
 */
#ifndef DEBUGGER_H
#define DEBUGGER_H
#include <stdint.h>                                                //For defining bits per integer
#include "Arduino.h"
#include <EEPROM.h>                                                //For reading from and writing to flash memory, used for resetting wifi
```

```

/*
 * 1 byte to store the enable debug flag.
 * Is done in EEPROM, because the
 * flag is then non-volatile and can
 * be used by multiple classes. Also
 * is saved during restart.
 */
#define ENABLE_DEBUG_FLAG_ADDRESS   0

void debug(String text);
void debugln(String text);
bool getDebugEnabled();
void setDebugEnabled(bool isEnabled);

#endif
```

5.19 hackableEspDevice/hackableEspDevice.ino File Reference

```
#include <ESP8266WebServer.h>
#include <FS.h>
#include <stdint.h>
#include <WiFiManager.h>
#include "config.h"
#include "UserHandler.h"
#include "SerialCommandExecuter.h"
#include "Debugger.h"
#include "HostnameWrite.h"
#include "StartupText.h"
```

Include dependency graph for hackableEspDevice.ino:



Macros

- #define ON HIGH
- #define OFF LOW
- #define MIN_BRIGHTNESS 1022

Functions

- ESP8266WebServer server (80)
- void setup ()

  Setup microchip.
- void initializeHostname ()

  Initializes hostname.

- void setupWifi ()

  Connects to WiFi if it can, otherwise starts as AP to configure WiFi.
- void initializeServer ()

  Takes care of the webservices like pageloading.
- void sendToFrontend (String var)

  Replaces placeholders with actual data in HTML page.
- void loop ()

  Mainloop.
- String getContentType (String filename)

  Converts the file extension to the MIME type.
- void handleFileRequest (String path, uint8_t permissionLevel)

  Sends the requested file if the user has permission.
- void handleFileUpload ()

  Handles the file upload to the SPIFFS.
- void handleFileDownload ()

  Handles the file download to the SPIFFS.

Variables

- uint8_t ledState = OFF
- uint16_t brightness = 1023
- UserHandler userHandler & server
- SerialCommandExecuter cliExecuter
- File fsUploadFile

5.19.1 Macro Definition Documentation

5.19.1.1 MIN_BRIGHTNESS

```
#define MIN_BRIGHTNESS 1022
```

Definition at line 24 of file hackableEspDevice.ino.

5.19.1.2 OFF

```
#define OFF LOW
```

Definition at line 22 of file hackableEspDevice.ino.

5.19.1.3 ON

```
#define ON HIGH
```

Definition at line 21 of file hackableEspDevice.ino.

5.19.2 Function Documentation

5.19.2.1 getContentType()

```
String getContentType (
             String filename )
```

Converts the file extension to the MIME type.

Parameters

filename	Name of the file

Returns

String MIME type of the file

Definition at line 304 of file hackableEspDevice.ino.

5.19.2.2 handleFileDownload()

```
void handleFileDownload ( )
```

Handles the file download to the SPIFFS.

Definition at line 388 of file hackableEspDevice.ino.

5.19.2.3 handleFileRequest()

```
void handleFileRequest (
             String path,
             uint8_t permissionLevel )
```

Sends the requested file if the user has permission.

Parameters

path	Path to the file
permissionLevel	0 = not logged in, 1 = user, 2 = admin

Definition at line 321 of file hackableEspDevice.ino.

5.19.2.4 handleFileUpload()

```
void handleFileUpload ( )
```

Handles the file upload to the SPIFFS.

Definition at line 353 of file hackableEspDevice.ino.

5.19.2.5 initializeHostname()

```
void initializeHostname ( )
```

Initializes hostname.

Definition at line 74 of file hackableEspDevice.ino.

5.19.2.6 initializeServer()

```
void initializeServer ( )
```

Takes care of the webservices like pageloading.

Definition at line 132 of file hackableEspDevice.ino.

5.19.2.7 loop()

```
void loop ( )
```

Mainloop.

Definition at line 289 of file hackableEspDevice.ino.

5.19.2.8 sendToFrontend()

```
void sendToFrontend (
             String var )
```

Replaces placeholders with actual data in HTML page.

Definition at line 276 of file hackableEspDevice.ino.

5.19.2.9 server()

```
ESP8266WebServer server (
             80  )
```

5.19.2.10 setup()

```
void setup ( )
```

Setup microchip.

Definition at line 40 of file hackableEspDevice.ino.

5.19.2.11 setupWifi()

```
void setupWifi ( )
```

Connects to WiFi if it can, otherwise starts as AP to configure WiFi.

Definition at line 105 of file hackableEspDevice.ino.

5.19.3 Variable Documentation

5.19.3.1 brightness

```
uint16_t brightness = 1023
```

Definition at line 28 of file hackableEspDevice.ino.

5.19.3.2 cliExecuter

```
SerialCommandExecuter cliExecuter
```

Definition at line 31 of file hackableEspDevice.ino.

5.19.3.3 fsUploadFile

```
File fsUploadFile
```

Definition at line 33 of file hackableEspDevice.ino.

5.19.3.4 ledState

```
uint8_t ledState = OFF
```

Definition at line 27 of file hackableEspDevice.ino.

5.19.3.5 server

```
UserHandler userHandler& server
```

Definition at line 30 of file hackableEspDevice.ino.

5.20 hackableEspDevice.ino

Go to the documentation of this file.

```
/*
 * File:      hackableEspDevice.ino
 * Authors:   ESPinoza (Team 1)
 * Version:   1.0
 *
 * The main file of the firmware of a vunerable-by-design ESP8266 controller.
 * For more information, go to: https://gitlab.fdmci.hva.nl/munkl/hackable_esp_device
 *
 */
#include <ESP8266WebServer.h>                                                  //For running the
 webserver
#include <FS.h>                                                                //For SPIFFS
#include <stdint.h>                                                            //For defining bits per
 integer
#include <WiFiManager.h>                                                       //For web-based wifi
 configuration
#include "config.h"                                                            //For the configuration
#include "UserHandler.h"                                                       //For handling the users
 from the config.conf
#include "SerialCommandExecuter.h"                                             //For handling serial
 commands
#include "Debugger.h"                                                          //For handling debug
 messages
#include "HostnameWrite.h"                                                     //For handling the
 hostname changes
```

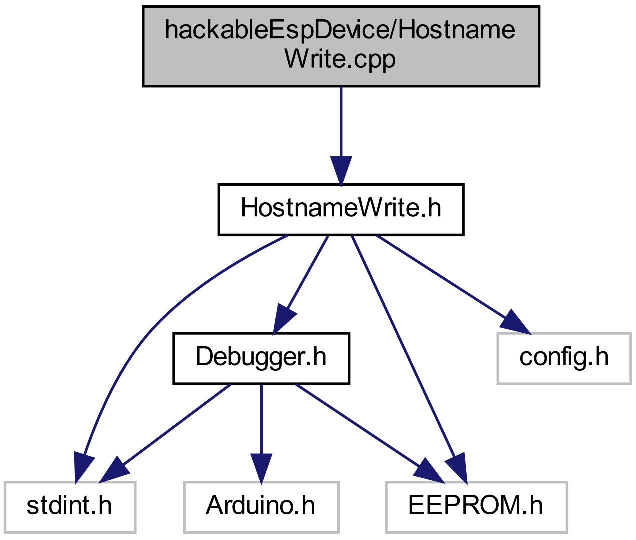
```
#include "StartupText.h"                                         //For printing startup log files

#define ON                      HIGH
#define OFF                     LOW

#define MIN_BRIGHTNESS          1022                             //analogWrite() on ESP8266 D1 Mini board is inverted

ESP8266WebServer server(80);                                     //Object that listens for HTTP requests on port 80
uint8_t ledState = OFF;                                          //For led state variable
uint16_t brightness = 1023;                                      //For led brightness

UserHandler userHandler(&server);                                //For handling the authentication
SerialCommandExecuter cliExecuter;                               //For handling serial commands

File fsUploadFile;                                               //A File object to temporarily store the received file

/**************************************************************************/
/**************************************************************************/
void setup() {
    Serial.begin(115200);                                        //Serial port for debugging purposes

    /* Initialize SPIFFS */
    if (!SPIFFS.begin()) {
        Serial.println("An Error has occurred while mounting SPIFFS");
        return;
    }

    debugln("Debug is enabled");

    /* If debug is enabled, the root password is printed in a big string of text */
    if (getDebugEnabled()) {
      String mess = "ROOT: " + String(ROOT_PASSWORD);
      printStartupText(mess);
    }

    pinMode(LED_BUILTIN, OUTPUT);
    analogWrite(LED_BUILTIN, 1023);

    initializeHostname();
    setupWifi();
    initializeServer();
    userHandler.updateUsers();
    cliExecuter.setUsers(userHandler.getUsers(), userHandler.getNumberOfUsers()); //Send users to the command executer for the 'users' command

    Serial.println("Serial commands available. Typ 'help' for help.");
}

/**************************************************************************/
/**************************************************************************/
void initializeHostname() {
    String customHostname = getHostname();
    /* Check if custom hostname is set, otherwise use default */
    if (customHostname != "") {
        /* Check if hostname can be set */
        if (WiFi.hostname(customHostname)){
            debug(customHostname);
            debugln(" is the hostname.");
        } else {
            debug("Could not set '");
            debug(customHostname);
            debugln("' as hostname.");
        }
    } else {
        if (WiFi.hostname(DEFAULT_HOSTNAME)) {
            debug(DEFAULT_HOSTNAME);
            debugln(" is the hostname.");
        } else {
            debug("Could not set '");
            debug(DEFAULT_HOSTNAME);
            debugln("' as hostname.");
        }
    }
}

/**************************************************************************/
/**************************************************************************/
void setupWifi() {
    WiFiManager wifiManager;

    if (wifiManager.autoConnect(WIFI_CONF_AP_NAME)) {
        Serial.print("Connected to: ");
        Serial.println(WiFi.SSID());
        Serial.print("IP: ");
        Serial.println(WiFi.localIP());
    } else {
        Serial.println("Failed to connect, connect with AP");
        ESP.restart();
    }

    debug("Copy and paste the following URL: http://");

    if (WiFi.hostname(DEFAULT_HOSTNAME)) {
        debugln(DEFAULT_HOSTNAME);
    } else {
        debugln(WiFi.hostname().c_str());
    }
}

/****************************************************************************/
/****************************************************************************/
void initializeServer() {
    /*
    *  Routes for loading all the necessary files
    */
    /* Route for home page */
    server.on("/", HTTP_GET, []() {
        handleFileRequest("/index.html", PERMISSION_LVL_ALL);
    });

    /* Route for sending c++ variables */
    server.on("/state", HTTP_GET, []() {
        sendToFrontend("ledState");
    });

    /* Route for sending c++ variables */
    server.on("/brightness", HTTP_GET, []() {
        sendToFrontend("brightness");
    });

    /* Route for admin controls */
    server.on("/admin", HTTP_GET, []() {
        handleFileRequest("/admin.html", PERMISSION_LVL_ADMIN);
    });

    /* Route for user controls */
    server.on("/user", HTTP_GET, []() {
        handleFileRequest("/user.html", PERMISSION_LVL_USER);
    });

    /* Route for file upload page */
    server.on("/upload", HTTP_GET, []() {
        handleFileRequest("/upload.html", PERMISSION_LVL_ADMIN);
    });

    /* Route for file download page */
    server.on("/download", HTTP_GET, []() {
        handleFileRequest("/download.html", PERMISSION_LVL_USER);
    });

    /* Load styles.css file, styling for desktop version */
    server.on("/styles.css", HTTP_GET, []() {
        handleFileRequest("/styles.css", PERMISSION_LVL_ALL);
    });

    /* Load styles_mobile.css file, styling for mobile version */
    server.on("/styles_mobile.css", HTTP_GET, []() {
        handleFileRequest("/styles_mobile.css", PERMISSION_LVL_ALL);
    });

    /* Load style_switch.css file, styling for the on/off switch */
    server.on("/style_switch.css", HTTP_GET, []() {
        handleFileRequest("/style_switch.css", PERMISSION_LVL_ALL);
    });

    /* Load favicon.ico file, site icon */
    server.on("/favicon.ico", HTTP_GET, []() {
        handleFileRequest("/favicon.ico", PERMISSION_LVL_ALL);
    });

    /* Load jquery.min.js file, for ajax */
    server.on("/jquery.min.js", HTTP_GET, []() {
        handleFileRequest("/jquery.min.js", PERMISSION_LVL_ALL);
    });

    /* Load base.js file, JavaScript for site */
    server.on("/base.js", HTTP_GET, []() {
        handleFileRequest("/base.js", PERMISSION_LVL_ALL);
    });

    /* Load switch.js file, JavaScript for on/off switch */
    server.on("/switch.js", HTTP_GET, []() {
        handleFileRequest("/switch.js", PERMISSION_LVL_ALL);
    });
    /*
    * End of file loading
    */

