wolfSentry Documentation

2025-04-30

Contents

1 wolfSentry – The Wolfssl Embedded Firewall/IDPS **5**
1.1 Description . . . 5
1.2 Documentation . . . 5
1.3 Dependencies . . . 5
1.4 Building . . . 6
1.4.1 Build and Self-Test Examples . . . 10
1.5 Project Examples . . . 10

2 Building and Initializing wolfSentry for an application on FreeRTOS/lwIP **12**

3 Configuring wolfSentry using a JSON document **18**
3.1 JSON Basics . . . 18
3.2 JSON load flags . . . 19
3.3 Overview of JSON syntax . . . 19
3.4 Descriptions of elements . . . 21
3.5 Formal ABNF grammar . . . 24

4 wolfSentry Release History and Change Log **30**

5 wolfSentry Release 1.6.3 (January 22, 2025) **30**
5.1 New Features . . . 30
5.2 Noteworthy Changes and Additions . . . 30
5.3 Bug Fixes, Cleanups, and Debugging Aids . . . 30
5.4 Self-Test Enhancements . . . 30

6 wolfSentry Release 1.6.2 (January 2, 2024) **31**
6.1 Noteworthy Changes and Additions . . . 31
6.2 Bug Fixes, Cleanups, and Debugging Aids . . . 31
6.3 Self-Test Enhancements . . . 31

7 wolfSentry Release 1.6.1 (November 18, 2023) **32**
7.1 New Features . . . 32
7.2 Noteworthy Changes and Additions . . . 32
7.3 Bug Fixes, Cleanups, and Debugging Aids . . . 32
7.4 Self-Test Enhancements . . . 33

8 wolfSentry Release 1.6.0 (October 24, 2023) **34**
8.1 New Features . . . 34
8.2 Noteworthy Changes and Additions . . . 34
8.3 Performance Improvements . . . 34
8.4 Documentation . . . 34
8.5 Bug Fixes and Cleanups . . . 35
8.6 Self-Test Enhancements . . . 35

9 wolfSentry Release 1.5.0 (September 13, 2023) **37**
9.1 Noteworthy Changes and Additions . . . 37
9.2 Performance Improvements . . . 37
9.3 Documentation . . . 37
9.4 Bug Fixes and Cleanups . . . 38
9.5 Self-Test Enhancements . . . 38

10 wolfSentry Release 1.4.1 (July 20, 2023) **40**
10.1 Bug Fixes and Cleanups . . . 40

10.2 Self-Test Enhancements . . . 40

11 wolfSentry Release 1.4.0 (July 19, 2023) 41
11.1 New Features . . . 41
11.2 Noteworthy Changes and Additions . . . 41
11.3 Bug Fixes and Cleanups . . . 42
11.4 Self-Test Enhancements . . . 43

12 wolfSentry Release 1.3.1 (July 5, 2023) 44
12.1 Bug Fixes and Cleanups . . . 44
12.2 Self-Test Enhancements . . . 44

13 wolfSentry Release 1.3 (May 19, 2023) 45
13.1 New Features . . . 45
13.1.1 Route dump to JSON . . . 45
13.2 Bug Fixes and Cleanups . . . 45
13.3 Self-Test Enhancements . . . 45

14 wolfSentry Release 1.2.2 (May 4, 2023) 46
14.1 Noteworthy Changes and Additions . . . 46
14.2 Bug Fixes and Cleanups . . . 46
14.3 Self-Test Enhancements . . . 46

15 wolfSentry Release 1.2.1 (Apr 5, 2023) 48
15.1 Noteworthy Changes and Additions . . . 48
15.2 Bug Fixes . . . 48

16 wolfSentry Release 1.2.0 (Mar 24, 2023) 49
16.1 New Features . . . 49
16.1.1 lwIP full firewall integration . . . 49
16.2 Noteworthy Changes and Additions . . . 49
16.3 Bug Fixes . . . 49

17 wolfSentry Release 1.1.0 (Feb 23, 2023) 50
17.1 New Features . . . 50
17.2 Noteworthy Changes and Additions . . . 50
17.3 Bug Fixes . . . 50

18 wolfSentry Release 1.0.0 (Jan 18, 2023) 51
18.1 Noteworthy Changes and Additions . . . 51
18.2 Bug Fixes . . . 51

19 wolfSentry Release 0.8.0 (Jan 6, 2023) 52
19.1 New Features . . . 52
19.1.1 Multithreaded application support . . . 52
19.2 Updated Examples . . . 52
19.2.1 examples/notification-demo . . . 52
19.3 Noteworthy Changes and Additions . . . 52
19.3.1 New public utility APIs, macros, types, etc. . . . 52
19.4 Bug Fixes . . . 53

20 wolfSentry Release 0.7.0 (Nov 7, 2022) 54
20.1 New Features . . . 54
20.1.1 Support for freeform user-defined JSON objects in the “user-values” (key-value pair) section of the config package. . . . 54
20.1.2 Support for setting a user KV as read-only. . . . 54

20.2 Updated Examples . . . 54
20.2.1 examples/notification-demo . . . 54
20.3 Noteworthy Changes and Additions . . . 55
20.3.1 Cleanup of JSON DOM implementation . . . 55
20.3.2 New utility APIs . . . 55
20.4 Bug Fixes . . . 55

21 wolfSentry Release 0.6.0 (Sep 30, 2022) 56
21.1 New Features . . . 56
21.1.1 Core support for automatic penalty boxing, with configurable threshold when derogatory count reaches threshold . . . 56
21.2 Noteworthy Changes and Additions . . . 56
21.3 Bug Fixes . . . 56

22 wolfSentry Release 0.5.0 (Aug 1, 2022) 57
22.1 New Example . . . 57
22.2 Noteworthy Changes . . . 57
22.3 Bug Fixes . . . 57

23 wolfSentry Release 0.4.0 (May 27, 2022) 58
23.1 New Features . . . 58
23.2 Noteworthy Changes . . . 58
23.3 Bug Fixes . . . 58

24 wolfSentry Release 0.3.0 (Dec 30, 2021) 59
24.1 New Ports and Examples . . . 59
24.2 New Features . . . 59
24.3 Bug Fixes . . . 59

1 wolfSentry – The Wolfssl Embedded Firewall/IDPS

1.1 Description

wolfSentry is the wolfSSL embedded IDPS (Intrusion Detection and Prevention System). In simple terms, wolfSentry is an embedded firewall engine (both static and fully dynamic), with prefix-based and wildcard-capable lookup of known hosts/netblocks qualified by interface, address family, protocol, port, and other traffic parameters. Additionally, wolfSentry can be used as a dynamically configurable logic hub, arbitrarily associating user-defined events with user-defined actions, contextualized by connection attributes. The evolution of client-server relationships can thus be tracked in detail, freely passing traffic matching expected usage patterns, while efficiently rejecting abusive traffic.

wolfSentry is fully integrated with the lwIP stack, through a patchset in the `lwip/` subdirectory of the source tree, and has basic integration with the wolfSSL library for application-level filtering of inbound and outbound connections.

The wolfSentry engine is dynamically configurable programmatically through an API, or from a textual input file in JSON supplied to the engine, or dynamically and incrementally with JSON fragments, or any combination of these methods. Reconfiguration is protected by transactional semantics, and advanced internal locks on threaded targets assure seamless service availability with atomic policy transition. Callbacks allow for transport-agnostic remote logging, e.g. through MQTT, syslog, or DDS message buses.

wolfSentry is designed from the ground up to function well in resource-constrained, bare-metal, and realtime environments, with algorithms to stay within designated maximum memory footprints and maintain deterministic throughput. This allows full firewall and IDPS functionality on embedded targets such as FreeRTOS, Nucleus, NUTTX, Zephyr, VxWorks, and Green Hills Integrity, and on ARM and other common embedded CPUs and MCUs. wolfSentry with dynamic firewalling can add as little as 64k to the code footprint, and 32k to the volatile state footprint, and can fully leverage the existing logic and state of applications and sibling libraries.

1.2 Documentation

With doxygen installed, the HTML version of the full API reference manual can be generated from the top of the wolfSentry source tree with `make doc-html`. This, and the source code itself, are the recommended API references.

The PDF version of the API reference manual is pregenerated and included with source distributions in the `doc/` subdirectory at `doc/wolfSentry_refman.pdf`. The latest version is always available on GitHub.

1.3 Dependencies

In its default build, wolfSentry depends on a POSIX runtime, specifically the heap allocator, clock_gettime, stdio, semaphore, pthreads, and string APIs. However, these dependencies can be avoided with various build-time options. The recipe

```
make STATIC=1 SINGLETHREADED=1 NO_STDIO=1 EXTRA_CFLAGS="-DWOLFSENTRY_NO_CLOCK_BUILTIN
-DWOLFSENTRY_NO_MALLOC_BUILTIN"
```

builds a libwolfsentry.a that depends on only a handful of basic string functions and the `inet_ntop()` library function (from POSIX.1-2001, and also implemented by lwIP). Allocator and time callbacks must then be set in a `struct wolfsentry_host_platform_interface` supplied to `wolfsentry_init()`.

The wolfSentry `Makefile` depends on a modern (v4.0+) Gnu `make`. The library itself can be built outside `make`, within another project/framework, by creating a user settings macro file and passing its

path to the compiler with the WOLFSENTRY_USER_SETTINGS_FILE macro.

1.4 Building

wolfSentry was written with portability in mind, with provisions for non-POSIX and C89 targets. For example, all its dependencies can be met with the FreeRTOS/newlib-nano/lwIP runtime. If you have difficulty building wolfSentry, please don't hesitate to seek support through our support forums or contact us directly at support@wolfssl.com.

