wolfSentry Documentation

2022-07-01

Contents

1 Introduction

This manual is writen as a technical guide to the wolfSentry embedded Intrusion Detection Protection System (IDPS). It will explain how to build and get started with wolfSentry, provide an overview of build options, features, portability enhancements, support, and much more.

1.1 Why Choose wolfSentry?

wolfSentry is incredibly lightweight in both size and resource overhead, but is still a very powerful IDPS for embedded needs.

2 Building wolfSentry

wolfSentry was written with portability in mind and should generally be easy to build on most systems. If you have difficulty building wolfSentry, please don't hesitate to seek support through our support forums (https://www.wolfssl.com/forums) or contact us directly at support@wolfssl.com.

2.1 Getting wolfSentry Source Code

The most recent version of wolfSentry can be downloaded from the wolfSSL website as a ZIP file:

https://www.wolfssl.com/download

After downloading the ZIP file, unzip the file using the `unzip` command. To use native line endings, enable the `-a` modifier when using unzip. From the unzip man page, the `-a` modifier functionality is described:

> [...] The -a option causes files identified by zip as text files (those with the 't' label in zipinfo listings, rather than 'b') to be automatically extracted as such, converting line endings, end- of-file characters and the character set itself as necessary. [...]

2.2 Building

For platforms that support GNU Make then running `make` will build wolfSentry as normal. For other platforms you will need to compile using the C files in the `src` directory and its `json` subdirectory.

To build and run the test suite you can use `make test`.

2.2.1 Build Options

There are several flags that can be added to `make` and flags that be used as build-time definitions in either a header file or `CFLAGS`. To use the `make` flags you can use:

```
make SINGLETHREADED=1 EXTRA_CFLAGS='-DWOLFSENTRY_NO_CLOCK_BUILTIN'
```

These will be stored at build time in the `wolfsentry_options.h` file where wolfSentry is built. If you are not using `make` then you can create this file with the following template:

```
#ifndef WOLFSENTRY_OPTIONS_H
#define WOLFSENTRY_OPTIONS_H

#endif /* WOLFSENTRY_OPTIONS_H */
```

The following table lists the possible options:

make option	Preprocesssor Macro	Description
v		Verbose make output
USER_MAKE_CONF		A user-defined Makefile to include
SRC_TOP		The source code top level directory (default pwd -P)
DEBUG		Compiler debugging flag to use (default -ggdb)
OPTIM		The optimizer flag to use (default -03)
C_WARNFLAGS		The warning flags to use (default See [1])
NO_STDIO	WOLFSENTRY_NO_STDIO	The platform does not have STDIO

make option	Preprocesssor Macro	Description
NO_JSON	WOLFSENTRY_NO_JSON	Do not compile JSON configuration support
USER_SETTINGS_FILE	WOLFSENTRY_USER_SETTINGS_FILE	An additional header file to include
SINGLETHEADED	WOLFSENTRY_SINGLETHREADED	Do not use thread safe semantics for single threaded uses
STATIC		Build a static binary
STRIPPED		Strip binaries of debugging symbols
BUILD_DYNAMIC		Build dynamic library
VERY_QUIET		Enable a very quiet build
TAR		Path to GNU tar binary for `make dist`, should be set to `gtar` for macOS
VERSION		The version number to compile in (default See [2])
	CENTIJSON_USE_LOCALE	JSON parser should use locale-dependent characters
	WOLFSENTRY_NO_PROTOCOL_NAMES	Disable support for protocol names in configuration files
	DEBUG_JSON	Add debugging `printf()` to the JSON parser
	WOLFSENTRY_NO_ERROR_STRINGS	Disable error code to error string functions
	WOLFSENTRY_NO_MALLOC_BUILTINS	Disable builtin malloc functions
	WOLFSENTRY_HAVE_NONGNU_ATOMICS	Atomics are non-GNU (ignored if SINGLETHREADED is set)
	WOLFSENTRY_NO_CLOCK_BUILTIN	Do not use bulitin time functions
	WOLFSENTRY_LWIP	wolfSentry is being built against LWIP instead of BSD sockets
	FREERTOS	Build with FreeRTOS support

[1] `-Wall -Wextra -Werror -Wformat=2 -Winit-self -Wmissing-include-dirs -Wunknown-pragmas -Wshadow -Wpointer-arith -Wcast-align -Wwrite-strings -Wconversion -Wstrict-prototypes -Wold-style-definition -Wmissing-declarations -Wmissing-format-attribute -Wpointer-arith -Woverlength-strings -Wredundant-decls -Winline -Winvalid-pch -Wdouble-promotion -Wvla -Wno-missing-field-initializers -Wno-bad-function-cast -Wno-type-limits` and if GCC is used the additional flags of `-Wjump-misses-init -Wlogical-op` are used.

[2] Defined as `VERSION := $(shell git rev-parse --short=8 HEAD 2>/dev/null || echo xxxxxxxx)$(shell git diff --quiet 2>/dev/null || [ $$? -ne 1 ] || echo "-dirty")`

3 wolfSentry API