    /*
    * Routes for JavaScript data receiving
    */
    /* Route for setting power */
    server.on("/set_power", HTTP_GET, []() {
        if (server.arg("state")) {
            ledState = atoi(server.arg("state").c_str());
            if(ledState == ON) {
                analogWrite(LED_BUILTIN, MIN_BRIGHTNESS-brightness);
            } else {
                analogWrite(LED_BUILTIN, 1023);
            }
        }
        handleFileRequest("/index.html", PERMISSION_LVL_ALL);
    });

    /* Route for setting brightness */
    server.on("/set_brightness", HTTP_GET, []() {
        if (server.arg("brightness")) {
            brightness = atoi(server.arg("brightness").c_str());
            if(ledState == ON) {
                analogWrite(LED_BUILTIN, MIN_BRIGHTNESS-brightness);
            }
        }
        handleFileRequest("/index.html", PERMISSION_LVL_ALL);
    });


    /* Route for restarting the server */
    server.on("/restart", HTTP_GET, []() {
        handleFileRequest("/", PERMISSION_LVL_ALL);
        ESP.restart();
    });
    /*
    * End of JavaScript data receiving
    */

    /*
    * Routes for file management
    */
    /* Route for file upload request */
    server.on("/upload", HTTP_POST, []() {
        server.send(200);                                                   //HTTP code 200 == OK
        debugln("Wait, something got uploaded");
      }, handleFileUpload                                                   //Receive and save the file
    );
    /* Route for file upload request */
    server.on("/download", HTTP_POST, []() {
        debugln("File download request");
      }, handleFileDownload                                                 //Receive and save the file
    );
    /*
    * End of routes for file management
    */
    /* Not found */
    server.onNotFound([]() {                                                //If the client requests any URI
        handleFileRequest(server.uri(), PERMISSION_LVL_ALL);                //Send it if it exists
        debugln("Route not found");
    });
    server.begin();                                                         //Start server
}

/**************************************************************************/
/**************************************************************************/
void sendToFrontend(String var){
    if (var == "ledState") {
        server.send(200, "text/plain", String (ledState));
    } else if (var == "brightness") {
        server.send(200, "text/plain", String (brightness));
    }
}

/**************************************************************************/
```

```
/*****************************************************************************/
void loop() {
  server.handleClient();

  if (Serial.available()) {
    cliExecuter.executeCommand();
  }
}

/*****************************************************************************/
/*****************************************************************************/
String getContentType(String filename) {
  if (filename.endsWith(".html")) return "text/html";
  else if (filename.endsWith(".css")) return "text/css";
  else if (filename.endsWith(".js")) return "application/javascript";
  else if (filename.endsWith(".ico")) return "image/x-icon";
  else if (filename.endsWith(".gz")) return "application/x-gzip";
  else if (filename.endsWith(".txt")) return "text/plain";
  return "text/plain";
}

/*****************************************************************************/
/*****************************************************************************/
void handleFileRequest(String path, uint8_t permissionLevel) {
    if(!userHandler.checkPermission(permissionLevel, &server)) {
        server.requestAuthentication();
        return;
    }

    debugln(String("Requested file: ") + path);

    String contentType = getContentType(path);                              //Get the MIME type
    String pathWithGz = path + ".gz";

    if (SPIFFS.exists(pathWithGz)) {                                        //If there's a compressed version available
        path += ".gz";                                                      //Use the compressed verion
    }

    if (SPIFFS.exists(path)) {
        File file = SPIFFS.open(path, "r");                                 //Open the file
        size_t sent = server.streamFile(file, contentType);                 //Send it to the client
        file.close();                                                       //Close the file again
        debugln(String("Sent file: ") + path);
        return;
    }

    debugln(String("File Not Found: ") + path);                             //If the file doesn't exist, return false
    server.send(404, "text/plain", "404: Not Found");                       //otherwise, respond with a 404 (Not Found) error
}

/*****************************************************************************/
/*****************************************************************************/
void handleFileUpload() {
    HTTPUpload& upload = server.upload();

    if (upload.status == UPLOAD_FILE_START) {
        String filename = upload.filename;

        if (!filename.startsWith("/")) {
            filename = "/" + filename;
        }

        debugln(String("Upload file named: ") + filename);

        fsUploadFile = SPIFFS.open(filename, "w");                          //Open the file for writing in SPIFFS (create if it doesn't exist)

    } else if (upload.status == UPLOAD_FILE_WRITE && fsUploadFile ) {
        fsUploadFile.write(upload.buf, upload.currentSize);                 //Write the received bytes to the file
    } else if (upload.status == UPLOAD_FILE_END) {
        if (fsUploadFile) {                                                 //If the file was successfully created
            fsUploadFile.close();                                           //Close the file again
            debugln(String("handleFileUpload Size: ") + upload.totalSize);
            server.sendHeader("Location","/success.html");                  //Redirect the client to the success page
            server.send(303);
            userHandler.updateUsers();
            cliExecuter.setUsers(userHandler.getUsers(), userHandler.getNumberOfUsers()); //Update users for cli as well
        } else {
            server.send(500, "text/plain", "500: couldn't create file");
        }
    }
}

/***************************************************************************/
/***************************************************************************/
void handleFileDownload() {
    String filename = server.arg("filekey");                           //Get user input for
 filename

    if (!filename.startsWith("/")) {
        filename = "/" + filename;
    }

    if (!SPIFFS.exists(filename)) {
        server.send(404, "text/plain", "404: file not found!");
        return;
    }

    File download = SPIFFS.open(filename, "r");

    debugln("Start sending file");

    server.sendHeader("Content-Type", "text/text");
    server.sendHeader("Content-Disposition", "attachment; filename="+filename);
    server.sendHeader("Connection", "close");
    server.streamFile(download, "application/octet-stream");
    download.close();
    server.send(200);                                                  //HTTP code 200 == OK
}
```

5.21 hackableEspDevice/HostnameWrite.cpp File Reference

`#include "HostnameWrite.h"`

Include dependency graph for HostnameWrite.cpp:



Functions

- String getHostname ()

Gets the hostname from the EEPROM.
- void writeHostname (char hostname[MAX_HOSTNAME_LENGTH])

  Writes the new hostname to the EEPROM.
- void setEepromToNull (uint8_t writeLength, uint8_t startAdress)

  Resets the given EEPROM adresses.
- void checkEepromCommit ()

  Checks if the eeprom was actually committed.

5.21.1 Function Documentation

5.21.1.1 checkEepromCommit()

```
void checkEepromCommit ( )
```

Checks if the eeprom was actually committed.

Definition at line 79 of file HostnameWrite.cpp.

5.21.1.2 getHostname()

```
String getHostname ( )
```

Gets the hostname from the EEPROM.

Returns

String Current hostname

Definition at line 16 of file HostnameWrite.cpp.

5.21.1.3 setEepromToNull()

```
void setEepromToNull (
            uint8_t writeLength,
            uint8_t startAdress )
```

Resets the given EEPROM adresses.

Parameters

writeLength	Length to be erased
startAdress	Start adress

Definition at line 56 of file HostnameWrite.cpp.

5.21.1.4 writeHostname()

```
void writeHostname (
              char hostname[MAX_HOSTNAME_LENGTH] )
```

Writes the new hostname to the EEPROM.

Parameters

hostname	String that contains the hostname to be written

Definition at line 37 of file HostnameWrite.cpp.

5.22 HostnameWrite.cpp

Go to the documentation of this file.

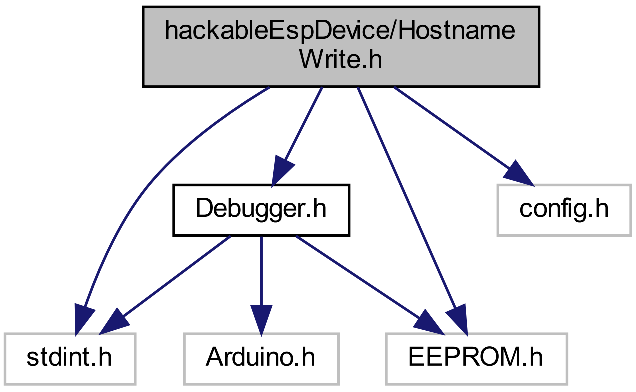
```
00001 /*
00002  * File:       HostnameWrite.cpp
00003  * Author:     Twenne Elffers
00004  * Version:    1.0
00005  *
00006  * Handles hostname saving and reading to EEPROM.
00007  */
00008 #include "HostnameWrite.h"
00009
00010 /**************************************************************************/
00015 /**************************************************************************/
00016 String getHostname() {
00017     char hostname[MAX_HOSTNAME_LENGTH];
00018     EEPROM.begin(MAX_HOSTNAME_LENGTH);
00019
00020     for (uint8_t i = 0; i < MAX_HOSTNAME_LENGTH; i++) {
00021         EEPROM.get(HOSTNAME_ADRESS+i, hostname[i]);
00022         if (hostname[i] == 0xFF) {
00023             break;                                              //Skips the unreadable
       chars
00024             }
00025       }
00026
00027     EEPROM.end();
00028     return String(hostname);
00029 }
00030
00031 /**************************************************************************/
00036 /**************************************************************************/
00037 void writeHostname(char hostname[MAX_HOSTNAME_LENGTH]) {
00038     EEPROM.begin(MAX_HOSTNAME_LENGTH);
00039
00040     for (uint8_t i = 0; i< MAX_HOSTNAME_LENGTH; i++){
00041         EEPROM.write(HOSTNAME_ADRESS+i, hostname[i]);
00042         yield();
00043     }
00044
00045     checkEepromCommit();
00046     EEPROM.end();
00047 }
00048
00049 /**************************************************************************/
00055 /**************************************************************************/
00056 void setEepromToNull(uint8_t writeLength, uint8_t startAdress) {
00057     EEPROM.begin(writeLength);
00058
00059     for (uint8_t i = 0; i < writeLength; i++){
00060         EEPROM.write(startAdress+i, 0);
00061         yield();
```

```
    }

    checkEepromCommit();

    debug("Reset Value at: ");
    debug(String(startAdress));
    debug(" till ");
    debugln(String(startAdress+writeLength));

    EEPROM.end();
}

/*****************************************************************************/
/*****************************************************************************/
void checkEepromCommit() {
    if (EEPROM.commit()) {
        Serial.println("Data written!");
    } else {
        Serial.println("ERROR! Data not written!");
    }
}
```

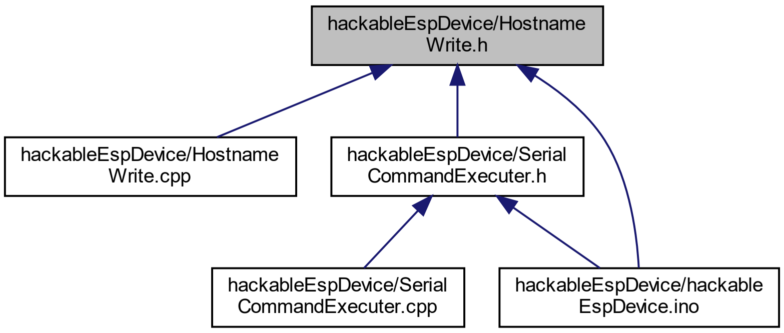
5.23 hackableEspDevice/HostnameWrite.h File Reference

```
#include <stdint.h>
#include <EEPROM.h>
#include "Debugger.h"
#include "config.h"
```

Include dependency graph for HostnameWrite.h:



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


hackableEspDevice/Hostname Write.h
hackableEspDevice/Hostname Write.cpp
hackableEspDevice/Serial CommandExecuter.h
hackableEspDevice/Serial CommandExecuter.cpp
hackableEspDevice/hackable EspDevice.ino

Functions

- String getHostname ()

  Gets the hostname from the EEPROM.
- void writeHostname (char hostname[32])
- void setEepromToNull (uint8_t writeLength, uint8_t startAdress)

  Resets the given EEPROM adresses.
- void checkEepromCommit ()

  Checks if the eeprom was actually committed.

5.23.1 Function Documentation

5.23.1.1 checkEepromCommit()

```
void checkEepromCommit ( )
```

Checks if the eeprom was actually committed.

Definition at line 79 of file HostnameWrite.cpp.

5.23.1.2 getHostname()

```
String getHostname ( )
```

Gets the hostname from the EEPROM.

Returns

String Current hostname

Definition at line 16 of file HostnameWrite.cpp.

5.23.1.3 setEepromToNull()

```
void setEepromToNull (
            uint8_t writeLength,
            uint8_t startAdress )
```

Resets the given EEPROM adresses.

Parameters

writeLength	Length to be erased
startAdress	Start adress

Definition at line 56 of file HostnameWrite.cpp.

5.23.1.4 writeHostname()

```
void writeHostname (
            char hostname[32] )
```

5.24 HostnameWrite.h

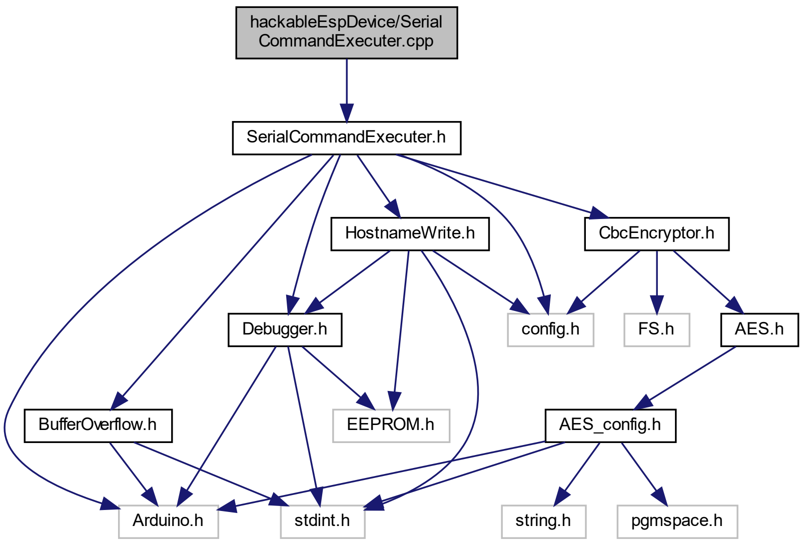
Go to the documentation of this file.