The current wolfSentry release can be downloaded from the wolfSSL website as a ZIP file, and developers can browse the release history and clone the wolfSentry Git repository for the latest pre-release updates.

There are several flags that can be passed to make to control the build parameters. make will store them at build time in wolfsentry/wolfsentry_options.h in the build tree. If you are not using make, then the C macro WOLFSENTRY_USER_SETTINGS_FILE should be defined to the path to a file containing settings, both when building wolfSentry and when building the application.

The following feature control variables are recognized. True/false features (LWIP, NO_STDIO, NO_JSON, etc.) are undefined by default, and activated when defined. Macros can be supplied using the EXTRA_CFLAGS option, or by placing them in a USER_SETTINGS_FILE. More detailed documentation for macros is available in the reference manual "Startup/Configuration/Shutdown Subsystem" topic.

make Option	Macro Option	Description
SHELL		Supplies an explicit/alternative path to bash.
AWK		Supplies an explicit/alternative path to Gnu awk.
V		Verbose make output e.g. make V=1 -j test
USER_MAKE_CONF		User-defined make clauses to include at the top of the main Makefile e.g. make -j USER_MAKE_CONF=Makefile.settings
EXTRA_CFLAGS		Additional arguments to be passed verbatim to the compiler
EXTRA_LDFLAGS		Additional arguments to be passed verbatim to the linker
SRC_TOP		The source code top level directory (default pwd -P)
BUILD_TOP		Build with artifacts in an alternate location (outside or in a subdirectory of the source tree) e.g. make BUILD_TOP=./build -j test
DEBUG		Compiler debugging flag to use (default -ggdb)
OPTIM		The optimizer flag to use (default -O3)

make Option	Macro Option	Description
HOST		The target host tuple, for cross-compilation (default unset, i.e. native targeting)
RUNTIME		The target runtime ecosystem – default unset, FreeRTOS-lwIP and Linux-lwIP are recognized
C_WARNFLAGS		The warning flags to use (overriding the generally applicable defaults)
STATIC		Build statically linked unit tests
STRIPPED		Strip binaries of debugging symbols
FUNCTION_SECTIONS		Cull any unused object code (with function granularity) to minimize total size.
BUILD_DYNAMIC		Build dynamically linked library
VERY_QUIET		Inhibit all non-error output during build
TAR		Path to GNU tar binary for make dist, should be set to gtar for macOS
VERSION		The version to package for make dist
LWIP	WOLFSENTRY_LWIP	True/false – Activates appropriate build settings for lwIP
NO_STDIO_STREAMS	WOLFSENTRY_NO_STDIO_STREAMS	Define to omit functionality that depends on stdio stream I/O
	WOLFSENTRY_NO_STDIO_H	Define to inhibit inclusion of stdio.h
NO_ADDR_BITMASK_MATCHING	WOLFSENTRY_NO_ADDR_BITMASK_MATCHING	Define to omit support for bitmask matching of addresses, i.e. support only prefix matching.
NO_IPV6	WOLFSENTRY_NO_IPV6	Define to omit support for the IPv6 address family.
NO_JSON	WOLFSENTRY_NO_JSON	Define to omit JSON configuration support
NO_JSON_DOM	WOLFSENTRY_NO_JSON_DOM	Define to omit JSON DOM API
CALL_TRACE	WOLFSENTRY_DEBUG_CALL_TRACE	Define to activate runtime call stack logging (profusely verbose)
USER_SETTINGS_FILE	WOLFSENTRY_USER_SETTINGS_FILE	Substitute settings file, replacing autogenerated wolfsentry_settings.h

make Option	Macro Option	Description
SINGLETHREADED	WOLFSENTRY_SINGLETHREADED	Define to omit thread safety logic, and replace thread safety functions and macros with no-op macros.
	WOLFSENTRY_NO_PROTOCOL_NAMES	If defined, omit APIs for rendering error codes and source code files in human readable form. They will be rendered numerically.
	WOLFSENTRY_NO_GETPROTOBY	Define to disable lookup and rendering of protocols and services by name.
	WOLFSENTRY_NO_ERROR_STRINGS	If defined, omit APIs for rendering error codes and source code files in human readable form. They will be rendered numerically.
	WOLFSENTRY_NO_MALLOC_BUILTINS	If defined, omit built-in heap allocator primitives; the wolfsentry_host_platform_interface supplied to wolfSentry APIs must include implementations of all functions in struct wolfsentry_allocator.
	WOLFSENTRY_HAVE_NONGNU_ATOMICS	Define if gnu-style atomic intrinsics are not available. WOLFSENTRY_ATOMIC_*() macro definitions for intrinsics will need to be supplied in WOLFSENTRY_USER_SETTINGS_FILE (see wolfsentry_util.h).
	WOLFSENTRY_NO_CLOCK_BUILTIN	If defined, omit built-in time primitives; the wolfsentry_host_platform_interface supplied to wolfSentry APIs must include implementations of all functions in struct wolfsentry_timecbs.

make Option	Macro Option	Description
	WOLFSENTRY_NO_SEM_BUILTIN	If defined, omit built-in semaphore primitives; the wolfsentry_host_platform_interface supplied to wolfSentry APIs must include implementations of all functions in struct wolfsentry_semcbs.
	WOLFSENTRY_USE_NONPOSIX_SEMAPHORES	Define if POSIX semaphore API is not available. If no non-POSIX builtin implementation is present in wolfsentry_util.c, then #WOLFSENTRY_NO_SEM_BUILTIN must be set, and the wolfsentry_host_platform_interface supplied to wolfSentry APIs must include a full semaphore implementation (shim set) in its wolfsentry_semcbs slot.
	WOLFSENTRY_SEMAPHORE_INCLUDE	Define to the path of a header file declaring a semaphore API.
	WOLFSENTRY_USE_NONPOSIX_THREADS	Define if POSIX thread API is not available. WOLFSENTRY_THREAD_INCLUDE, WOLFSENTRY_THREAD_ID_T, and WOLFSENTRY_THREAD_GET_ID_HANDLER will need to be defined.
	WOLFSENTRY_THREAD_INCLUDE	Define to the path of a header file declaring a threading API.
	WOLFSENTRY_THREAD_ID_T	Define to the appropriate type analogous to POSIX pthread_t.
	WOLFSENTRY_THREAD_GET_ID_HANDLER	Define to the name of a void function analogous to POSIX pthread_self, returning a value of type WOLFSENTRY_THREAD_ID_T.
	FREERTOS	Build for FreeRTOS

1.4.1 Build and Self-Test Examples

Building and testing libwolfsentry.a on Linux:

```
make -j test
```

Build verbosely:

```
make V=1 -j test
```

Build with artifacts in an alternate location (outside or in a subdirectory of the source tree):

```
make BUILD_TOP=./build -j test
```

Install from an alternate build location to a non-standard destination:

```
make BUILD_TOP=./build INSTALL_DIR=/usr INSTALL_LIBDIR=/usr/lib64 install
```

Build libwolfsentry.a and test it in various configurations:

```
make -j check
```

Build and test libwolfsentry.a without support for multithreading:

```
make -j SINGLETHREADED=1 test
```

Other available make flags are `STATIC=1`, `STRIPPED=1`, `NO_JSON=1`, and `NO_JSON_DOM=1`, and the defaults values for `DEBUG`, `OPTIM`, and `C_WARNFLAGS` can also be usefully overridden.

Build with a user-supplied makefile preamble to override defaults:

```
make -j USER_MAKE_CONF=Makefile.settings
```

(`Makefile.settings` can contain simple settings like `OPTIM := -Os`, or elaborate makefile code including additional rules and dependency mechanisms.)

Build the smallest simplest possible library:

```
make -j SINGLETHREADED=1 NO_STDIO=1 DEBUG= OPTIM=-Os EXTRA_CFLAGS="-DWOLFSENTRY_NO_CLOCK_BUIL
-DWOLFSENTRY_NO_MALLOC_BUILTIN -DWOLFSENTRY_NO_ERROR_STRINGS -Wno-error=inline
-Wno-inline"
```

Build and test with user settings:

```
make -j USER_SETTINGS_FILE=user_settings.h test
```

Build for FreeRTOS on ARM32, assuming FreeRTOS and lwIP source trees are located as shown:

```
make -j HOST=arm-none-eabi RUNTIME=FreeRTOS-lwIP FREERTOS_TOP=../third/FreeRTOSv202212.00
LWIP_TOP=../third/lwip EXTRA_CFLAGS=-mcpu=cortex-m7
```

1.5 Project Examples

In the `wolfsentry/examples/` subdirectory are a set of example ports and applications, including a demo pop-up notification system implementing a toy TLS-enabled embedded web server, integrating with the Linux D-Bus facility.

More comprehensive examples of API usage are in `tests/unittests.c`, particularly `test_static_routes()`, `test_dynamic_rules()`, and `test_json()`, and the JSON configuration files at `tests/test-config*.json`.

In the wolfSSL repository, see code in `wolfssl/test.h` gated on `WOLFSSL_WOLFSENTRY_HOOKS`, including `wolfsentry_store_endpoints()`, `wolfSentry_NetworkFilterCallback()`, `wolfsentry_setup()`, and `tcp_connect_with_wolfSentry()`. See also code in `examples/server/server.c` and `examples/client/client.c` gated on `WOLFSSL_WOLFSENTRY_HOOKS`. Configure wolfssl with `--enable-wolfsentry` to build with wolfSentry integration, and use `--with-wolfsentry=/the/install/path`

if wolfSentry is installed in a nonstandard location. The wolfSSL test client/server can be loaded with user-supplied wolfSentry JSON configurations from the command line, using `--wolfsentry-config <file>`.