```
/*
 * File:      HostnameWrite.h
 * Author:    Twenne Elffers
 * Version:   1.0
 *
 * Handles hostname saving and reading to EEPROM.
 */
#ifndef HOSTNAME_WRITE_H
#define HOSTNAME_WRITE_H
#include <stdint.h>                                                   //For defining bits per
 integer
#include <EEPROM.h>                                                   //For reading from and
 writing to EEPROM
#include "Debugger.h"                                                 //For handling debug
 messages
#include "config.h"                                                   //For the configuration

String getHostname();
void writeHostname(char hostname[32]);
void setEepromToNull(uint8_t writeLength, uint8_t startAdress);
void checkEepromCommit();
#endif
```

5.25 hackableEspDevice/SerialCommandExecuter.cpp File Reference

`#include "SerialCommandExecuter.h"`

Include dependency graph for SerialCommandExecuter.cpp:



5.26 SerialCommandExecuter.cpp

Go to the documentation of this file.

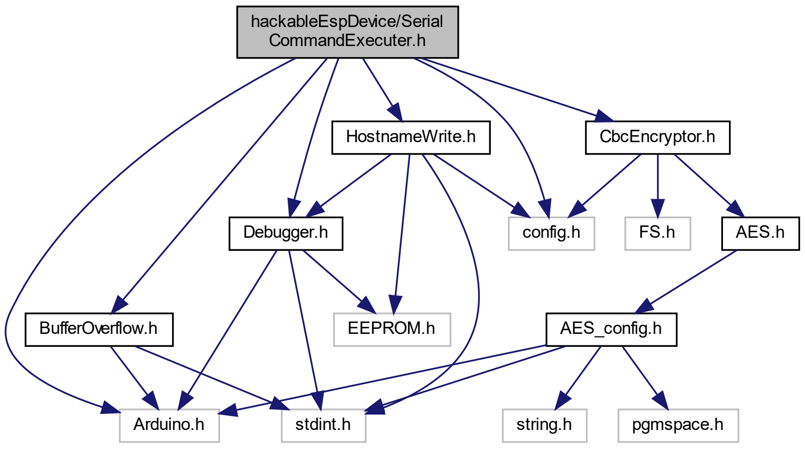
```
/*
 * File:      SerialCommandExecuter.cpp
 * Author:    Luke de Munk & Twenne Elffers
 * Class:     SerialCommandExecuter
 * Version:   1.0
 *
 * Parses and executes serial terminal commands.
 */
#include "SerialCommandExecuter.h"

/**************************************************************************/
/**************************************************************************/
SerialCommandExecuter::SerialCommandExecuter() {
    _isLoggedIn = false;
}

/**************************************************************************/
/**************************************************************************/
void SerialCommandExecuter::setUsers(String* users, uint8_t numUsers) {
    /* Copy users */
    for (uint8_t i = 0; i < MAX_NUMBER_USERS*USER_INFO_LENGTH; i++) {
        _users[i] = users[i];
    }
    _numberUsers = numUsers;
}

/**************************************************************************/
/**************************************************************************/
void SerialCommandExecuter::executeCommand() {
    String command = Serial.readString();

    if (command != "") {
        if (_isLoggedIn) {
            Serial.print("~# ");                                        //For the Linux feeling,
 superuser
        } else {
            Serial.print("~$ ");                                           //For the Linux feeling, no superuser
        }
        Serial.print(command);                                             //Echo command (command ends with \n)

        _parseCommand(command);
    }
}

/***************************************************************************/
/***************************************************************************/
bool SerialCommandExecuter::_parseCommand(String commandString) {
    String* trimmedCmdLine = _trimCommand(commandString);
    String command = trimmedCmdLine[0].c_str();
    String params[MAX_NUMBER_PARAMS] = {""};
    uint8_t numParams = 0;

    while (numParams < MAX_NUMBER_PARAMS) {
        if (trimmedCmdLine[numParams+1] == "") {                           //+1, because the command is in the first cell
            break;
        }
        numParams++;
    }

    for (uint8_t i = 1; i-1 < numParams; i++){                             //+1, because the command is in the first cell
        params[i-1] = trimmedCmdLine[i].c_str();
    }

    /* Check which command is given */
    if (command == COMMAND_HELP) {
        _printHelp(COMMAND_HELP);
        return true;
    } else {
        /* If help needs to be printed, print it and return */
        if (_checkHelp(params[0], command)) {
            return true;
        }
    }

    if (command == COMMAND_DEBUG) {
        if (!_checkParams(numParams, 1, 1) || !_enableDebug(params[0])) {
            return false;
        }
    } else if (command == COMMAND_SU) {
        if (!_checkParams(numParams, 1, 1) || !_superUserLogin(params[0])) {
            return false;
        }
    } else if (command == COMMAND_KEYS) {
        if (!_viewKey()) {
            return false;
        }
    } else if ((command == COMMAND_RESTART)) {
        _restart();
    } else if (command == COMMAND_USERS) {
        if (!_viewUsers()) {
            return false;
        }
    } else if (command == COMMAND_HOSTNAME) {
        if (!_checkParams(numParams, 0, 2) || !_hostname(params)) {
            return false;
        }
    } else if (command == COMMAND_WHOAMI) {
        if (_isLoggedIn) {
            Serial.println("superuser");
        } else {
            Serial.println("user");
        }
        return true;
    } else if (command == COMMAND_LS) {
        buffOverflow.ls();
    } else if (command == COMMAND_VI) {
        if (_checkParams(numParams, 1, 1)) {
            if (params[0] == ARG_LS_FILE_1_1 || params[0] == ARG_LS_FILE_1_2) {
                buffOverflow.vi();
            } else {
                Serial.println(ERROR_NO_FILE);
                return false;
            }
        }
    } else if (command.substring(0, 2) == COMMAND_RUN) {                  //Substring == "./" the rest is filename
        if (_checkParams(numParams, 0, 1)) {
            String filename = command.substring(2);                        //The rest of the command is filename

            if (filename == ARG_LS_FILE_1_1) {
                Serial.println(ERROR_PERM_DENIED);
                return false;
            }

            if (filename != ARG_LS_FILE_2_1) {
                Serial.println(ERROR_NO_FILE_DIR);
                return false;
            }

            if (numParams == 1) {
                /* If buffer overflow is done correctly,
                 * user is logged in.
                 */
                if (buffOverflow.runCProgram(params[0])) {
                    _isLoggedIn = true;
                    Serial.println(MESS_SUPER_USER);
                }
            } else {
                buffOverflow.runCProgram("");
            }
        }
    } else if (command == COMMAND_OBJDUMP) {
        if (_checkParams(numParams, 2, 2)) {
            if (params[0] != "-d") {
                Serial.println(ERROR_WRONG_ARGS);
                return false;
            }
            if (params[1] == ARG_LS_FILE_2_1 || params[1] == ARG_LS_FILE_2_2) {
                buffOverflow.objectDump();
            } else {
                Serial.println(ERROR_NO_FILE);
                return false;
            }
        }
    } else if (command == COMMAND_GPG) {
        if (_checkParams(numParams, 3, 3)) {
            if (params[0] == ARG_GPG_ENCRYPT) {
              _encrypt(params);
            } else if (params[0] == ARG_GPG_DECRYPT) {
              _decrypt(params);
            } else {
                Serial.println(ERROR_WRONG_ARGS);
                return false;
            }
        }
    } else {
        Serial.println(ERROR_CMD_NOT_FOUND);
        return false;
    }
    return true;
}

/***************************************************************************/
/***************************************************************************/
String* SerialCommandExecuter::_trimCommand(String commandString) {
    static String commandItems[1+MAX_NUMBER_PARAMS] = {""};                  //To save command and
 parameters, each in own cell
    String item = "";                                                        //Can be a command or
 parameter
    uint8_t paramCounter = 0;

    /* Reset static array */
    for (uint16_t x = 0; x < 1+MAX_NUMBER_PARAMS; x++) {
        commandItems[x] = "";
    }

    /* Count number of parameters by adding to temp variable if not a whitespace or end of line*/
    for (uint16_t c = 0; c < commandString.length(); c++) {
        if (commandString[c] == ' ' || commandString[c] == '\n') {
            /* If item is not empty: add to item array */
            if (item != ""){
                commandItems[paramCounter] = item;                           //Save param to items list
                item = "";                                                   //Reset item value
                paramCounter++;
           }
        } else {
            item += commandString[c];
        }
    }
    return commandItems;
}

/***************************************************************************/
/***************************************************************************/
bool SerialCommandExecuter::_checkParams(uint8_t numParams, uint8_t minNumberParams, uint8_t
maxNumberParams) {
    if (numParams < minNumberParams) {
        Serial.println(ERROR_TOO_FEW_ARGS);
        return false;
    } else if (numParams > maxNumberParams) {
        Serial.println(ERROR_TOO_MANY_ARGS);
        return false;
    }
    return true;
}

/**************************************************************************/
/**************************************************************************/
void SerialCommandExecuter::_printHelp(String command) {
    /* Print help lines according to command */
    if (command == "" || command == COMMAND_HELP) {                         //Default help
        Serial.println("|---------------------------HELP---------------------------|");
        Serial.println("This is a commandline interface that allows access to the smartlight config");
        _printCommands();
    } else if (command == COMMAND_DEBUG) {
        Serial.println("Usage: debug [--off]                   Turns the debug off");
        Serial.println("       debug [--on]                    Turns the debug on");
    } else if (command == COMMAND_SU) {
        Serial.println("Usage: su {passwd}                     Login as superuser");
    } else if (command == COMMAND_KEYS) {
        Serial.println("Usage: privatekeys                     Shows private encryption keys");
    } else if (command == COMMAND_RESTART) {
        Serial.println("Usage: reboot                          Reboots the device");
    } else if (command == COMMAND_USERS) {
        Serial.println("Usage: users                           Shows usertable of website");
    } else if (command == COMMAND_HOSTNAME) {
        Serial.println("Usage: hostname                        Gives the current hostname");
        Serial.println("       hostname [--set] {newhostname}   Set new hostname. (needs reboot)");
        Serial.println("       hostname [--default]             Sets the hostname to the default hostname");
    } else if (command == COMMAND_LS) {
        Serial.println("Usage: ls                              Shows files in current folder");
    } else if (command == COMMAND_VI) {
        Serial.println("Usage: vi {filename}                   Opens file in text editor");
    }else if (command == COMMAND_RUN) {
        Serial.println("Usage: ./{filename}                    Runs an executable file");
    } else if (command == COMMAND_OBJDUMP) {
        Serial.println("Usage: objdump -d {filename}           Prints disassembled code of an executable file");
    } else if (command == COMMAND_GPG) {
        Serial.println("Usage: gpg --encrypt {key} {line}      Prints disassembled code of an executable file");
        Serial.println("Usage: gpg --decrypt {key} {line}      Prints disassembled code of an executable file");
    } else {
        Serial.println(ERROR_CMD_NOT_FOUND);
    }
}

/**************************************************************************/
/**************************************************************************/
void SerialCommandExecuter::_printCommands() {
    Serial.println("Available commands:");
    Serial.println(COMMAND_HELP);
    Serial.println(COMMAND_DEBUG);
    Serial.println(COMMAND_SU);
    Serial.println(COMMAND_KEYS);
    Serial.println(COMMAND_RESTART);
    Serial.println(COMMAND_USERS);
    Serial.println(COMMAND_HOSTNAME);
    Serial.println(COMMAND_LS);
    Serial.println(COMMAND_VI);
    Serial.println(COMMAND_OBJDUMP);
    Serial.println(COMMAND_WHOAMI);
    Serial.println(COMMAND_GPG);
}

/**************************************************************************/
/**************************************************************************/
bool SerialCommandExecuter::_enableDebug(String enable) {
    if (enable == ARG_DEBUG_ON) {
        setDebugEnabled(true);
        Serial.println("debug = true");
    } else if (enable == ARG_DEBUG_OFF) {
        setDebugEnabled(false);
        Serial.println("debug = false");
    } else {
        Serial.println(ERROR_WRONG_ARGS);
        return false;
    }
    return true;
}
```

```

/***************************************************************************/
/***************************************************************************/
bool SerialCommandExecuter::_superUserLogin(String password) {
    if (password == ROOT_PASSWORD) {
        _isLoggedIn = true;
        Serial.println(MESS_SUPER_USER);
    } else {
        Serial.println(ERROR_WRONG_PWD);
        return false;
    }
    return true;
}

/***************************************************************************/
/***************************************************************************/
bool SerialCommandExecuter::_viewKey() {
    if (!_isLoggedIn) {
        Serial.println(ERROR_NO_PERMISSION);
        return false;
    }
    Serial.println("Private encryption keys (Don't share!!!):");
    Serial.println(AES_KEY);
    return true;
}

/***************************************************************************/
/***************************************************************************/
void SerialCommandExecuter::_restart() {
    Serial.print("Restarting in ");

    /* Wait 3 seconds */
    for (uint8_t s = 3; s > 0; s--) {
        Serial.print(s);
        Serial.print(" ");
        delay(1000);
    }
    ESP.restart();
}

/***************************************************************************/
/***************************************************************************/
bool SerialCommandExecuter::_viewUsers() {
    String userPrints[USER_INFO_LENGTH] = {""};

    if (!_isLoggedIn) {
        Serial.println(ERROR_NO_PERMISSION);
        return false;
    }

    Serial.println("|-USERNAME-------|-PASSWORD-------|-ROLE---|");

    for (uint8_t i = 0; i < _numberUsers; i += 3) {
        userPrints[0] = _users[i].c_str();                                  //Username
        if(atoi(_users[i+2].c_str()) == PERMISSION_LVL_USER) {
            userPrints[1] = _users[i+1].c_str();                            //Password
            userPrints[2] = "User";                                         //Permission level/role
        } else if (atoi(_users[i+2].c_str()) == PERMISSION_LVL_ADMIN) {
            userPrints[1] = "******";                                       //Password, not printed
            userPrints[2] = "Admin";                                        //Permission level/role
        }
        Serial.printf("| %-15s| %-15s| %-7s|\n", userPrints[0].c_str(), userPrints[1].c_str(), userPrints[2].c_str());
    }
    return true;
}

/***************************************************************************/
/***************************************************************************/
bool SerialCommandExecuter::_hostname(String* params) {
    uint8_t numParams = params->length();
    if (numParams == 0) {                                                   //If empty: show hostname
        Serial.print("Hostname is: ");
        Serial.println(String(getHostname()));
        return true;
    }

    if (params[0] ==  ARG_HOSTNAME_SET && params[1] != "") {                //If parameter 'set' check if next value is not empty
        char newHostname[MAX_HOSTNAME_LENGTH];
        params[1].toCharArray(newHostname, MAX_HOSTNAME_LENGTH);
        writeHostname(newHostname);
    } else if (params[0] == ARG_HOSTNAME_DEFAULT) {
        writeHostname(DEFAULT_HOSTNAME);
    } else {
        Serial.println(ERROR_WRONG_ARGS);                                   //If it can't find suitable params: give error
        return false;
    }
    return true;
}


/***************************************************************************/
/***************************************************************************/
bool SerialCommandExecuter::_encrypt(String* params) {
    if (!cryptor.setKey(params[1])) {
        Serial.println(ERROR_NO_VALID_KEY);
        return false;
    }

    Serial.print("Encrypted output: ");
    Serial.println(cryptor.encryptLine(params[2]));
    return true;
}

/***************************************************************************/
/***************************************************************************/
bool SerialCommandExecuter::_decrypt(String* params) {
    if (!cryptor.setKey(params[1])) {
        Serial.println(ERROR_NO_VALID_KEY);
        return false;
    }

    Serial.print("Decrypted output: ");
    Serial.println(cryptor.decryptLine(params[2]));
    return true;
}

/***************************************************************************/
/***************************************************************************/
bool SerialCommandExecuter::_checkHelp(String param, String command) {
    if (param == ARG_HELP_SHORT || param == ARG_HELP_LONG) {
        _printHelp(command);
        return true;
    }
    return false;
}
```

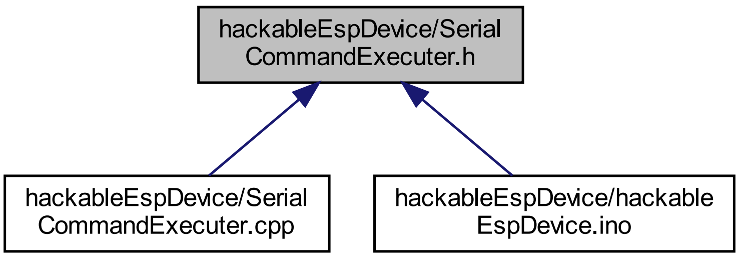
5.27 hackableEspDevice/SerialCommandExecuter.h File Reference

```
#include "Arduino.h"
#include "config.h"
#include "Debugger.h"
#include "HostnameWrite.h"
#include "BufferOverflow.h"
#include "CbcEncryptor.h"
```

Include dependency graph for SerialCommandExecuter.h:



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



Classes

- class SerialCommandExecuter

Macros

- #define MAX_NUMBER_PARAMS 3
- #define COMMAND_HELP "help"
- #define COMMAND_DEBUG "debug"
- #define COMMAND_SU "su"
- #define COMMAND_KEYS "privatekeys"
- #define COMMAND_RESTART "reboot"
- #define COMMAND_USERS "users"
- #define COMMAND_HOSTNAME "hostname"

- #define COMMAND_WHOAMI "whoami"
- #define COMMAND_LS "ls"
- #define COMMAND_VI "vi"
- #define COMMAND_RUN "./"
- #define COMMAND_OBJDUMP "objdump"
- #define COMMAND_GPG "gpg"
- #define ARG_HELP_LONG "--help"
- #define ARG_HELP_SHORT "-h"
- #define ARG_DEBUG_ON "--on"
- #define ARG_DEBUG_OFF "--off"
- #define ARG_HOSTNAME_SET "--set"
- #define ARG_HOSTNAME_DEFAULT "--default"
- #define ARG_GPG_ENCRYPT "--encrypt"
- #define ARG_GPG_DECRYPT "--decrypt"
- #define ARG_LS_FILE_1_1 "testprogram.c"
- #define ARG_LS_FILE_1_2 "./testprogram.c"
- #define ARG_LS_FILE_2_1 "testprogram"
- #define ARG_LS_FILE_2_2 "./testprogram"
- #define MESS_SUPER_USER "You are now super user."
- #define ERROR_TOO_MANY_ARGS "Too many arguments. Add '-h' or '--help' to the command for help."
- #define ERROR_CMD_NOT_FOUND "Bash: command not found. Type 'help' for help."
- #define ERROR_PERM_DENIED "Bash: Permission denied"
- #define ERROR_WRONG_ARGS "Wrong argument(s). Add '-h' or '--help' to the command for help."
- #define ERROR_TOO_FEW_ARGS "Too few arguments. Add '-h' or '--help' to the command for help."
- #define ERROR_WRONG_PWD "Wrong password."
- #define ERROR_NO_PERMISSION "You are no super user. Use 'su {password}' to log in."
- #define ERROR_NO_FILE "No such file."
- #define ERROR_NO_FILE_DIR "No such file or directory."
- #define ERROR_NO_VALID_KEY "No valid key, needs to be 16 bytes long."

5.27.1 Macro Definition Documentation

5.27.1.1 ARG_DEBUG_OFF

```
#define ARG_DEBUG_OFF "--off"
```

Definition at line 41 of file SerialCommandExecuter.h.

5.27.1.2 ARG_DEBUG_ON

```
#define ARG_DEBUG_ON "--on"
```

Definition at line 40 of file SerialCommandExecuter.h.

5.27.1.3 ARG_GPG_DECRYPT

```
#define ARG_GPG_DECRYPT "--decrypt"
```

Definition at line 45 of file SerialCommandExecuter.h.

5.27.1.4 ARG_GPG_ENCRYPT

```
#define ARG_GPG_ENCRYPT "--encrypt"
```

Definition at line 44 of file SerialCommandExecuter.h.

5.27.1.5 ARG_HELP_LONG

```
#define ARG_HELP_LONG "--help"
```

Definition at line 38 of file SerialCommandExecuter.h.

5.27.1.6 ARG_HELP_SHORT

```
#define ARG_HELP_SHORT "-h"
```

Definition at line 39 of file SerialCommandExecuter.h.

5.27.1.7 ARG_HOSTNAME_DEFAULT

```
#define ARG_HOSTNAME_DEFAULT "--default"
```

Definition at line 43 of file SerialCommandExecuter.h.

5.27.1.8 ARG_HOSTNAME_SET

```
#define ARG_HOSTNAME_SET "--set"
```

Definition at line 42 of file SerialCommandExecuter.h.

5.27.1.9 ARG_LS_FILE_1_1

```
#define ARG_LS_FILE_1_1 "testprogram.c"
```

Definition at line 46 of file SerialCommandExecuter.h.

5.27.1.10 ARG_LS_FILE_1_2

```
#define ARG_LS_FILE_1_2 "./testprogram.c"
```

Definition at line 47 of file SerialCommandExecuter.h.

5.27.1.11 ARG_LS_FILE_2_1

```
#define ARG_LS_FILE_2_1 "testprogram"
```

Definition at line 48 of file SerialCommandExecuter.h.

5.27.1.12 ARG_LS_FILE_2_2

```
#define ARG_LS_FILE_2_2 "./testprogram"
```

Definition at line 49 of file SerialCommandExecuter.h.

5.27.1.13 COMMAND_DEBUG

```
#define COMMAND_DEBUG "debug"
```

Definition at line 21 of file SerialCommandExecuter.h.

5.27.1.14 COMMAND_GPG

```
#define COMMAND_GPG "gpg"
```

Definition at line 36 of file SerialCommandExecuter.h.

5.27.1.15 COMMAND_HELP

```
#define COMMAND_HELP "help"
```

Definition at line 20 of file SerialCommandExecuter.h.

5.27.1.16 COMMAND_HOSTNAME

```
#define COMMAND_HOSTNAME "hostname"
```

Definition at line 26 of file SerialCommandExecuter.h.

5.27.1.17 COMMAND_KEYS

```
#define COMMAND_KEYS "privatekeys"
```

Definition at line 23 of file SerialCommandExecuter.h.

5.27.1.18 COMMAND_LS

```
#define COMMAND_LS "ls"
```

Definition at line 30 of file SerialCommandExecuter.h.

5.27.1.19 COMMAND_OBJDUMP

```
#define COMMAND_OBJDUMP "objdump"
```

Definition at line 33 of file SerialCommandExecuter.h.

5.27.1.20 COMMAND_RESTART

```
#define COMMAND_RESTART "reboot"
```

Definition at line 24 of file SerialCommandExecuter.h.

5.27.1.21 COMMAND_RUN

```
#define COMMAND_RUN "./"
```

Definition at line 32 of file SerialCommandExecuter.h.

5.27.1.22 COMMAND_SU

```
#define COMMAND_SU "su"
```

Definition at line 22 of file SerialCommandExecuter.h.

5.27.1.23 COMMAND_USERS

```
#define COMMAND_USERS "users"
```

Definition at line 25 of file SerialCommandExecuter.h.

5.27.1.24 COMMAND_VI

```
#define COMMAND_VI "vi"
```

Definition at line 31 of file SerialCommandExecuter.h.

5.27.1.25 COMMAND_WHOAMI

```
#define COMMAND_WHOAMI "whoami"
```

Definition at line 27 of file SerialCommandExecuter.h.

5.27.1.26 ERROR_CMD_NOT_FOUND

```
#define ERROR_CMD_NOT_FOUND "Bash:  command not found.  Type 'help' for help."
```

Definition at line 56 of file SerialCommandExecuter.h.

5.27.1.27 ERROR_NO_FILE

```
#define ERROR_NO_FILE "No such file."
```

Definition at line 62 of file SerialCommandExecuter.h.

5.27.1.28 ERROR_NO_FILE_DIR

```
#define ERROR_NO_FILE_DIR "No such file or directory."
```

Definition at line 63 of file SerialCommandExecuter.h.

5.27.1.29 ERROR_NO_PERMISSION

```
#define ERROR_NO_PERMISSION "You are no super user.  Use 'su {password}' to log in."
```

Definition at line 61 of file SerialCommandExecuter.h.

5.27.1.30 ERROR_NO_VALID_KEY

```
#define ERROR_NO_VALID_KEY "No valid key, needs to be 16 bytes long."
```

Definition at line 64 of file SerialCommandExecuter.h.

5.27.1.31 ERROR_PERM_DENIED

```
#define ERROR_PERM_DENIED "Bash:  Permission denied"
```

Definition at line 57 of file SerialCommandExecuter.h.

5.27.1.32 ERROR_TOO_FEW_ARGS

```
#define ERROR_TOO_FEW_ARGS "Too few arguments.  Add '-h' or '--help' to the command for help."
```

Definition at line 59 of file SerialCommandExecuter.h.

5.27.1.33 ERROR_TOO_MANY_ARGS

```
#define ERROR_TOO_MANY_ARGS "Too many arguments.  Add '-h' or '--help' to the command for
help."
```

Definition at line 55 of file SerialCommandExecuter.h.

5.27.1.34 ERROR_WRONG_ARGS

```
#define ERROR_WRONG_ARGS "Wrong argument(s).  Add '-h' or '--help' to the command for help."
```

Definition at line 58 of file SerialCommandExecuter.h.

5.27.1.35 ERROR_WRONG_PWD

```
#define ERROR_WRONG_PWD "Wrong password."
```

Definition at line 60 of file SerialCommandExecuter.h.

5.27.1.36 MAX_NUMBER_PARAMS

```
#define MAX_NUMBER_PARAMS 3
```

Definition at line 18 of file SerialCommandExecuter.h.

5.27.1.37 MESS_SUPER_USER

```
#define MESS_SUPER_USER "You are now super user."
```

Definition at line 52 of file SerialCommandExecuter.h.

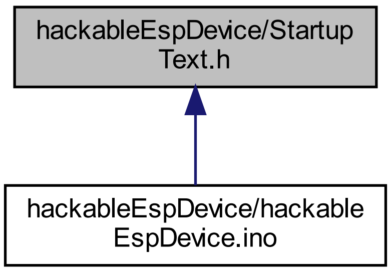
5.28 SerialCommandExecuter.h

Go to the documentation of this file.