2 Building and Initializing wolfSentry for an application on FreeRTOS/lwIP

Building the wolfSentry library for FreeRTOS with lwIP and newlib-nano is supported directly by the top level `Makefile`. E.g., for an ARM Cortex M7, `libwolfsentry.a` can be built with

```
make HOST=arm-none-eabi EXTRA_CFLAGS='-mcpu=cortex-m7' RUNTIME=FreeRTOS-lwIP
   FREERTOS_TOP="$FREERTOS_TOP" LWIP_TOP="$LWIP_TOP"
```

`FREERTOS_TOP` is the path to the top of the FreeRTOS distribution, with `FreeRTOS/Source` directly under it, and `LWIP_TOP` is the path to the top of the lwIP distribution, with `src` directly under it.

The below code fragments can be added to a FreeRTOS application to enable wolfSentry with dynamically loaded policies (JSON). Many of the demonstrated code patterns are optional. The only calls that are indispensable are `wolfsentry_init()`, `wolfsentry_config_json_oneshot()`, and `wolfsentry_install_lwip_filter_callbacks()`. Each of these also has API variants that give the user more control.

```
#define WOLFSENTRY_SOURCE_ID WOLFSENTRY_SOURCE_ID_USER_BASE
#define WOLFSENTRY_ERROR_ID_USER_APP_ERR0 (WOLFSENTRY_ERROR_ID_USER_BASE-1)
 /* user-defined error IDs count down starting at
    WOLFSENTRY_ERROR_ID_USER_BASE (which is negative). */

#include <wolfsentry/wolfsentry_json.h>
#include <wolfsentry/wolfsentry_lwip.h>

static struct wolfsentry_context *wolfsentry_lwip_ctx = NULL;

static const struct wolfsentry_eventconfig demo_config = {
#ifdef WOLFSENTRY_HAVE_DESIGNATED_INITIALIZERS
        .route_private_data_size = 64,
        .route_private_data_alignment = 0,         /* default alignment --
           same as sizeof(void *). */
        .max_connection_count = 10,                /* by default, don't allow
           more than 10 simultaneous
                                                    * connections that match
                                                       the same route.
                                                    */
        .derogatory_threshold_for_penaltybox = 4,  /* after 4 derogatory
           events matching the same route,
                                                    * put the route in penalty
                                                       box status.
                                                    */
        .penaltybox_duration = 300,                /* keep routes in penalty
           box status for 5 minutes.
                                                    * denominated in seconds
                                                       when passing to
                                                    * wolfsentry_init().
                                                    */
        .route_idle_time_for_purge = 0,            /* 0 to disable --
           autopurge doesn't usually make
                                                    * much sense as a default
                                                       config.
                                                    */
```

```
        .flags = WOLFSENTRY_EVENTCONFIG_FLAG_COMMENDABLE_CLEARS_DEROGATORY, /*
            automatically clear
                                                    * derogatory count for a
                                                       route when a
                                                       commendable
                                                    * event matches the route.
                                                    */
        .route_flags_to_add_on_insert = 0,
        .route_flags_to_clear_on_insert = 0,
        .action_res_filter_bits_set = 0,
        .action_res_filter_bits_unset = 0,
        .action_res_bits_to_add = 0,
        .action_res_bits_to_clear = 0
#else
        64,
        0,
        10,
        4,
        300,
        0,
        WOLFSENTRY_EVENTCONFIG_FLAG_COMMENDABLE_CLEARS_DEROGATORY,
        0,
        0,
        0,
        0,
        0,
        0
#endif
    };

/* This routine is to be called once by the application before any direct
   calls
 * to lwIP -- i.e., before lwip_init() or tcpip_init().
 */
wolfsentry_errcode_t activate_wolfsentry_lwip(const char *json_config, int
   json_config_len)
{
    wolfsentry_errcode_t ret;
    char err_buf[512]; /* buffer for detailed error messages from
                        * wolfsentry_config_json_oneshot().
                        */

    /* Allocate a thread state struct on the stack.  Note that the final
     * semicolon is supplied by the macro definition, so that in single-
        threaded
     * application builds this expands to nothing at all.
     */
    WOLFSENTRY_THREAD_HEADER_DECLS

    if (wolfsentry_lwip_ctx != NULL) {
        printf("activate_wolfsentry_lwip() called multiple times.\n");
        WOLFSENTRY_ERROR_RETURN(ALREADY);
    }
```

```
#ifdef WOLFSENTRY_ERROR_STRINGS
    /* Enable pretty-printing of the app source code filename for
     * WOLFSENTRY_ERROR_FMT/WOLFSENTRY_ERROR_FMT_ARGS().
     */
    ret = WOLFSENTRY_REGISTER_SOURCE();
    WOLFSENTRY_RERETURN_IF_ERROR(ret);

    /* Enable pretty-printing of an app-specific error code. */
    ret = WOLFSENTRY_REGISTER_ERROR(USER_APP_ERR0, "failure in application
       code");
    WOLFSENTRY_RERETURN_IF_ERROR(ret);
#endif

    /* Initialize the thread state struct -- this sets the thread ID. */
    WOLFSENTRY_THREAD_HEADER_INIT_CHECKED(WOLFSENTRY_THREAD_FLAG_NONE);

    /* Call the main wolfSentry initialization routine.
     *
     * WOLFSENTRY_CONTEXT_ARGS_OUT() is a macro that abstracts away
     * conditionally passing the thread struct pointer to APIs that need it.
        If
     * this is a single-threaded build (!defined(WOLFSENTRY_THREADSAFE)), then
     * the thread arg is omitted entirely.
     *
     * WOLFSENTRY_CONTEXT_ARGS_OUT_EX() is a variant that allows the caller to
     * supply the first arg explicitly, when "wolfsentry" is not the correct
        arg
     * to pass.  This is used here to pass a null pointer for the host
        platform
     * interface ("hpi").
     */
    ret = wolfsentry_init(
        wolfsentry_build_settings,
        WOLFSENTRY_CONTEXT_ARGS_OUT_EX(NULL /* hpi */),
        &demo_config,
        &wolfsentry_lwip_ctx);
    if (ret < 0) {
        printf("wolfsentry_init() failed: " WOLFSENTRY_ERROR_FMT "\n",
               WOLFSENTRY_ERROR_FMT_ARGS(ret));
        goto out;
    }

    /* Insert user-defined actions here, if any. */
    ret = wolfsentry_action_insert(
        WOLFSENTRY_CONTEXT_ARGS_OUT_EX(wolfsentry_lwip_ctx),
        "my-action",
        WOLFSENTRY_LENGTH_NULL_TERMINATED,
        WOLFSENTRY_ACTION_FLAG_NONE,
        my_action_handler,
        NULL,
        NULL);
    if (ret < 0) {
        printf("wolfsentry_action_insert() failed: " WOLFSENTRY_ERROR_FMT "\n
           ",
```

```
                WOLFSENTRY_ERROR_FMT_ARGS(ret));
        goto out;
    }

    if (json_config) {
        if (json_config_len < 0)
            json_config_len = (int)strlen(json_config);

        /* Do the initial load of the policy. */
        ret = wolfsentry_config_json_oneshot(
            WOLFSENTRY_CONTEXT_ARGS_OUT_EX(wolfsentry_lwip_ctx),
            (unsigned char *)json_config,
            (size_t)json_config_len,
            WOLFSENTRY_CONFIG_LOAD_FLAG_NONE,
            err_buf,
            sizeof err_buf);
        if (ret < 0) {
            printf("wolfsentry_config_json_oneshot() failed: %s\n", err_buf);
            goto out;
        }
    } /* else the application will need to set up the policy programmatically,
       * or itself call wolfsentry_config_json_oneshot() or sibling APIs.
       */

    /* Install lwIP callbacks.  Once this call returns with success, all lwIP
     * traffic designated for filtration by the mask arguments shown below
         will
     * be subject to filtering (or other supplementary processing) according
         to
     * the policy loaded above.
     *
     * Note that if a given protocol is gated out of LWIP, its mask argument
     * must be passed as zero here, else the call will return
     * IMPLEMENTATION_MISSING error will occur.
     *
     * The callback installation also registers a cleanup routine that will be
     * called automatically by wolfsentry_shutdown().
     */

#define LWIP_ALL_EVENTS (                       \
        (1U << FILT_BINDING) |                  \
        (1U << FILT_DISSOCIATE) |               \
        (1U << FILT_LISTENING) |                \
        (1U << FILT_STOP_LISTENING) |           \
        (1U << FILT_CONNECTING) |               \
        (1U << FILT_ACCEPTING) |                \
        (1U << FILT_CLOSED) |                   \
        (1U << FILT_REMOTE_RESET) |             \
        (1U << FILT_RECEIVING) |                \
        (1U << FILT_SENDING) |                  \
        (1U << FILT_ADDR_UNREACHABLE) |         \
        (1U << FILT_PORT_UNREACHABLE) |         \
        (1U << FILT_INBOUND_ERR) |              \
        (1U << FILT_OUTBOUND_ERR))
```

```
    ret = wolfsentry_install_lwip_filter_callbacks(
        WOLFSENTRY_CONTEXT_ARGS_OUT_EX(wolfsentry_lwip_ctx),

#if LWIP_ARP || LWIP_ETHERNET
        LWIP_ALL_EVENTS, /* ethernet_mask */
#else
        0,
#endif
#if LWIP_IPV4 || LWIP_IPV6
        LWIP_ALL_EVENTS, /* ip_mask */
#else
        0,
#endif
#if LWIP_ICMP || LWIP_ICMP6
        LWIP_ALL_EVENTS, /* icmp_mask */
#else
        0,
#endif
#if LWIP_TCP
        LWIP_ALL_EVENTS, /* tcp_mask */
#else
        0,
#endif
#if LWIP_UDP
        LWIP_ALL_EVENTS /* udp_mask */
#else
        0
#endif
        );
    if (ret < 0) {
        printf("wolfsentry_install_lwip_filter_callbacks: "
               WOLFSENTRY_ERROR_FMT "\n", WOLFSENTRY_ERROR_FMT_ARGS(ret));
    }

out:
    if (ret < 0) {
        /* Clean up if initialization failed. */
        wolfsentry_errcode_t shutdown_ret =
            wolfsentry_shutdown(WOLFSENTRY_CONTEXT_ARGS_OUT_EX(&
               wolfsentry_lwip_ctx));
        if (shutdown_ret < 0)
            printf("wolfsentry_shutdown: "
                   WOLFSENTRY_ERROR_FMT "\n", WOLFSENTRY_ERROR_FMT_ARGS(
                       shutdown_ret));
    }

    WOLFSENTRY_THREAD_TAILER_CHECKED(WOLFSENTRY_THREAD_FLAG_NONE);

    WOLFSENTRY_ERROR_RERETURN(ret);
}

/* to be called once by the application after any final calls to lwIP. */
wolfsentry_errcode_t shutdown_wolfsentry_lwip(void)
```

```
{
    wolfsentry_errcode_t ret;
    if (wolfsentry_lwip_ctx == NULL) {
        printf("shutdown_wolfsentry_lwip() called before successful activation
            .\n");
        return -1;
    }

    /* after successful shutdown, wolfsentry_lwip_ctx will once again be a
       null
     * pointer as it was before init.
     */
    ret = wolfsentry_shutdown(WOLFSENTRY_CONTEXT_ARGS_OUT_EX4(&
       wolfsentry_lwip_ctx, NULL));
    if (ret < 0) {
        printf("wolfsentry_shutdown: "
               WOLFSENTRY_ERROR_FMT "\n", WOLFSENTRY_ERROR_FMT_ARGS(ret));
    }

    return ret;
}
```