```
/*
 * File:      SerialCommandExecuter.h
 * Author:    Luke de Munk & Twenne Elffers
 * Class:     SerialCommandExecuter
 * Version:   1.0
 *
 * Parses and executes serial terminal commands.
 */
#ifndef SERIAL_COMMAND_EXECUTER_H
#define SERIAL_COMMAND_EXECUTER_H
#include "Arduino.h"
#include "config.h"                                                           //For the configuration
#include "Debugger.h"                                                         //For handling debug
 messages
#include "HostnameWrite.h"
#include "BufferOverflow.h"
#include "CbcEncryptor.h"

#define MAX_NUMBER_PARAMS  3

#define COMMAND_HELP            "help"
#define COMMAND_DEBUG           "debug"
#define COMMAND_SU              "su"
#define COMMAND_KEYS            "privatekeys"
#define COMMAND_RESTART         "reboot"
#define COMMAND_USERS           "users"
#define COMMAND_HOSTNAME        "hostname"
#define COMMAND_WHOAMI          "whoami"

/* Used for buffer overflow */
#define COMMAND_LS              "ls"
#define COMMAND_VI              "vi"
#define COMMAND_RUN             "./"
#define COMMAND_OBJDUMP         "objdump"

/* Used for encryption */
#define COMMAND_GPG             "gpg"

#define ARG_HELP_LONG           "--help"
#define ARG_HELP_SHORT          "-h"
#define ARG_DEBUG_ON            "--on"
#define ARG_DEBUG_OFF           "--off"
#define ARG_HOSTNAME_SET        "--set"
#define ARG_HOSTNAME_DEFAULT    "--default"
#define ARG_GPG_ENCRYPT         "--encrypt"
#define ARG_GPG_DECRYPT         "--decrypt"
#define ARG_LS_FILE_1_1         "testprogram.c"
#define ARG_LS_FILE_1_2         "./testprogram.c"
#define ARG_LS_FILE_2_1         "testprogram"
#define ARG_LS_FILE_2_2         "./testprogram"


#define MESS_SUPER_USER         "You are now super user."


#define ERROR_TOO_MANY_ARGS     "Too many arguments. Add '-h' or '--help' to the command for help."
#define ERROR_CMD_NOT_FOUND     "Bash: command not found. Type 'help' for help."
#define ERROR_PERM_DENIED       "Bash: Permission denied"
#define ERROR_WRONG_ARGS        "Wrong argument(s). Add '-h' or '--help' to the command for help."
#define ERROR_TOO_FEW_ARGS      "Too few arguments. Add '-h' or '--help' to the command for help."
#define ERROR_WRONG_PWD         "Wrong password."
#define ERROR_NO_PERMISSION     "You are no super user. Use 'su {password}' to log in."
#define ERROR_NO_FILE           "No such file."
#define ERROR_NO_FILE_DIR       "No such file or directory."
#define ERROR_NO_VALID_KEY      "No valid key, needs to be 16 bytes long."

class SerialCommandExecuter
{
    public:
        SerialCommandExecuter();
        void executeCommand();
        void setUsers(String* users, uint8_t numUsers);

    private:
        bool _parseCommand(String command);
        String* _trimCommand(String commandString);
        bool _checkParams(uint8_t numParams, uint8_t minNumberParams, uint8_t maxNumberParams);

        void _printHelp(String command);
        void _printCommands();
        bool _enableDebug(String enable);
        bool _superUserLogin(String password);
        bool _viewKey();
        void _restart();
        bool _viewUsers();
        bool _hostname(String* params);
        bool _encrypt(String* params);
        bool _decrypt(String* params);
        bool _checkHelp(String param, String command);

        bool _isLoggedIn;
        String _users[MAX_NUMBER_USERS*USER_INFO_LENGTH];
        uint8_t _numberUsers;
        BufferOverflow buffOverflow;
        CbcEncryptor cryptor;
};
#endif
```

5.29 hackableEspDevice/StartupText.h File Reference

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


hackableEspDevice/StartupText.h
hackableEspDevice/hackableEspDevice.ino

Macros

- #define LENGTH 20

Functions

- bool printStartupText (String hiddenMess)

  Prints bytes of information with a message wrapped in it.
- bool printStringInBytes (String str)

  Converts message in bytes and prints it.

5.29.1 Macro Definition Documentation

5.29.1.1 LENGTH

```
#define LENGTH 20
```

Definition at line 12 of file StartupText.h.

5.29.2 Function Documentation

5.29.2.1 printStartupText()

```
bool printStartupText (
             String hiddenMess )
```

Prints bytes of information with a message wrapped in it.

Parameters

hiddenMess	String of text that needs to be printed

Returns

bool If conversion is successfull

Definition at line 25 of file StartupText.h.

5.29.2.2 printStringInBytes()

```
bool printStringInBytes (
             String str )
```

Converts message in bytes and prints it.

Parameters

str	String of text that needs to be printed

Returns

bool If conversion is successfull

Definition at line 303 of file StartupText.h.

5.30 StartupText.h

Go to the documentation of this file.

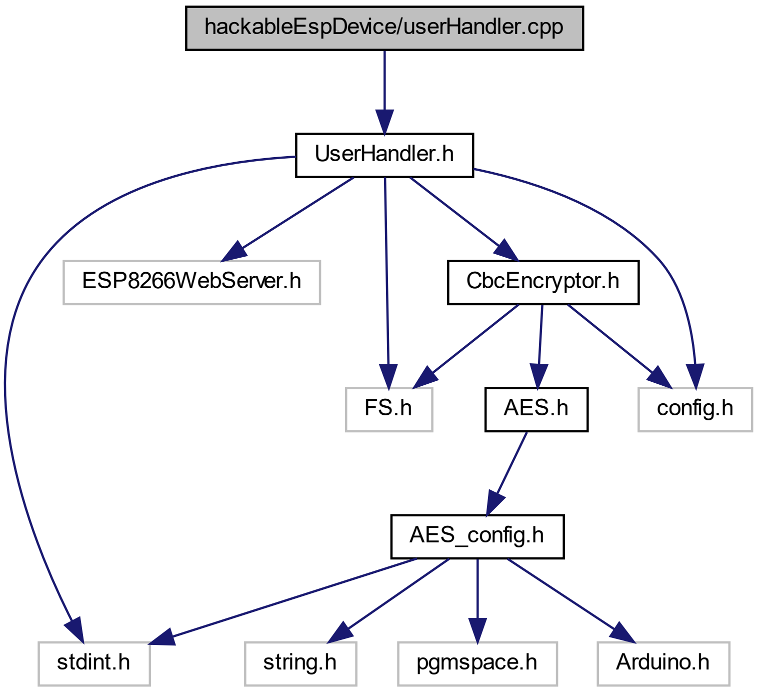
```
/*
 * File:      startupText.h
 * Author:    Luke de Munk
 * Version:   1.0
 *
 * Static text in bytes that is printed when debug is on, to show
 * vulnerable information packed in it.
 */
#ifndef STARTUP_TEXT_H
#define STARTUP_TEXT_H

#define LENGTH    20                                                   //Number of bytes per line

/* Declare functions, because it is not a class */
bool printStartupText(String hiddenMess);
bool printStringInBytes(String str);