3 Configuring wolfSentry using a JSON document

Most of the capabilities of wolfSentry can be configured, and dynamically reconfigured, by supplying JSON documents to the library. To use this capability, add the following to wolfSentry initialization in the application:

```
#include <wolfsentry/wolfsentry_json.h>
```

After initialization and installation of application-supplied callbacks (if any), call one of the APIs to load the config:

- `wolfsentry_config_json_oneshot()`
- `wolfsentry_config_json_oneshot_ex()`, with an additional `json_config` arg for fine control of JSON parsing (see `struct JSON_CONFIG` in `wolfsentry/centijson_sax.h`)
- streaming API:
  - `wolfsentry_config_json_init()` or `wolfsentry_config_json_init_ex()`
  - `wolfsentry_config_json_feed()`
  - `wolfsentry_config_json_fini()`

See `wolfsentry/wolfsentry_json.h` for details on arguments.

3.1 JSON Basics

wolfSentry configuration uses standard JSON syntax as defined in RFC 8259, as restricted by RFC 7493, with certain additional requirements. In particular, certain sections in the JSON document are restricted in their sequence of appearance.

- `"wolfsentry-config-version"` shall appear first, and each event definition shall appear before any definitions for events, routes, or default policies that refer to it through `"aux-parent-event"`, `"parent-event"`, or `"default-event"` clauses.
- Within event definitions, the `"label"`, `"priority"`, and `"config"` elements shall appear before any other elements.

These sequence constraints are necessary to allow for high efficiency SAX-style (sequential-incremental) loading of the configuration.

All wildcard flags are implicitly set on routes, and are cleared for fields with explicit assignments in the configuration. For example, if a route designates a particular `"family"`, then `WOLFSENTRY_ROUTE_FLAG_SA_FAMILY_WILDCARD` will be implicitly cleared. Thus, wildcard flags need not be explicitly set or cleared in route definitions.

Note that certain element variants may be unavailable due to build settings:

- `address_family_name`: available if `defined(WOLFSENTRY_PROTOCOL_NAMES)`
- `route_protocol_name`: available if `!defined(WOLFSENTRY_NO_GETPROTOBY)`
- `address_port_name`: available if `!defined(WOLFSENTRY_NO_GETPROTOBY)`
- `json_value_clause`: available if `defined(WOLFSENTRY_HAVE_JSON_DOM)`

Caller-supplied event and action labels shall not begin with `WOLFSENTRY_BUILTIN_LABEL_PREFIX` (by default `"%"`), as these are reserved for built-ins.

`"config-update"` allows the default configuration to be updated. It is termed an “update” because wolfSentry is initially configured by the `config` argument to `wolfsentry_init()` (which can be passed in `NULL`, signifying built-in defaults). Note that times (`wolfsentry_eventconfig.penaltybox_duration` and `wolfsentry_eventconfig.route_idle_time_for_purge`) shall be passed to `wolfsentry_init()` denominated in seconds, notwithstanding the `wolfsentry_time_t` type of the members.

3.2 JSON load flags

The `flags` argument to `wolfsentry_config_json_init()` and `wolfsentry_config_json_oneshot()`, constructed by bitwise-or, changes the way the JSON is processed, as follows:

- `WOLFSENTRY_CONFIG_LOAD_FLAG_NONE` – Not a flag, but all-zeros, signifying default behavior: The wolfSentry core is locked, the current configuration is flushed, and the new configuration is loaded incrementally. Any error during load leaves wolfSentry in an undefined state that can be recovered with a subsequent flush and load that succeeds.
- `WOLFSENTRY_CONFIG_LOAD_FLAG_NO_FLUSH` – Inhibit initial flush of configuration, to allow incremental load. Error during load leaves wolfSentry in an undefined state that can only be recovered with a subsequent flush and load that succeeds, unless `WOLFSENTRY_CONFIG_LOAD_FLAG_DRY_RUN` or `WOLFSENTRY_CONFIG_LOAD_FLAG_LOAD_THEN_COMMIT` was also supplied.
- `WOLFSENTRY_CONFIG_LOAD_FLAG_DRY_RUN` – Load into a temporary configuration, and deallocate before return. Running configuration is unchanged.
- `WOLFSENTRY_CONFIG_LOAD_FLAG_LOAD_THEN_COMMIT` – Load into a newly allocated configuration, and install it only if load completes successfully. On error, running configuration is unchanged. On success, the old configuration is deallocated.
- `WOLFSENTRY_CONFIG_LOAD_FLAG_NO_ROUTES_OR_EVENTS` – Inhibit loading of `"routes"` and `"events"` sections in the supplied JSON.
- `WOLFSENTRY_CONFIG_LOAD_FLAG_FLUSH_ONLY_ROUTES` – At beginning of load process, retain all current configuration except for routes, which are flushed. This is convenient in combination with `wolfsentry_route_table_dump_json_*()` for save/restore of dynamically added routes.
- `WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_DUPKEY_ABORT` – When processing user-defined JSON values, abort load on duplicate keys.
- `WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_DUPKEY_USEFIRST` – When processing user-defined JSON values, for any given key in an object use the first occurrence encountered.
- `WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_DUPKEY_USELAST` – When processing user-defined JSON values, for any given key in an object use the last occurrence encountered.
- `WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_MAINTAINDICTORDER` – When processing user-defined JSON values, store sequence information so that subsequent calls to `wolfsentry_kv_render_value()` or `json_dom_dump(..., JSON_DOM_DUMP_PREFERDICTORDER)` render objects in their supplied sequence, rather than lexically sorted.

Note that `WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_*` flags are allowed only if `WOLFSENTRY_HAVE_JSON_DOM` is defined in the build, as it is with default settings.

3.3 Overview of JSON syntax

Below is a JSON “lint” pseudodocument demonstrating all available configuration nodes, with value specifiers that refer to the ABNF definitions below. The allowed values are as in the ABNF formal syntax later in this document.

```
{
    "wolfsentry-config-version" : 1,
    "config-update" : {
        "max-connection-count" : uint32,
        "penalty-box-duration" : duration,
        "route-idle-time-for-purge" : duration,
```

```
        "derog-thresh-for-penalty-boxing" : uint16,
        "derog-thresh-ignore-commendable" : boolean,
        "commendable-clears-derogatory" : boolean,
        "route-flags-to-add-on-insert" : route_flag_list,
        "route-flags-to-clear-on-insert" : route_flag_list,
        "action-res-filter-bits-set" : action_res_flag_list,
        "action-res-filter-bits-unset" : action_res_flag_list,
        "action-res-bits-to-add" : action_res_flag_list,
        "action-res-bits-to-clear" : action_res_flag_list,
        "max-purgeable-routes" : uint32,
        "max-purgeable-idle-time" : duration
    },
    "events" : [
       { "label" : label,
         "priority" : uint16,
         "config" : {
            "max-connection-count" : uint32,
            "penalty-box-duration" : duration,
            "route-idle-time-for-purge" : duration,
            "derog-thresh-for-penalty-boxing" : uint16,
            "derog-thresh-ignore-commendable" : boolean,
            "commendable-clears-derogatory" : boolean,
            "route-flags-to-add-on-insert" : route_flag_list,
            "route-flags-to-clear-on-insert" : route_flag_list,
            "action-res-filter-bits-set" : action_res_flag_list,
            "action-res-filter-bits-unset" : action_res_flag_list,
            "action-res-bits-to-add" : action_res_flag_list,
            "action-res-bits-to-clear" : action_res_flag_list
         },
         "aux-parent-event"  : label,
         "post-actions" : action_list,
         "insert-actions" : action_list,
         "match-actions" : action_list,
         "update-actions" : action_list,
         "delete-actions" : action_list,
         "decision-actions" : action_list
      }
    ],
    "default-policies" : {
        "default-policy" : default_policy_value,
        "default-event" ":" label
    },
    "routes" : [
      {
        "parent-event" : label,
        "af-wild" : boolean,
        "raddr-wild" : boolean,
        "rport-wild" : boolean,
        "laddr-wild" : boolean,
        "lport-wild" : boolean,
        "riface-wild" : boolean,
        "liface-wild" : boolean,
        "tcplike-port-numbers" : boolean,
        "direction-in" : boolean,
```

```
        "direction-out" : boolean,
        "penalty-boxed" : boolean,
        "green-listed" : boolean,
        "dont-count-hits" : boolean,
        "dont-count-current-connections" : boolean,
        "port-reset" : boolean,

        "family" : address_family,
        "protocol" : route_protocol,
        "remote" : {
          "interface" : uint8,
          "address" : route_address,
          "prefix-bits" : uint16,
          "bitmask" : route_address,
          "port" : endpoint_port
        },
        "local" : {
          "interface" : uint8,
          "address" : route_address,
          "prefix-bits" : uint16,
          "bitmask" : route_address,
          "port" : endpoint_port
        }
      }
    ],
    "user-values" : {
      label : null,
      label : true,
      label : false,
      label : number_sint64,
      label : number_float,
      label : string,
      label : { "uint" : number_uint64 },
      label : { "sint" : number_sint64 },
      label : { "float" : number_float },
      label : { "string" : string_value },
      label : { "base64" : base64_value },
      label : { "json" : json_value }
    }
}
```

3.4 Descriptions of elements

`wolfsentry-config-version` – Shall appear first, with the value 1.

`config-update` – Sets default and global parameters. The default parameters apply to routes that have no parent event, or a parent event with no config of its own.

- `max-connection-count` – If nonzero, the concurrent connection limit, beyond which additional connection requests are rejected.
- `penalty-box-duration` – If nonzero, the duration that a route stays in penalty box status before automatic release.
- `derog-thresh-for-penalty-boxing` – If nonzero, the threshold at which accumulated

derogatory counts (from WOLFSENTRY_ACTION_RES_DEROGATORY incidents) automatically penalty boxes a route.

- derog-thresh-ignore-commendable – If true, then counts from WOLFSENTRY_ACTION_RES_COMMENDABLE are not subtracted from the derogatory count when checking for automatic penalty boxing.
- commendable-clears-derogatory – If true, then each count from WOLFSENTRY_ACTION_RES_COMMENDABLE zeroes the derogatory count.
- max-purgeable-routes – Global limit on the number of ephemeral routes to allow in the route table, beyond which the least recently matched ephemeral route is forced out early. Not allowed in config clauses of events.
- max-purgeable-idle-time – Global absolute maximum idle time for ephemeral routes, controlling purges of stale (expired) ephemeral routes with nonzero wolfsentry_route_metadata_exports.con Default is no limit. Not allowed in config clauses of events.
- route-idle-time-for-purge – If nonzero, the time after the most recent dispatch match for a route to be garbage-collected. Useful primarily in config clauses of events (see events below).
- route-flags-to-add-on-insert – List of route flags to set on new routes upon insertion. Useful primarily in config clauses of events (see events below).
- route-flags-to-clear-on-insert – List of route flags to clear on new routes upon insertion. Useful primarily in config clauses of events (see events below).
- action-res-filter-bits-set – List of action_res flags that must be set at lookup time (dispatch) for referring routes to match. Useful primarily in config clauses of events (see events below).
- action-res-filter-bits-unset – List of action_res flags that must be clear at lookup time (dispatch) for referring routes to match. Useful primarily in config clauses of events (see events below).
- action-res-bits-to-add – List of action_res flags to be set upon match.
- action-res-bits-to-clear – List of action_res flags to be cleared upon match.

events – The list of events with their respective definitions. This section can appear more than once, but any given event definition shall precede any definitions that refer to it.

Each event is composed of the following elements, all of which are optional except for label. label, priority, and config shall appear before the other elements.

- label – The name by which the event is identified. See the definition of label in the ABNF grammar below for permissible values.
- priority – The priority of routes that have this event as their parent-event (see routes below). Lower number means higher priority.
- config – The configuration to associate with routes with this parent-event, as above for config-update.
- aux-parent-event – An event reference for use by action handlers, e.g. built-in "%track-peer-v1" creates routes with aux-parent-event as the new route's parent-event.
- post-actions – List of actions to take when this event is passed via event_label to a dispatch routine such as wolfsentry_route_event_dispatch().
- insert-actions – List of actions to take when a route is inserted with this event as parent-event.
- match-actions – List of actions to take when a route is matched by a dispatch routine, and the route has this event as its parent-event.

- update-actions – List of actions to take when a route has a status update, such as a change of penalty box status, and has this event as its parent-event.
- delete-actions – List of actions to take when a route is deleted, and has this event as its parent-event.
- decision-actions – List of actions to take when dispatch final decision (final value of action_results) is determined, and the matched route has this event as its parent-event.

default-policies – The global fallthrough default policies for dispatch routines such as wolfsentry_route_event_dispatch().

- default-policy – A simple action_result flag to set by default, either **accept**, **reject**, or **reset**, the latter of which causes generation of TCP reset and ICMP unreachable reply packets where relevant.
- default-event – An event to use when a dispatch routine is called with a null event_label.

routes – The list of routes with their respective definitions. This section can appear more than once.

Each route is composed of the following elements, all of which are optional.

- parent-event – The event whose attributes determine the dynamics of the route.
- family – The address family to match. See address_family definition in the ABNF grammar below for permissible values.
- protocol – The protocol to match. See route_protocol definition in the ABNF grammar below for permissible values.
- remote – The attributes to match for the remote endpoint of the traffic.
  - interface – Network interface ID, as an arbitrary integer chosen and used consistently by the caller or IP stack integration.
  - address – The network address, in idiomatic form. IPv4, IPv6, and MAC addresses shall enumerate all octets. See route_address definition in the ABNF grammar below for permissible values.
  - prefix-bits – The number of bits in the address that traffic must match (mutually exclusive with bitmask).
  - bitmask – A bitmask to be applied to the traffic address before matching with the route address (mutually exclusive with prefix-bits).
  - port – The port number that traffic must match.
- local – The attributes to match for the local endpoint of the traffic. The same nodes are available as for remote.
- direction-in – If true, match inbound traffic.
- direction-out – If true, match outbound traffic.
- penalty-boxed – If true, traffic matching the route is penalty boxed (rejected or reset).
- green-listed – If true, traffic matching the route is accepted.
- dont-count-hits – If true, inhibit statistical bookkeeping (no effect on dynamics).
- dont-count-current-connections – If true, inhibit tracking of concurrent connections, so that max-connection-count has no effect on traffic matching this route.
- port-reset – If true, set the WOLFSENTRY_ACTION_RES_PORT_RESET flag in the action_results when this route is matched, causing TCP reset or ICMP unreachable reply packet to be generated if IP stack integration is activated (e.g. wolfsentry_install_lwip_filter_callbacks()).

user-values – One or more sections of fully user-defined data available to application code for any use. Each key is a label as defined in the ABNF grammar below. The value can be any of:

- `null`
- `true`
- `false`
- an integral number, implicitly a signed 64 bit integer
- a floating point number, as defined in the ABNF grammar below for `number_float`
- a quoted string allowing standard JSON escapes
- any of several explicitly typed constructs, with values as defined in the ABNF grammar below.
  - `{ "uint" : number_uint64 }`
  - `{ "sint" : number_sint64 }`
  - `{ "float" : number_float }`
  - `{ "string" : string_value }`
  - `{ "base64" : base64_value }`
  - `{ "json" : json_value }`

3.5 Formal ABNF grammar

Below is the formal ABNF definition of the configuration syntax and permitted values.

This definition uses ABNF syntax as prescribed in RFC 5234 and 7405, except:

- Whitespace is ignored, as provided in RFC 8259.
- a `-` operator is added, accepting a quoted literal string or a group of literal characters, to provide for omitted character(s) in the target text (here, trailing comma separators) by performing all notional matching operations of the containing group up to that point with the target text notionally extended with the argument to the operator.