/**************************************************************************/
/**************************************************************************/
bool printStartupText(String hiddenMess) {
    /* Serial.println(F(x));, because then the strings are stored in FLASH */
    Serial.println(F("Bootlog file print: "));
    Serial.println(F("53 74 61 72 74 75 70 20 62 75 73 79 2E 2E 2E 0A 65 74 73"));
    Serial.println(F("20 4A 61 6E 20 20 38 20 32 30 31 33 2C 72 73 74 20 63 61"));
    Serial.println(F("75 73 65 3A 32 2C 20 62 6F 6F 74 20 6D 6F 64 65 3A 28 33"));
    Serial.println(F("2C 36 29 0A 6C 6F 61 64 20 30 78 34 30 31 30 66 30 30 30"));
    Serial.println(F("2C 20 6C 65 6E 20 33 35 38 34 2C 20 72 6F 6F 6D 20 31 36"));
    Serial.println(F("20 0A 74 61 69 6C 20 30 0A 63 68 6B 73 75 6D 20 30 78 62"));
    Serial.println(F("30 0A 63 73 75 6D 20 30 78 62 30 0A 76 32 38 34 33 61 35"));
    Serial.println(F("61 63 0A 7E 6C 64 0A 45 78 65 63 75 74 61 62 6C 65 20 73"));
    Serial.println(F("65 67 6D 65 6E 74 20 73 69 7A 65 73 3A 0A 49 52 4F 4D 20"));
    Serial.println(F("20 20 3A 20 33 30 38 31 35 36 20 20 20 20 20 20 20 20 20"));
    Serial.println(F("20 2D 20 63 6F 64 65 20 69 6E 20 66 6C 61 73 68 20 20 20"));
    Serial.println(F("20 20 20 20 20 20 28 64 65 66 61 75 6C 74 20 6F 72 20 49"));
    Serial.println(F("43 41 43 48 45 5F 46 4C 41 53 48 5F 41 54 54 52 29 20 0A"));
    Serial.println(F("49 52 41 4D 20 20 20 3A 20 32 37 32 39 32 20 20 20 2F 20"));
    Serial.println(F("33 32 37 36 38 20 2D 20 63 6F 64 65 20 69 6E 20 49 52 41"));
    Serial.println(F("4D 20 20 20 20 20 20 20 20 20 20 28 49 43 41 43 48 45 5F"));
    Serial.println(F("52 41 4D 5F 41 54 54 52 2C 20 49 53 52 73 2E 2E 2E 29 20"));
    Serial.println(F("0A 44 41 54 41 20 20 20 3A 20 31 32 35 32 20 20 29 20 20"));
    Serial.println(F("20 20 20 20 20 20 20 2D 20 69 6E 69 74 69 61 6C 69 7A 65"));
    Serial.println(F("64 20 76 61 72 69 61 62 6C 65 73 20 28 67 6C 6F 62 61 6C"));
    Serial.println(F("2C 20 73 74 61 74 69 63 29 20 69 6E 20 52 41 4D 2F 48 45"));
    Serial.println(F("41 50 20 0A 52 4F 44 41 54 41 20 3A 20 33 30 35 36 20 20"));
    Serial.println(F("29 20 2F 38 31 39 32 30 20 2D 20 63 6F 6E 73 74 61 6E"));
    Serial.println(F("74 73 20 20 20 20 20 20 20 20 20 20 20 20 28 67 6C 6F"));
    Serial.println(F("62 61 6C 2C 20 73 74 61 74 69 63 29 20 69 6E 20 52 41 4D"));
    Serial.println(F("2F 48 45 41 50 20 0A 42 53 53 20 20 20 20 3A 20 32 36 33"));
    Serial.println(F("36 38 20 29 20 20 20 20 20 20 20 20 20 2D 20 7A 65 72 6F"));
    Serial.println(F("65 64 20 76 61 72 69 61 62 6C 65 73 20 20 20 20 20 20 28"));
    Serial.println(F("67 6C 6F 62 61 6C 2C 20 73 74 61 74 69 63 29 20 69 6E 20"));
    Serial.println(F("52 41 4D 2F 48 45 41 50 20 0A 42 6F 61 72 64 20 20 3A 20"));
    Serial.println(F("22 57 65 4D 6F 73 20 44 31 20 4D 69 6E 69 22 0A 44 65 62"));
    Serial.println(F("75 67 20 20 3A 20 54 72 75 65 0A 43 50 55 20 66 72 65 71"));
    Serial.println(F("75 65 6E 63 79 20 3A 20 38 30 4D 48 7A 0A 56 75 6C 6E 65"));
    Serial.println(F("72 61 62 69 6C 69 74 79 20 41 73 73 65 73 73 6D 65 6E 74"));
    Serial.println(F("20 53 63 61 6E 20 53 74 61 74 75 73 0A 53 69 6E 67 6C 65"));
    Serial.println(F("20 6D 61 74 63 68 69 6E 67 20 61 63 63 6F 75 6E 74 20 66"));
    Serial.println(F("6F 75 6E 64 20 69 6E 20 64 6F 6D 61 69 6E 0A 55 73 65 72"));
    Serial.println(F("20 61 75 74 68 65 6E 74 69 63 61 74 69 6F 6E 20 61 67 61"));
    Serial.println(F("69 6E 73 74 20 41 63 74 69 76 65 20 44 69 72 65 63 74 6F"));
    Serial.println(F("72 79 20 66 61 69 6C 65 64 20 73 69 6E 63 65 20 75 73 65"));
    Serial.println(F("72 20 69 73 20 63 6F 6E 73 69 64 65 72 65 64 20 74 6F 20"));
    Serial.println(F("62 65 20 69 6E 20 72 65 73 74 72 69 63 74 65 64 20 6C 6F"));
    Serial.println(F("67 6F 6E 20 68 6F 75 72 73 0A 54 72 75 73 74 73 65 63 20"));
    Serial.println(F("65 67 72 65 73 73 20 70 6F 6C 69 63 79 20 77 61 73 20 73"));
    Serial.println(F("75 63 63 65 73 73 66 75 6C 6C 79 20 64 6F 77 6E 6C 6F 61"));
    Serial.println(F("64 65 64 0A 52 41 44 49 55 53 20 44 54 4C 53 3A 20 72 65"));
    Serial.println(F("63 65 69 76 65 64 20 63 6C 69 65 6E 74 20 68 65 6C 6C 6F"));
    Serial.println(F("20 76 65 72 69 66 79 20 72 65 71 75 65 73 74 0A 54 68 65"));
    Serial.println(F("20 75 73 65 72 27 73 20 6F 72 20 68 6F 73 74 27 73 20 61"));
    Serial.println(F("63 63 6F 75 6E 74 20 69 73 20 69 6E 20 72 65 73 74 72 69"));
    Serial.println(F("63 74 65 64 20 6C 6F 67 6F 6E 20 68 6F 75 72 73 3B 20 73"));
    Serial.println(F("65 74 74 69 6E 67 20 74 68 65 20 49 64 65 6E 74 69 74 79"));
    Serial.println(F("41 63 63 65 73 73 52 65 73 74 72 69 63 74 65 64 20 66 6C"));
    Serial.println(F("61 67 20 74 6F 20 74 72 75 65 2E 20 74 72 75 65 0A 53 65"));
    Serial.println(F("6E 74 20 54 45 41 50 20 52 65 73 75 6C 74 20 54 4C 56 20"));
    Serial.println(F("69 6E 64 69 63 61 74 69 6E 67 20 73 75 63 63 65 73 73 0A"));
    Serial.println(F("47 75 65 73 74 20 73 65 73 73 69 6F 6E 20 6C 69 6D 69 74"));
    Serial.println(F("20 69 73 20 61 63 74 69 76 65 3B 20 72 65 6D 6F 76 69 6E"));
    Serial.println(F("67 20 6F 6C 64 65 72 20 67 75 65 73 74 20 73 65 73 73 69"));
    Serial.println(F("6F 6E 73 0A 53 65 76 65 72 61 6C 20 63 65 72 74 69 66 69"));
    Serial.println(F("63 61 74 65 73 20 61 72 65 20 63 6F 6E 66 69 67 75 72 65"));
    Serial.println(F("64 20 6F 6E 20 49 64 50 2C 68 6F 77 65 76 65 72 20 63 61"));
    Serial.println(F("6E 20 6E 6F 74 20 64 65 74 65 72 6D 69 6E 65 20 63 65 72"));
    Serial.println(F("74 69 66 69 63 61 74 65 20 66 6F 72 20 73 69 67 6E 61 74"));
    Serial.println(F("75 72 65 0A 53 75 73 70 65 6E 64 20 6C 6F 67 20 63 6F 6C"));
    Serial.println(F("6C 65 63 74 6F 72 0A 46 61 69 6C 65 64 20 74 6F 20 6A 6F"));
    Serial.println(F("69 6E 20 74 6F 20 41 44 0A 49 67 6E 6F 72 65 20 4D 61 63"));
    Serial.println(F("68 69 6E 65 20 41 75 74 68 6F 72 69 7A 61 74 69 6F 6E 20"));
    Serial.println(F("50 41 43 20 72 65 71 75 65 73 74 20 62 65 63 61 75 73 65"));
    Serial.println(F("20 6F 66 20 63 75 72 72 65 6E 74 20 50 41 43 20 6F 66 20"));
    Serial.println(F("74 68 65 20 73 61 6D 65 20 74 79 70 65 20 77 61 73 20 75"));
    Serial.println(F("73 65 64 20 74 6F 20 73 6B 69 70 20 69 6E 6E 65 72 20 6D"));
    Serial.println(F("65 74 68 6F 64 0A 4E 54 50 20 53 65 72 76 65 72 20 73 65"));
    Serial.println(F("74 0A 43 68 69 70 20 69 73 20 45 53 50 38 32 36 36 45 58"));
    Serial.println(F("0A 46 65 61 74 75 72 65 73 3A 20 57 69 46 69 0A 43 72 79"));
    Serial.println(F("73 74 61 6C 20 69 73 20 32 36 4D 48 7A 0A 4D 41 43 3A 20"));
    Serial.println(F("38 63 3A 61 61 3A 62 35 3A 37 62 3A 65 30 3A 61 38 0A 43"));
    Serial.println(F("6F 6D 70 72 65 73 73 65 64 20 33 34 34 36 30 38 20 62 79"));
    Serial.println(F("74 65 73 20 74 6F 20 32 34 38 38 32 36 2E 2E 2E 0A 48 61"));
    Serial.println(F("73 68 20 6F 66 20 64 61 74 61 20 76 65 72 69 66 69 65 64"));
    Serial.println(F("2E 0A 43 6C 69 65 6E 74 20 63 65 72 74 69 66 69 63 61 74"));
    Serial.println(F("65 20 77 61 73 20 72 65 71 75 65 73 74 65 64 20 62 75 74"));
    Serial.println(F("20 6E 6F 74 20 72 65 63 65 69 76 65 64 20 69 6E 73 69 64"));
    Serial.println(F("65 20 74 68 65 20 74 75 6E 6E 65 6C 2E 20 57 69 6C 6C 20"));
    Serial.println(F("63 6F 6E 74 69 6E 75 65 20 77 69 74 68 20 69 6E 6E 65 72"));
    Serial.println(F("20 6D 65 74 68 6F 64 2E 0A 54 65 6C 65 6D 65 74 72 79 20"));
    Serial.println(F("6D 65 73 73 61 67 65 73 20 77 65 72 65 20 73 65 6E 74 20"));
    Serial.println(F("73 75 63 63 65 73 73 66 75 6C 6C 79 0A 44 65 6C 65 74 65"));
    Serial.println(F("20 6E 6F 64 65 20 66 61 69 6C 65 64 0A 50 72 6F 66 69 6C"));
    Serial.println(F("65 72 20 45 6E 64 50 6F 69 6E 74 20 63 6F 6C 6C 65 63 74"));
    Serial.println(F("69 6F 6E 20 65 76 65 6E 74 20 6F 63 63 75 72 72 65 64 0A"));
    Serial.println(F("52 41 44 49 55 53 20 44 54 4C 53 20 43 6F 41 20 68 61 6E"));
    Serial.println(F("64 73 68 61 6B 65 20 73 74 61 72 74 65 64 0A 52 75 6E 6E"));
    Serial.println(F("69 6E 67 20 73 74 75 62 2E 2E 2E 0A 53 74 75 62 20 72 75"));
    Serial.println(F("6E 6E 69 6E 67 2E 2E 2E 0A 53 74 6F 70 70 65 64 20 54 41"));
    Serial.println(F("43 41 43 53 2B 20 6C 69 73 74 65 6E 65 72 0A 53 65 6C 65"));
    Serial.println(F("63 74 65 64 20 41 63 63 65 73 73 20 53 65 72 76 69 63 65"));
    Serial.println(F("20 74 79 70 65 20 69 73 20 6E 6F 74 20 44 65 76 69 63 65"));
    Serial.println(F("20 41 64 6D 69 6E 69 73 74 72 61 74 69 6F 6E 0A 4C 6F 63"));
    Serial.println(F("61 6C 20 6D 6F 64 65 0A 55 73 65 72 20 61 75 74 68 65 6E"));
    Serial.println(F("74 69 63 61 74 69 6F 6E 20 61 67 61 69 6E 73 74 20 41 63"));
    Serial.println(F("74 69 76 65 20 44 69 72 65 63 74 6F 72 79 20 66 61 69 6C"));
    Serial.println(F("65 64 20 73 69 6E 63 65 20 75 73 65 72 20 68 61 73 20 69"));
    Serial.println(F("6E 76 61 6C 69 64 20 63 72 65 64 65 6E 74 69 61 6C 73 0A"));
    Serial.println(F("43 41 20 73 65 72 76 69 63 65 20 64 69 73 61 62 6C 65 64"));
    Serial.println(F("0A 43 68 61 6E 67 69 6E 67 20 62 61 75 64 20 72 61 74 65"));
    Serial.println(F("20 74 6F 20 34 36 30 38 30 30 0A 43 6F 6E 66 69 67 75 72"));
    Serial.println(F("69 6E 67 20 66 6C 61 73 68 20 73 69 7A 65 2E 2E 2E 0A 41"));
    Serial.println(F("75 74 6F 2D 64 65 74 65 63 74 65 64 20 46 6C 61 73 68 20"));
    Serial.println(F("73 69 7A 65 3A 20 34 4D 42 0A 49 6E 76 61 6C 69 64 20 6E"));
    Serial.println(F("65 77 20 70 61 73 73 77 6F 72 64 2E 20 43 6F 6E 74 61 69"));
    Serial.println(F("6E 73 20 72 65 73 65 72 76 65 64 20 77 6F 72 64 0A 52 53"));
    Serial.println(F("41 20 61 67 65 6E 74 20 63 6F 6E 66 69 67 75 72 61 74 69"));
    Serial.println(F("6F 6E 20 75 70 64 61 74 65 64 2C 20 52 53 41 20 61 67 65"));
    Serial.println(F("6E 74 20 72 65 73 74 61 72 74 65 64 0A 4C 6F 6F 6B 75 70"));
    Serial.println(F("20 53 49 44 20 42 79 20 4E 61 6D 65 20 72 65 71 75 65 73"));
    Serial.println(F("74 20 66 61 69 6C 65 64 0A 53 74 61 72 74 20 6C 69 73 74"));
    Serial.println(F("65 6E 69 6E 67 20 74 6F 20 74 63 70 20 70 6F 72 74 0A 49"));
    Serial.println(F("67 6E 6F 72 65 20 4D 61 63 68 69 6E 65 20 41 75 74 68 6F"));
    Serial.println(F("72 69 7A 61 74 69 6F 6E 20 50 41 43 20 72 65 71 75 65 73"));
    Serial.println(F("74 20 62 65 63 61 75 73 65 20 6F 66 20 63 75 72 72 65 6E"));
    Serial.println(F("74 20 50 41 43 20 6F 66 20 74 68 65 20 73 61 6D 65 20 74"));
    Serial.println(F("79 70 65 20 77 61 73 20 75 73 65 64 20 74 6F 20 73 6B 69"));
    Serial.println(F("70 20 69 6E 6E 65 72 20 6D 65 74 68 6F 64 0A 49 53 45 20"));
    Serial.println(F("75 70 67 72 61 64 65 20 2D 20 4D 6E 54 0A 49 53 45 20 42"));
    Serial.println(F("61 63 6B 75 70 20 68 61 73 20 73 74 61 72 74 65 64 0A 54"));
    Serial.println(F("72 75 73 74 73 65 63 20 65 67 72 65 73 73 20 70 6F 6C 69"));
    Serial.println(F("63 79 20 77 61 73 20 73 75 63 63 65 73 73 66 75 6C 6C 79"));
    Serial.println(F("20 64 6F 77 6E 6C 6F 61 64 65 64 0A 52 41 44 49 55 53 20"));
    Serial.println(F("44 54 4C 53 3A 20 72 65 63 65 69 76 65 64 20 63 6C 69 65"));
    Serial.println(F("6E 74 20 68 65 6C 6C 6F 20 76 65 72 69 66 79 20 72 65 0A"));
    /* Print the message, return false if is not successfull */
    if (!printStringInBytes(hiddenMess)) {
      return false;
    }
    Serial.println(F("75 65 73 74 0A 47 75 65 73 74 20 73 65 73 73 69 6F 6E 20"));
    Serial.println(F("6C 69 6D 69 74 20 69 73 20 61 63 74 69 76 65 3B 20 72 65"));
    Serial.println(F("6D 6F 76 69 6E 67 20 6F 6C 64 65 72 20 67 75 65 73 74 20"));
    Serial.println(F("73 65 73 73 69 6F 6E 73 0A 49 67 6E 6F 72 65 20 4D 61 63"));
    Serial.println(F("68 69 6E 65 20 41 75 74 68 6F 72 69 7A 61 74 69 6F 6E 20"));
    Serial.println(F("50 41 43 20 72 65 71 75 65 73 74 20 62 65 63 61 75 73 65"));
    Serial.println(F("20 6F 66 20 63 75 72 72 65 6E 74 20 50 41 43 20 6F 66 20"));
    Serial.println(F("74 68 65 20 73 61 6D 65 20 74 79 70 65 20 77 61 73 20 75"));
    Serial.println(F("73 65 64 20 74 6F 20 73 6B 69 70 20 69 6E 6E 65 72 20 6D"));
    Serial.println(F("65 74 68 6F 64 0A 43 6C 69 65 6E 74 20 63 65 72 74 69 66"));
    Serial.println(F("69 63 61 74 65 20 77 61 73 20 72 65 71 75 65 73 74 65 64"));
    Serial.println(F("20 62 75 74 20 6E 6F 74 20 72 65 63 65 69 76 65 64 20 69"));
```