The length limits used in the definition assume the default values in wolfsentry_settings.h, 32 octets for labels (`WOLFSENTRY_MAX_LABEL_BYTES`), and 16384 octets for user-defined values (`WOLFSENTRY_KV_MAX_VALUE_BYTES`). These values can be overridden at build time with user-supplied values.

```
"{"
    DQUOTE %s"wolfsentry-config-version" DQUOTE ":" uint32
    [ "," DQUOTE %s"config-update" DQUOTE ":" top_config_list "," ]
    *("," DQUOTE %s"events" ":" "["
       event *("," event)
    "]")
    [ "," DQUOTE %s"default-policies" DQUOTE ":" "{"
        default_policy_item *("," default_policy_item)
    "}" ]
    *("," DQUOTE %s"routes" DQUOTE ":" "["
        route *("," route)
    "]")
    *("," DQUOTE %s"user-values" DQUOTE ":" "{"
        user_item *("," user_item)
    "}")
"}"

event = "{" label_clause
        [ "," priority_clause ]
        [ "," event_config_clause ]
        [ "," aux_parent_event_clause ]
        *("," action_list_clause) "}"

default_policy_item =
```

```
        (DQUOTE %s"default-policy" DQUOTE ":" default_policy_value) /
        (DQUOTE %s"default-event" DQUOTE ":" label)

default_policy_value = (%s"accept" / %s"reject" / %s"reset")

label_clause = DQUOTE %s"label" DQUOTE ":" label

priority_clause = DQUOTE %s"priority" DQUOTE ":" uint16

event_config_clause = DQUOTE %s"config" DQUOTE ":" event_config_list

aux_parent_event_clause = DQUOTE %s"aux-parent-event" DQUOTE ":" label

action_list_clause = DQUOTE (%s"post-actions" / %s"insert-actions" / %s"match-
   actions"
            / %s"update-actions" / %s"delete-actions" / %s"decision-actions")
               DQUOTE
            ":" action_list

action_list = "[" label *("," label) "]"

event_config_list = "{" event_config_item *("," event_config_item) "}"

top_config_list = "{" top_config_item *("," top_config_item) "}"

top_config_item = event_config_item / max_purgeable_routes_clause /
   max_purgeable_idle_time_clause

event_config_item =
  (DQUOTE %s"max-connection-count" DQUOTE ":" uint32) /
  (DQUOTE %s"penalty-box-duration" DQUOTE ":" duration) /
  (DQUOTE %s"route-idle-time-for-purge" DQUOTE ":" duration) /
  (DQUOTE %s"derog-thresh-for-penalty-boxing" DQUOTE ":" uint16 /
  (DQUOTE %s"derog-thresh-ignore-commendable" DQUOTE ":" boolean /
  (DQUOTE %s"commendable-clears-derogatory" DQUOTE ":" boolean /
  (DQUOTE (%s"route-flags-to-add-on-insert" / %s"route-flags-to-clear-on-
     insert") DQUOTE ":" route_flag_list) /
  (DQUOTE (%s"action-res-filter-bits-set" / %s"action-res-filter-bits-unset" /
      %s"action-res-bits-to-add" / %s"action-res-bits-to-clear") DQUOTE ":"
     action_res_flag_list)

duration = number_sint64 / (DQUOTE number_sint64 [ %s"d" / %s"h" / %s"m" / %s"
   s" ] DQUOTE)

max_purgeable_routes_clause = DQUOTE %s"max-purgeable-routes" DQUOTE ":"
   uint32

max_purgeable_idle_time_clause = DQUOTE %s"max-purgeable-idle-time" DQUOTE ":"
    duration

route_flag_list = "[" route_flag *("," route_flag) "]"

action_res_flag_list = "[" action_res_flag *("," action_res_flag) "]"
```

```
route = "{"
    [ parent_event_clause "," ]
    *(route_flag_clause ",")
    [ family_clause ","
      [ route_protocol_clause "," ]
    ]
    [ route_remote_endpoint_clause "," ]
    [ route_local_endpoint_clause "," ]
    -","
"}"

parent_event_clause = DQUOTE %s"parent-event" DQUOTE ":" label
route_flag_clause = route_flag ":" boolean
family_clause = DQUOTE %s"family" DQUOTE ":" address_family
route_protocol_clause = DQUOTE %s"protocol" DQUOTE ":" route_protocol

route_remote_endpoint_clause = DQUOTE %s"remote" DQUOTE ":" route_endpoint
route_local_endpoint_clause = DQUOTE %s"local" DQUOTE ":" route_endpoint

route_endpoint = "{"
    [ route_interface_clause "," ]
    [ route_address_clause ","
      [ (route_address_prefix_bits_clause / route_address_bitmask_clause) ","
        ]
    ]
    [ route_port_clause "," ]
    -","
"}"

route_interface_clause = DQUOTE %s"interface" DQUOTE ":" uint8

route_address_clause = DQUOTE %s"address" DQUOTE ":" route_address

route_address_bitmask_clause = DQUOTE %s"bitmask" DQUOTE ":" route_address

route_address = DQUOTE (route_address_ipv4 / route_address_ipv6 /
   route_address_mac / route_address_user) DQUOTE

route_address_ipv4 = uint8 3*3("." uint8)

route_address_ipv6 = < IPv6address from RFC 5954 section 4.1 >

route_address_mac = 1*2HEXDIG ( 5*5(":" 1*2HEXDIG) / 7*7(":" 1*2HEXDIG) )

route_address_user = < an address in a form recognized by a parser
                       installed with `wolfsentry_addr_family_handler_install
                          ()`
                     >

address_family = uint16 / address_family_name

address_family_name = DQUOTE ( "inet" / "inet6" / "link" / < a value
   recognized by wolfsentry_addr_family_pton() > ) DQUOTE
```

```
route_address_prefix_bits_clause = DQUOTE %s"prefix-bits" DQUOTE ":" uint16

route_protocol = uint16 / route_protocol_name

route_protocol_name = DQUOTE < a value recognized by getprotobyname_r(),
   requiring address family inet or inet6 >

route_port_clause = DQUOTE %s"port" DQUOTE ":" endpoint_port

endpoint_port = uint16 / endpoint_port_name

endpoint_port_name = DQUOTE < a value recognized by getservbyname_r() for the
   previously designated protocol > DQUOTE

route_flag = DQUOTE (
  %s"af-wild" /
  %s"raddr-wild" /
  %s"rport-wild" /
  %s"laddr-wild" /
  %s"lport-wild" /
  %s"riface-wild" /
  %s"liface-wild" /
  %s"tcplike-port-numbers" /
  %s"direction-in" /
  %s"direction-out" /
  %s"penalty-boxed" /
  %s"green-listed" /
  %s"dont-count-hits" /
  %s"dont-count-current-connections" /
  %s"port-reset"
) DQUOTE

action_res_flag = DQUOTE (
  %s"none" /
  %s"accept" /
  %s"reject" /
  %s"connect" /
  %s"disconnect" /
  %s"derogatory" /
  %s"commendable" /
  %s"stop" /
  %s"deallocated" /
  %s"inserted" /
  %s"error" /
  %s"fallthrough" /
  %s"update" /
  %s"port-reset" /
  %s"sending" /
  %s"received" /
  %s"binding" /
  %s"listening" /
  %s"stopped-listening" /
  %s"connecting-out" /
  %s"closed" /
```

```
  %s"unreachable" /
  %s"sock-error" /
  %s"user+0" /
  %s"user+1" /
  %s"user+2" /
  %s"user+3" /
  %s"user+4" /
  %s"user+5" /
  %s"user+6" /
  %s"user+7"
) DQUOTE

user_item = label ":" ( null / true / false / number_sint64_decimal /
    number_float / string / strongly_typed_user_item )

strongly_typed_user_item =
  ( "{" DQUOTE %s"uint" DQUOTE ":" number_uint64 "}" ) /
  ( "{" DQUOTE %s"sint" DQUOTE ":" number_sint64 "}" ) /
  ( "{" DQUOTE %s"float" DQUOTE ":" number_float "}" ) /
  ( "{" DQUOTE %s"string" DQUOTE ":" string_value "}" ) /
  ( "{" DQUOTE %s"base64" DQUOTE ":" base64_value "}" ) /
  json_value_clause

json_value_clause = "{" DQUOTE %s"json" DQUOTE ":" json_value "}"

null = %s"null"

true = %s"true"

false = %s"false"

boolean = true / false

number_uint64 = < decimal number in the range 0...18446744073709551615 > /
                ( DQUOTE < hexadecimal number in the range 0x0...0
                   xffffffffffffffff > DQUOTE ) /
                ( DQUOTE < octal number in the range
                   00...01777777777777777777777 > DQUOTE )

number_sint64_decimal = < decimal number in the range
    -9223372036854775808...9223372036854775807 >

number_sint64 = number_sint64_decimal /
                ( DQUOTE < hexadecimal number in the range -0x8000000000000000
                   ...0x7fffffffffffffff > DQUOTE ) /
                ( DQUOTE < octal number in the range
                   -0100000000000000000000...0777777777777777777777 > DQUOTE
                   )

number_float = < floating point value in a form and range recognized by the
    linked strtod() implementation >

string_value = DQUOTE < any RFC 8259 JSON-valid string that decodes to at most
     16384 octets > DQUOTE
```

```
base64_value = DQUOTE < any valid RFC 4648 base64 encoding that decodes to at
   most 16384 octets > DQUOTE

json_value = < any valid, complete and balanced RFC 8259 JSON expression, with
               keys limited to WOLFSENTRY_MAX_LABEL_BYTES (default 32 bytes),
               overall input length limited to WOLFSENTRY_JSON_VALUE_MAX_BYTES
               if set (default unset), and overall depth limited to
               WOLFSENTRY_MAX_JSON_NESTING (default 16) including the 4 parent
               levels
             >

label = DQUOTE < any RFC 8259 JSON-valid string that decodes to at at least 1
   and at most 32 octets > DQUOTE

uint32 = < decimal integral number in the range 0...4294967295 >

uint16 = < decimal integral number in the range 0...65535 >

uint8 = < decimal integral number in the range 0...255 >
```

4 wolfSentry Release History and Change Log

5 wolfSentry Release 1.6.3 (January 22, 2025)

Release 1.6.3 of the wolfSentry embedded firewall/IDPS has enhancements, additions, and improvements including:

5.1 New Features

Implemented default policy in decisions on lock failures, to better support hard deadline use cases. The lwIP integrated firewall has been updated to leverage this change. Client code calling the dispatch interfaces directly can now check `action_results` for disposition even on error returns.

5.2 Noteworthy Changes and Additions

Add `wolfsentry_set_deadline_rel()`, `wolfsentry_get_deadline_rel()`, and `wolfsentry_get_deadline_rel_usecs()`, to facilitate deployment to deadline-scheduled runtimes. `wolfsentry_get_deadline_rel*()` can be used within implementations of computationally expensive plugins to prevent overrun or limit it to an application-defined tolerance.

Added `WOLFSENTRY_SUCCESS_ID_NO_DEADLINE`, `WOLFSENTRY_SUCCESS_ID_EXPIRED`, and `WOLFSENTRY_SUCCESS_ID_NO_WAITING`, returned by `wolfsentry_get_deadline_rel*()`.

Added `wolfsentry_lock_shared2mutex_is_reserved()`.

5.3 Bug Fixes, Cleanups, and Debugging Aids

Added `WOLFSENTRY_STACKBUF()` to refactor on-stack flexible-element struct instances for additional portability, clarity, and efficiency.

Numerous minor fixes for analyzer hygiene on LLVM 19 and 20, gcc-15, and cppcheck 2.16.

5.4 Self-Test Enhancements

Fixes for several leaks and missing error handling in unit tests.

Added new C23 and `-D_FORTIFY_SOURCE=3` tests.

6 wolfSentry Release 1.6.2 (January 2, 2024)

Release 1.6.2 of the wolfSentry embedded firewall/IDPS has enhancements, additions, and improvements including:

6.1 Noteworthy Changes and Additions

In scripts and Makefile, interpreters (`bash` and `awk`) now follow search `PATH`. Explicit override paths to `bash` and `awk` can be supplied by passing values for `SHELL` and `AWK` to `make`.

Change type of length argument to `wolfsentry_action_res_assoc_by_name()` to `int`, to allow it to accept `WOLFSENTRY_LENGTH_NULL_TERMINATED` (negative number).

Makefile option `STRIPPED` has been split into `STRIPPED` and `FUNCTION_SECTIONS`, the latter directing the compiler and linker to cull any unused object code (with function granularity) to minimize total size.

6.2 Bug Fixes, Cleanups, and Debugging Aids

In `handle_route_endpoint_clause()`, add casts to work around an implicit-promotion bug in gcc-7.5.

In `wolfsentry_route_table_max_purgeable_idle_time_get()` and `_set()`, don't use atomic operations, as the context is already locked and the operand is an `int64_t`. This avoids an inadvertent dependency on software __atomic_load_8() and __atomic_store_8() on 32 bit targets.

Various fixes for benign `cppcheck` reports (`duplicateCondition`, `unsignedLessThanZero`, `unreadVariable`, `invalidPrintfArgType_uint`, `invalidPrintfArgType_sint`, `shadowFunction`, `constVariablePointer`, `preprocessorErrorDirective`).

6.3 Self-Test Enhancements

Add `replace_rule_transactionally()`, now used in `test_static_routes()` for a thorough workout.

Enhance `freertos-arm32-build-test` target to do two builds, one with and one without `FUNCTION_SECTIONS`, for more thorough coverage.

In `test_lwip()` (`tests/unittests.c`), pass a trivial JSON config to `activate_wolfsentry_lwip()`, to avoid compiler optimizing away `wolfsentry_config_json_oneshot()` and its dependencies.

Split cppcheck-analyze recipe into cppcheck-library, cppcheck-force-library, cppcheck-extras, and cppcheck-force-extras, with increased coverage. Only cppcheck-library and cppcheck-extras are included in the "check-all" dependency list.

7 wolfSentry Release 1.6.1 (November 18, 2023)

Release 1.6.1 of the wolfSentry embedded firewall/IDPS has enhancements, additions, and improvements including:

7.1 New Features

Dynamic rules with nonzero connection counts are now subject to deferred expiration, to assure traffic over established connections is allowed until all connections are closed, even with pauses in traffic flow exceeding the max idle time configured for the rule.

When a rule with a nonzero connection count is deleted, actual deletion is deferred until all connections are closed or the `"max-purgeable-idle-time"` is reached (see below). New success code `WOLFSENTRY_SUCCESS_ID_DEFERRED` is returned in that case. If an identical rule is inserted before the deferred deletion, the existing rule is unmarked for deletion and the insertion call returns another new success code, `WOLFSENTRY_SUCCESS_ID_ALREADY_OK`.

A `"max-purgeable-idle-time"` JSON configuration option has been added, forcing expiration and purge of a zombie dynamic rule even if its current connection count is nonzero. New related APIs are also added: `wolfsentry_route_table_max_purgeable_idle_time_get()`, `wolfsentry_route_table_max_purgeable_idle_time_set()`, and `wolfsentry_route_purge_time_set()`.

7.2 Noteworthy Changes and Additions

A new `FILT_CLOSE_WAIT` event type is added to the lwIP integration patch, and a corresponding `WOLFSENTRY_ACTION_RES_CLOSE_WAIT` result bit is added. Appropriate callbacks are added to lwIP `tcp_process()` and `tcp_receive()`, and the lwIP glue logic now handles mapping from `FILT_CLOSE_WAIT` to `WOLFSENTRY_ACTION_RES_CLOSE_WAIT`.

The lwIP patch has been rebased on upstream 5e3268cf3e (Oct 14 2023), while maintaining compatibility with lwIP 2.1.3-RELEASE.

7.3 Bug Fixes, Cleanups, and Debugging Aids

The lwIP patch includes several fixes: * In `tcp_process()`, when handling passive close and entering `CLOSE_WAIT`, don't `tcp_filter_dispatch_incoming(FILT_CLOSED, ...)` – this happens later, at deallocation. * Fix `TCP FILT_CLOSED` callbacks to assure accurate interface ID and local_port are passed.

The route/rule system includes several fixes: * Add error checking to `meta.connection_count` decrement in `wolfsentry_route_event_dispatch_0()`, so that rule churn can never result in count underflow. * Mask out internal flags (via new macro `WOLFSENTRY_ROUTE_INTERNAL_FLAGS`) from `route_exports->flags` in `wolfsentry_route_init_by_exports()`. * In `wolfsentry_route_init_by_exports()`, fix pointer math in `memset()` argument to correctly treat `route_exports->private_data_size` as a byte count. * In `wolfsentry_route_new_by_exports()`, fix check on `route_exports->private_data_size` to properly reflect `config->route_private_data_padding`. * Add missing implementation of `wolfsentry_route_insert_by_exports()`. * In `wolfsentry_route_clone()`, fix allocation to use `WOLFSENTRY_MEMALIGN_1()` when `.route_private_data_alignment` is nonzero. * In `wolfsentry_route_event_dispatch_0()`, don't increment/decrement counts when `WOLFSENTRY_ACTION_RES_FALLTHROUGH`.

In `src/lwip/packet_filter_glue.c`, add `action_results` and `local.sa.interface` to `WOLFSENTRY_DEBUG_LWIP` messages, and add missing gates for `LWIP_IPV6` in `WOLFSENTRY_DEBUG_LWIP` paths.

In `tcp_filter_with_wolfsentry()`, don't set `WOLFSENTRY_ROUTE_FLAG_DIRECTION_IN` for `FILT_REMOTE_RESET`, and fix typo "&event" in call to `wolfsentry_route_event_dispatch_with_inited_result`.

Remove several incorrect calls to `wolfsentry_table_ent_delete_by_id_1()` immediately following failed calls to `wolfsentry_table_ent_insert()` – the former is implicit to the latter.

7.4 Self-Test Enhancements

Add to `test_json()` a workout of `connection_count` and deferred deletion dynamics.

`Makefile.analyzers`: add `sanitize-all-NO_POSIX_MEMALIGN-gcc`; tweak `notification-demo-build-test` to explicitly use the `master` branch of wolfssl.

`Makefile`,`Makefile.analyzers`: tweaks for MacOS X compatibility.

8 wolfSentry Release 1.6.0 (October 24, 2023)

Release 1.6.0 of the wolfSentry embedded firewall/IDPS has enhancements, additions, and improvements including:

8.1 New Features

This release adds native support for the CAN bus address family, and for bitmask-based address matching. CAN addresses and bitmasks are now handled in configuration JSON, as numbers in decimal, octal, or hexadecimal, supporting both 11 bit (part A) and 29 bit (part B) identifiers.

8.2 Noteworthy Changes and Additions

`wolfsentry/wolfsentry.h`:

- Add `WOLFSENTRY_ROUTE_FLAG_REMOTE_ADDR_BITMASK` and `WOLFSENTRY_ROUTE_FLAG_LOCAL_ADDR_BITMAS` to `wolfsentry_route_flags_t`.
- Add `WOLFSENTRY_ACTION_RES_USER0`-`WOLFSENTRY_ACTION_RES_USER6` to `wolfsentry_action_res_t` enum, add `WOLFSENTRY_ACTION_RES_USER7` macro, and refactor `WOLFSENTRY_ACTION_RES_USER_BASE` as a macro aliased to `WOLFSENTRY_ACTION_RES_USER0`.
- Remove `!WOLFSENTRY_NO_STDIO` gate around `wolfsentry_kv_render_value()`.

`wolfsentry/wolfsentry_settings.h`:

- Rename `WOLFSENTRY_NO_STDIO` to `WOLFSENTRY_NO_STDIO_STREAMS`.
- Rename `WOLFSENTRY_HAVE_NONGNU_ATOMICS` to `WOLFSENTRY_NO_GNU_ATOMICS`.
- Added handling for `WOLFSENTRY_NO_SEM_BUILTIN`, `WOLFSENTRY_NO_ADDR_BITMASK_MATCHING`, and `WOLFSENTRY_NO_IPV6`.
- Gate inclusion of `stdio.h` on `!WOLFSENTRY_NO_STDIO_H`, formerly `!WOLFSENTRY_NO_STDIO`.
- Added `WOLFSENTRY_CONFIG_FLAG_ADDR_BITMASKS`, and rename `WOLFSENTRY_CONFIG_FLAG_NO_STDIO` to `WOLFSENTRY_CONFIG_FLAG_NO_STDIO_STREAMS`.

`src/addr_families.c` and `wolfsentry/wolfsentry_af.h`: Split `WOLFSENTRY_AF_LINK` into `WOLFSENTRY_AF_LINK48` and `WOLFSENTRY_AF_LINK64`, with `WOLFSENTRY_AF_LINK` aliased to `WOLFSENTRY_AF_LINK48`.

`src/kv.c`: remove `!WOLFSENTRY_NO_STDIO` gate around `wolfsentry_kv_render_value()`.

`src/json/load_config.c`: In `convert_sockaddr_address()`, add separate handling for `WOLFSENTRY_AF_LINK48` and `WOLFSENTRY_AF_LINK64`.