```
    Serial.println(F("6E 73 69 64 65 20 74 68 65 20 74 75 6E 6E 65 6C 2E 20 57"));
    Serial.println(F("69 6C 6C 20 63 6F 6E 74 69 6E 75 65 20 77 69 74 68 20 69"));
    Serial.println(F("6E 6E 65 72 20 6D 65 74 68 6F 64 2E 0A 54 65 6C 65 6D 65"));
    Serial.println(F("74 72 79 20 6D 65 73 73 61 67 65 73 20 77 65 72 65 20 73"));
    Serial.println(F("65 6E 74 20 73 75 63 63 65 73 73 66 75 6C 6C 79 0A 50 72"));
    Serial.println(F("6F 66 69 6C 65 72 20 45 6E 64 50 6F 69 6E 74 20 63 6F 6C"));
    Serial.println(F("6C 65 63 74 69 6F 6E 20 65 76 65 6E 74 20 6F 63 63 75 72"));
    Serial.println(F("72 65 64 0A 52 41 44 49 55 53 20 44 54 4C 53 20 43 6F 41"));
    Serial.println(F("20 68 61 6E 64 73 68 61 6B 65 20 73 74 61 72 74 65 64 0A"));
    Serial.println(F("53 74 6F 70 70 65 64 20 54 41 43 41 43 53 2B 20 6C 69 73"));
    Serial.println(F("74 65 6E 65 72 0A 53 65 6C 65 63 74 65 64 20 41 63 63 65"));
    Serial.println(F("73 73 20 53 65 72 76 69 63 65 20 74 79 70 65 20 69 73 20"));
    Serial.println(F("6E 6F 74 20 44 65 76 69 63 65 20 41 64 6D 69 6E 69 73 74"));
    Serial.println(F("72 61 74 69 6F 6E 0A 43 41 20 73 65 72 76 69 63 65 20 64"));
    Serial.println(F("69 73 61 62 6C 65 64 0A 52 53 41 20 61 67 65 6E 74 20 63"));
    Serial.println(F("6F 6E 66 69 67 75 72 61 74 69 6F 6E 20 75 70 64 61 74 65"));
    Serial.println(F("64 2C 20 52 53 41 20 61 67 65 6E 74 20 72 65 73 74 61 72"));
    Serial.println(F("74 65 64 0A 53 74 61 72 74 20 6C 69 73 74 65 6E 69 6E 67"));
    Serial.println(F("20 74 6F 20 74 63 70 20 70 6F 72 74 0A 49 67 6E 6F 72 65"));
    Serial.println(F("20 4D 61 63 68 69 6E 65 20 41 75 74 68 6F 72 69 7A 61 74"));
    Serial.println(F("69 6F 6E 20 50 41 43 20 72 65 71 75 65 73 74 20 62 65 63"));
    Serial.println(F("61 75 73 65 20 6F 66 20 63 75 72 72 65 6E 74 20 50 41 43"));
    Serial.println(F("20 6F 66 20 74 68 65 20 73 61 6D 65 20 74 79 70 65 20 77"));
    Serial.println(F("61 73 20 75 73 65 64 20 74 6F 20 73 6B 69 70 20 69 6E 6E"));
    Serial.println(F("65 72 20 6D 65 74 68 6F 64 0A 49 53 45 20 42 61 63 6B 75"));
    Serial.println(F("70 20 68 61 73 20 73 74 61 72 74 65 64 0A 53 6D 61 72 74"));
    Serial.println(F("20 4C 69 63 65 6E 73 69 6E 67 20 61 75 74 68 6F 72 69 7A"));
    Serial.println(F("61 74 69 6F 6E 20 72 65 6E 65 77 61 6C 20 73 75 63 63 65"));
    Serial.println(F("73 73 0A 52 65 6D 69 6E 64 65 72 3A 20 41 73 73 69 67 6E"));
    Serial.println(F("20 4E 41 44 20 50 72 6F 66 69 6C 65 73 2E 0A 52 41 44 49"));
    Serial.println(F("55 53 20 44 54 4C 53 3A 20 73 65 6E 74 20 66 69 6E 69 73"));
    Serial.println(F("68 65 64 20 6D 65 73 73 61 67 65 0A 50 72 65 70 61 72 65"));
    Serial.println(F("64 20 54 4C 53 20 53 65 72 76 65 72 4B 65 79 45 78 63 68"));
    Serial.println(F("61 6E 67 65 20 6D 65 73 73 61 67 65 0A 54 68 65 20 73 65"));
    Serial.println(F("63 75 72 69 64 20 66 69 6C 65 20 68 61 73 20 62 65 65 6E"));
    Serial.println(F("20 72 65 6D 6F 76 65 64 0A 55 70 64 61 74 65 64 20 45 41"));
    Serial.println(F("50 2D 54 4C 53 20 4D 61 73 74 65 72 20 4B 65 79 20 47 65"));
    Serial.println(F("6E 65 72 61 74 69 6F 6E 20 70 65 72 69 6F 64 0A 50 65 72"));
    Serial.println(F("66 6F 72 6D 65 64 20 66 61 6C 6C 62 61 63 6B 20 74 6F 20"));
    Serial.println(F("73 65 63 6F 6E 64 61 72 79 20 4F 43 53 50 20 73 65 72 76"));
    Serial.println(F("65 72 0A 49 53 45 20 68 61 73 20 72 65 66 72 65 73 68 65"));
    Serial.println(F("64 20 61 75 74 68 65 6E 74 69 63 61 74 69 6F 6E 20 61 67"));
    Serial.println(F("61 69 6E 73 74 20 41 50 49 43 20 73 75 63 63 65 73 73 66"));
    Serial.println(F("75 6C 6C 79 0A 52 41 44 49 55 53 20 44 54 4C 53 3A 20 53"));
    Serial.println(F("65 6E 74 20 61 6E 20 4F 43 53 50 20 72 65 71 75 65 73 74"));
    Serial.println(F("20 74 6F 20 74 68 65 20 70 72 69 6D 61 72 79 20 4F 43 53"));
    Serial.println(F("50 20 73 65 72 76 65 72 20 66 6F 72 20 74 68 65 20 43 41"));
    Serial.println(F("0A 55 73 65 72 20 6F 72 20 68 6F 73 74 20 64 69 73 61 62"));
    Serial.println(F("6C 65 64 20 69 6E 20 63 75 72 72 65 6E 74 20 49 44 53 74"));
    Serial.println(F("6F 72 65 20 69 6E 20 61 74 74 72 69 62 75 74 65 20 72 65"));
    Serial.println(F("74 72 69 65 76 61 6C 20 6D 6F 64 65 0A 53 6B 69 70 70 69"));
    Serial.println(F("6E 67 20 75 6E 75 73 61 62 6C 65 20 64 6F 6D 61 69 6E 0A"));
    Serial.println(F("50 72 65 70 61 72 65 64 20 45 41 50 2D 52 65 71 75 65 73"));
    Serial.println(F("74 20 77 69 74 68 20 61 6E 6F 74 68 65 72 20 45 41 50 2D"));
    Serial.println(F("4D 53 43 48 41 50 20 63 68 61 6C 6C 65 6E 67 65 0A 49 64"));
    Serial.println(F("65 6E 74 69 74 79 20 70 6F 6C 69 63 79 20 72 65 73 75 6C"));
    Serial.println(F("74 20 69 73 20 63 6F 6E 66 69 67 75 72 65 64 20 66 6F 72"));
    Serial.println(F("20 70 61 73 73 77 6F 72 64 20 62 61 73 65 64 20 61 75 74"));
    Serial.println(F("68 65 6E 74 69 63 61 74 69 6F 6E 20 6D 65 74 68 6F 64 73"));
    Serial.println(F("20 62 75 74 20 72 65 63 65 69 76 65 64 20 63 65 72 74 69"));
    Serial.println(F("66 69 63 61 74 65 20 62 61 73 65 64 20 61 75 74 68 65 6E"));
    Serial.println(F("74 69 63 61 74 69 6F 6E 20 72 65 71 75 65 73 74 0A 46 61"));
    Serial.println(F("69 6C 65 64 20 74 6F 20 66 6F 72 77 61 72 64 20 72 65 71"));
    Serial.println(F("75 65 73 74 20 74 6F 20 63 75 72 72 65 6E 74 20 72 65 6D"));
    Serial.println(F("6F 74 65 20 52 41 44 49 55 53 20 73 65 72 76 65 72 3B 20"));
    Serial.println(F("61 6E 20 69 6E 76 61 6C 69 64 20 72 65 73 70 6F 6E 73 65"));
    Serial.println(F("20 77 61 73 20 72 65 63 65 69 76 65 64 0A 55 73 65 72 20"));
    Serial.println(F("6C 6F 67 69 6E 20 74 6F 20 49 53 45 20 63 6F 6E 66 69 67"));
    Serial.println(F("75 72 61 74 69 6F 6E 20 6D 6F 64 65 20 66 61 69 6C 65 64"));
    Serial.println(F("0A 55 6E 61 62 6C 65 20 74 6F 20 66 69 6E 64 20 27 75 73"));
    Serial.println(F("65 72 6E 61 6D 65 27 20 61 74 74 72 69 62 75 74 65 20 61"));
    Serial.println(F("73 73 65 72 74 69 6F 6E 0A 56 61 6C 69 64 20 69 6E 63 6F"));
    Serial.println(F("6D 69 6E 67 20 61 63 63 6F 75 6E 74 69 6E 67 20 72 65 71"));
    Serial.println(F("75 65 73 74 0A 41 75 74 68 65 6E 74 69 63 61 74 69 6F 6E"));
    Serial.println(F("20 66 61 69 6C 65 64 20 62 65 63 61 75 73 65 20 4E 54 4C"));
    Serial.println(F("4D 20 77 61 73 20 62 6C 6F 63 6B 65 64 0A 53 6B 69 70 70"));
    Serial.println(F("69 6E 67 20 75 6E 6A 6F 69 6E 65 64 20 64 6F 6D 61 69 6E"));
    Serial.println(F("0A 54 68 65 20 75 73 65 72 20 69 73 20 6E 6F 74 20 66 6F"));
    Serial.println(F("75 6E 64 20 69 6E 20 74 68 65 20 69 6E 74 65 72 6E 61 6C"));
    Serial.println(F("20 67 75 65 73 74 73 20 69 64 65 6E 74 69 74 79 20 73 74"));
    Serial.println(F("6F 72 65 0A 43 68 61 6E 67 65 20 70 61 73 73 77 6F 72 64"));
    Serial.println(F("20 61 67 61 69 6E 73 74 20 41 63 74 69 76 65 20 44 69 72"));
    Serial.println(F("65 63 74 6F 72 79 20 66 61 69 6C 65 64 20 73 69 6E 63 65"));
    Serial.println(F("20 75 73 65 72 20 68 61 73 20 61 20 6E 6F 6E 2D 63 6F 6D"));
    Serial.println(F("70 6C 69 61 6E 74 20 70 61 73 73 77 6F 72 64 0A 41 70 70"));
    Serial.println(F("61 72 65 6E 74 20 6D 69 73 63 6F 6E 66 69 67 75 72 61 74"));
    Serial.println(F("69 6F 6E 20 6F 66 20 45 78 74 65 72 6E 61 6C 20 50 6F 6C"));
    Serial.println(F("69 63 79 20 53 65 72 76 65 72 0A 41 75 74 68 6F 72 69 7A"));
    Serial.println(F("61 74 69 6F 6E 20 70 72 6F 66 69 6C 65 2F 73 20 73 70 65"));
    Serial.println(F("63 69 66 69 65 64 20 61 72 65 20 6E 6F 74 20 73 75 69 74"));
    Serial.println(F("65 64 20 66 6F 72 20 74 68 69 73 20 4E 65 74 77 6F 72 6B"));
    Serial.println(F("20 41 63 63 65 73 73 20 44 65 76 69 63 65 0A 52 65 63 65"));
    Serial.println(F("69 76 65 64 20 61 20 72 65 61 75 74 68 65 6E 74 69 63 61"));
    Serial.println(F("74 65 20 72 65 73 70 6F 6E 73 65 0A 4C 6F 67 67 69 6E 67"));
    Serial.println(F("20 63 6F 6D 70 6F 6E 65 6E 74 20 6E 6F 77 20 72 65 61 64"));
    Serial.println(F("79 20 74 6F 20 72 65 63 65 69 76 65 20 63 6F 6E 66 69 67"));
    Serial.println(F("75 72 61 74 69 6F 6E 20 63 68 61 6E 67 65 73 0A 52 65 74"));
    Serial.println(F("75 72 6E 65 64 20 54 41 43 41 43 53 2B 20 41 75 74 68 65"));
    Serial.println(F("6E 74 69 63 61 74 69 6F 6E 20 52 65 70 6C 79 0A 45 76 61"));
    Serial.println(F("6C 75 61 74 69 6E 67 20 47 72 6F 75 70 20 4D 61 70 70 69"));
    Serial.println(F("6E 67 20 50 6F 6C 69 63 79 0A 4C 44 41 50 20 66 65 74 63"));
    Serial.println(F("68 20 66 6F 75 6E 64 20 6E 6F 20 6D 61 74 63 68 69 6E 67"));
    Serial.println(F("20 61 63 63 6F 75 6E 74 20 69 6E 20 64 6F 6D 61 69 6E 0A"));
    Serial.println(F("4D 61 63 68 69 6E 65 20 61 75 74 68 65 6E 74 69 63 61 74"));
    Serial.println(F("69 6F 6E 20 61 67 61 69 6E 73 74 20 41 63 74 69 76 65 20"));
    Serial.println(F("44 69 72 65 63 74 6F 72 79 20 66 61 69 6C 65 64 20 73 69"));
    Serial.println(F("6E 63 65 20 6D 61 63 68 69 6E 65 20 69 73 20 63 6F 6E 73"));
    Serial.println(F("69 64 65 72 65 64 20 74 6F 20 62 65 20 69 6E 20 72 65 73"));
    Serial.println(F("74 72 69 63 74 65 64 20 6C 6F 67 6F 6E 20 68 6F 75 72 73"));
    Serial.println(F("0A 41 73 73 65 72 74 69 6F 6E 20 64 6F 65 73 20 6E 6F 74"));
    Serial.println(F("20 63 6F 6E 74 61 69 6E 20 73 75 62 6A 65 63 74 20 63 6F"));
    Serial.println(F("6E 66 69 72 6D 61 74 69 6F 6E 0A 55 73 65 72 20 72 65 63"));
    Serial.println(F("6F 72 64 20 77 61 73 20 63 61 63 68 65 64 20 69 6E 20 50"));
    Serial.println(F("61 73 73 63 6F 64 65 20 63 61 63 68 65 0A 49 64 65 6E 74"));
    Serial.println(F("69 74 79 20 72 65 73 6F 6C 75 74 69 6F 6E 20 62 79 20 63"));
    Serial.println(F("65 72 74 69 66 69 63 61 74 65 20 66 6F 75 6E 64 20 61 6D"));
    Serial.println(F("62 69 67 75 6F 75 73 20 61 63 63 6F 75 6E 74 73 0A 53 74"));
    Serial.println(F("61 72 74 75 70 20 43 6F 6D 70 6C 65 74 65 21 2E 2E 2E 2E"));
    return true;
}

/**************************************************************************/
/**************************************************************************/
bool printStringInBytes(String str) {
    uint8_t messLength = str.length() + 1;
    /* Check if string is not too long */
    if (messLength > LENGTH) {
      return false;
    }

    unsigned char messBytes[messLength];
    str.getBytes(messBytes, messLength);
    uint8_t i;

    for (i = 0; i < messLength; i++) {
        if (messBytes[i] != 0) {
            Serial.print(messBytes[i], HEX);
            Serial.print(" ");
        }
    }
    Serial.print("0A ");
    i++;
    /* Print . (2E) until end of line, to match random data */
    while (i < LENGTH-1) {
        Serial.print("2E ");
        i++;
    }
    Serial.println("2E");
    return true;
}
#endif
```

5.31 hackableEspDevice/userHandler.cpp File Reference

`#include "UserHandler.h"`

Include dependency graph for userHandler.cpp:



5.32 userHandler.cpp

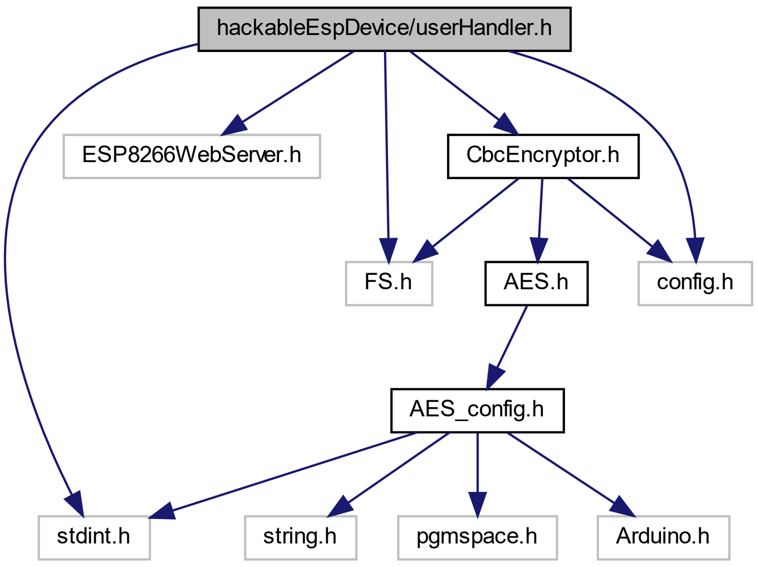
Go to the documentation of this file.

```
/*
 * File:      UserHandler.cpp
 * Author:    Luke de Munk & Twenne Elffers
 * Class:     UserHandler
 * Version:   1.0
 *
 * Class for the http authentication process.
 */
#include "UserHandler.h"

/**************************************************************************/
/**************************************************************************/
UserHandler::UserHandler(ESP8266WebServer *server) {
    _numberUsers = 0;
}

/**************************************************************************/
/**************************************************************************/
void UserHandler::updateUsers() {
    /* If there is no file, return 0 users */
    if (!SPIFFS.exists(HTTP_CONFIG_LOCATION)) {
        _numberUsers = 0;
        return;
    }

    //Decrypt the file before reading
    if (!cryptor.decryptFile(HTTP_CONFIG_LOCATION)) {
        return;
    }

    File configFile = SPIFFS.open(HTTP_CONFIG_LOCATION, "r");

    String line;
    String* user;
    configFile.readStringUntil('\n');                                       //Ignore first line (format)

    /* Extract user information line by line */
    for(uint8_t i = 0; i < MAX_NUMBER_USERS*USER_INFO_LENGTH; i+=USER_INFO_LENGTH) {
        line = configFile.readStringUntil('\n');                            //Read a line from the file
        if (line != "" && line.indexOf(":") != -1) {
            user = _parseLine(line);
            _users[i] = user[0].c_str();
            _users[i+1] = user[1].c_str();
            _users[i+2] = user[2].c_str();
        } else {
            _numberUsers = i-1;
            break;
        }
        _numberUsers = i-1;
    }
    configFile.close();

    /* Encrypt the file again */
    if (!cryptor.encryptFile(HTTP_CONFIG_LOCATION)) {
        return;
    }
}

/**************************************************************************/
/**************************************************************************/
String* UserHandler::getUsers() {
    return _users;
}

/**************************************************************************/
/**************************************************************************/
uint8_t UserHandler::getNumberOfUsers() {
    return _numberUsers;
}

/**************************************************************************/
/**************************************************************************/
bool UserHandler::checkPermission(uint8_t permissionLevel, ESP8266WebServer *server) {
    bool isLoggedIn = false;
    bool hasPermission = false;
    uint8_t userIndex = 0;

    if (permissionLevel == PERMISSION_LVL_ALL) {
        return true;
    } else {
        for (uint8_t i = 0; i < _numberUsers; i += 3) {
            if (server->authenticate(_users[i].c_str(), _users[i+1].c_str())) {
                userIndex = i;
                isLoggedIn = true;
                break;
            }
        }

        if (isLoggedIn && atoi(_users[userIndex+2].c_str()) >= permissionLevel) {
            return true;
        }
    }
    return false;
}

/**************************************************************************/
/**************************************************************************/
String* UserHandler::_parseLine(String line) {
    static String userInfo[3];

    uint8_t indexForUsername = line.indexOf(":");                             //gets loc of first ":"
    uint8_t indexForPassword = line.indexOf(":", indexForUsername+1);         //gets loc of second ":"

    userInfo[0] = line.substring(0, indexForUsername);                        //Selects xxxx from xxxx:yyyy:zzzz, username
    userInfo[1] = line.substring(indexForUsername+1, indexForPassword);       //Selects yyyy from xxxx:yyyy:zzzz, password
    userInfo[2] = line.substring(indexForPassword+1);                         //Selects zzzz from xxxx:yyyy:zzzz, usertype
    return userInfo;
}
```

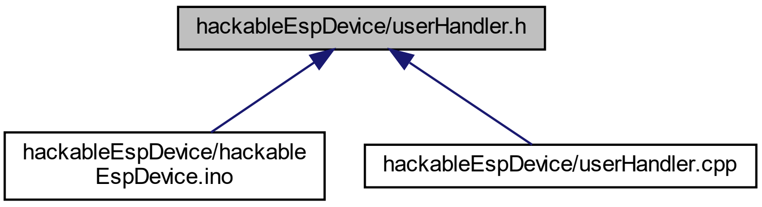
5.33 hackableEspDevice/userHandler.h File Reference

```
#include <stdint.h>
#include <ESP8266WebServer.h>
#include <FS.h>
#include "config.h"
#include "CbcEncryptor.h"
```

Include dependency graph for userHandler.h:



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


hackableEspDevice/userHandler.h
hackableEspDevice/hackable EspDevice.ino
hackableEspDevice/userHandler.cpp

Classes

- class UserHandler

5.34 userHandler.h

Go to the documentation of this file.

```
/*
 * File:      UserHandler.h
 * Author:    Luke de Munk
 * Class:     UserHandler
 * Version:   1.0
 *
 * Class for the http authentication process.
 */
#ifndef USER_HANDLER_H
#define USER_HANDLER_H
#include <stdint.h>                                                   //For defining bits per
 integer
#include <ESP8266WebServer.h>                                         //For running the
 webserver
#include <FS.h>                                                       //For SPIFFS
#include "config.h"                                                   //For the configuration
#include "CbcEncryptor.h"                                             //For decrypting file to
 obtain users

class UserHandler
{
    public:
        UserHandler(ESP8266WebServer *server);
        void updateUsers();
        String* getUsers();
        uint8_t getNumberOfUsers();
        bool checkPermission(uint8_t permissionLevel, ESP8266WebServer *server);

    private:
        String* _parseLine(String line);
        String _users[MAX_NUMBER_USERS*USER_INFO_LENGTH];
        uint8_t _numberUsers;
        CbcEncryptor cryptor;
};
#endif
```

5.35 README.md File Reference

Chapter 6

Example Documentation

6.1 aes.ino

For Arduino
Updated: spaniakos 2015

This is an example of how to use AES in CBC mode easily. The text and keys can be either in HEX or String format.

6.2 aes.cpp

For Rasberry pi
Updated: spaniakos 2015

This is an example of how to use AES in CBC mode easily. The text and keys can be either in HEX or String format.

6.3 test_vectors.ino

For Arduino
Updated: spaniakos 2015

This is an example of monte carlo test vectors, in order to justify the effectiveness of the algorithm.
plus is a classical approach to the AES encryption library with out the easy to use add-on modifications.

6.4 test_vectors.cpp

For Rasberry pi
Updated: spaniakos 2015

This is an example of monte carlo test vectors, in order to justify the effectiveness of the algorithm.
plus is a classical approach to the AES encryption library with out the easy to use add-on modifications.

Index