`Makefile`:

- Refactor `NO_STDIO`, `NO_JSON`, `NO_JSON_DOM`, `SINGLETHREADED`, `STATIC`, and `STRIPPED` to pivot on definedness, not oneness.
- Add feature flags `NO_ADDR_BITMASK_MATCHING` and `NO_IPV6`.
- Rename feature flag `NO_STDIO` to `NO_STDIO_STREAMS`.

8.3 Performance Improvements

`src/routes.c`: Added AF-mismatch optimization to `wolfsentry_route_lookup_0()`.

8.4 Documentation

Add inline documentation for `WOLFSENTRY_NO_GETPROTOBY`, `WOLFSENTRY_SEMAPHORE_INCLUDE`, `WOLFSENTRY_THREAD_INCLUDE`, `WOLFSENTRY_THREAD_ID_T`, and `WOLFSENTRY_THREAD_GET_ID_HANDLER`.

doc/json_configuration.md: add documentation and ABNF grammar for "bitmask" node in route endpoints.

8.5 Bug Fixes and Cleanups

Fixes for user settings file handling:

- Don't #include <wolfsentry/wolfsentry_options.h> if defined(WOLFSENTRY_USER_SETTINGS_FILE).
- Generate and install wolfsentry/wolfsentry_options.h only if USER_SETTINGS_FILE is undefined, and if USER_SETTINGS_FILE is defined, depend on it where previously the dependency was unconditionally on wolfsentry/wolfsentry_options.h.
- If USER_SETTINGS_FILE is set search it to derive JSON build settings.

Makefile: Don't add -pthread to LDFLAGS if RUNTIME is FreeRTOS-lwIP.

wolfsentry/wolfsentry_settings.h:

- Eliminate inclusion of errno.h – now included only in source files that need it.
- Fix handling for WOLFSENTRY_SEMAPHORE_INCLUDE to give it effect in all code paths (previously ignored in POSIX and FreeRTOS paths).

src/routes.c:

- in wolfsentry_route_event_dispatch_0(), move update of meta.purge_after inside the mutex.
- in wolfsentry_route_get_metadata(), conditionalize use of 64 bit WOLFSENTRY_ATOMIC_LOAD() on pointer size, to avoid dependency on library implementation of __atomic_load_8().

src/wolfsentry_internal.c: fix use-after-free bug in wolfsentry_table_free_ents(), using new table->coupled_ent_fn mechanism.

src/json/load_config.c: In convert_sockaddr_address(), handle sa->addr_len consistently – don't overwrite nonzero values.

src/json/{centijson_dom.c,centijson_sax.c,centijson_value.c}: eliminate direct calls to heap allocator functions in WOLFSENTRY code paths, i.e. use only wolfsentry_allocator.

src/json/centijson_value.c: fix uninited-variable defect on cmp in json_value_dict_get_or_add_().

8.6 Self-Test Enhancements

Makefile.analyzers new and enhanced test targets:

- user-settings-build-test: construct a user settings file, then build and self-test using it.
- library-dependency-singlethreaded-build-test and library-dependency-multithreaded-build-test: comprehensive check for unexpected unresolved symbols in the library.
- no-addr-bitmask-matching-test, no-ipv6-test, linux-lwip-test-no-ipv6: tests for new feature gates.
- freertos-arm32-build-test: newly refactored to perform a final link of test_lwip kernel using lwIP and FreeRTOS kernel files and newlib-nano, followed by a check on the size of the kernel.

Added wolfsentry/wolfssl_test.h, containing self-test and example logic relocated from wolfssl/wolfssl/test.h verbatim.

tests/test-config*.json: added several bitmask-matched routes, added several diagnostic events ("set-user-0" through "set-user-4"), and added no-bitmasks and no-ipv6 variants. Also removed AF-wildcard route from tests/test-config-numeric.json to increase test coverage.

tests/unittests.c:

- Additional tweaks for portability to 32 bit FreeRTOS
- Add FreeRTOS-specific implementations of `test_lwip()` and `main()`.
- In `test_json()`, add `wolfsentry_addr_family_handler_install(...,"my_AF2",...)`.
- In `test_json()`, add bitmask tests.
- Added stub implementations for various FreeRTOS/newlib dependencies to support final link in `freertos-arm32-build-test` target.

9 wolfSentry Release 1.5.0 (September 13, 2023)

Release 1.5.0 of the wolfSentry embedded firewall/IDPS has enhancements, additions, and improvements including:

9.1 Noteworthy Changes and Additions

In JSON configuration, recognize `"events"` as equivalent to legacy `"events-insert"`, and `"routes"` as equivalent to legacy `"static-routes-insert"`. Legacy keys will continue to be recognized.

In the `Makefile`, `FREERTOS_TOP` and `LWIP_TOP` now refer to actual distribution top – previously, `FREERTOS_TOP` expected a path to the `FreeRTOS/Source` subdirectory, and `LWIP_TOP` expected a path to the `src` subdirectory.

Added public functions `wolfsentry_route_default_policy_set()` and `wolfsentry_route_default_policy_` implicitly accessing the main route table.

Added public functions `wolfsentry_get_object_type()` and `wolfsentry_object_release()`, companions to existing `wolfsentry_object_checkout()` and `wolfsentry_get_object_id()`.

Added `wolfsentry_lock_size()` to facilitate caller-allocated `wolfsentry_rwlocks`.

`WOLFSENTRY_CONTEXT_ARGS_OUT` is now the first argument to utility routines `wolfsentry_object_checkout()`, `wolfsentry_defaultconfig_get()`, and `wolfsentry_defaultconfig_update()`, rather than a bare `wolfsentry` context pointer.

`ports/Linux-lwIP/include/lwipopts.h`: Add core locking code.

Removed unneeded routine `wolfsentry_config_json_set_default_config()`.

Improved `wolfsentry_kv_render_value()` to use `json_dump_string()` for `_KV_STRING` rendering, if available, to get JSON-style escapes in output.

Implemented support for user-supplied semaphore callbacks.

9.2 Performance Improvements

The critical paths for traffic evaluation have been streamlined by eliminating ephemeral heap allocations, eliminating redundant internal initializations, adding early shortcircuit paths to avoid frivolous processing, and eliminating redundant time lookups and context locking. This results in a 33%-49% reduction in cycles per `wolfsentry_route_event_dispatch()` on `benchmark-test`, and a 29%-61% reduction on `benchmark-singlethreaded-test`, at under 100 cycles for a simple default-policy scenario on a 64 bit target.

9.3 Documentation

Added `doc/freertos-lwip-app.md`, “Building and Initializing wolfSentry for an application on FreeRTOS/lwIP”.

Added `doc/json_configuration.md`, “Configuring wolfSentry using a JSON document”.

Doxygen-based annotations are now included in all wolfSentry header files, covering all functions, macros, types, enums, and structures.

The PDF version of the reference manual is now included in the repository and releases at `doc/wolfSentry_refman.pdf`.

The `Makefile` now has targets `doc-html`, `doc-pdf`, and related targets for generating and cleaning the documentation artifacts.

9.4 Bug Fixes and Cleanups

`lwip/LWIP_PACKET_FILTER_API.patch` has fixes for `-Wconversion` and `-Wshadow` warnings.

`src/json/centijson_sax.c`: Fix bug in `json_dump_double()` such that floating point numbers were rendered with an extra decimal place.

In `wolfsentry_config_json_init_ex()`, error if `json_config.max_key_len` is greater than `WOLFSENTRY_MAX_LABEL_BYTES` (required for memory safety).

In `wolfsentry_config_json_init_ex()`, call `wolfsentry_defaultconfig_get()` to initialize `jps->default_config` with settings previously passed to `wolfsentry_init()`.

`src/kv.c`: Fixed `_KV_STRING` and `_KV_BYTES` cases in `wolfsentry_kv_value_eq_1()` (inadvertently inverted `memcmp()`), and fixed `_KV_NONE` case to return true.

Fixed `wolfsentry_kv_render_value()` for `_KV_JSON` case to pass `JSON_DOM_DUMP_PREFERDICTORDER` to `json_dom_dump()`.

`src/lwip/packet_filter_glue.c`: In `wolfsentry_install_lwip_filter_callbacks()`, if error encountered, disable all callbacks to assure known state on return.

In `wolfsentry_init_ex()`, correctly convert user-supplied `route_idle_time_for_purge` from seconds to `wolfsentry_time_t`.

Pass `route_table->default_event` to `wolfsentry_route_event_dispatch_0()` if caller-supplied trigger event is null (changed in `wolfsentry_route_event_dispatch_1()`, `wolfsentry_route_event_dispatch_by_id_1()`, and `wolfsentry_route_event_dispatch_by_route_1()`).

In `wolfsentry_route_lookup_0()`, fixed scoping of `WOLFSENTRY_ACTION_RES_EXCLUDE_REJECT_ROUTES` to only check `WOLFSENTRY_ROUTE_FLAG_PENALTYBOXED`, not `WOLFSENTRY_ROUTE_FLAG_PORT_RESET`.

In `wolfsentry_route_delete_0()`, properly set `WOLFSENTRY_ROUTE_FLAG_PENDING_DELETE`.

In `wolfsentry_route_event_dispatch_0()` and `wolfsentry_route_event_dispatch_1()`, properly set `WOLFSENTRY_ACTION_RES_ERROR` at end if `ret < 0`.

In `wolfsentry_route_event_dispatch_1()`, properly set `WOLFSENTRY_ACTION_RES_FALLTHROUGH` when `route_table->default_policy` is used.

Added missing `action_results` reset to `wolfsentry_route_delete_for_filter()`.

In `wolfsentry_lock_init()`, properly forbid all inapplicable flags.

Fixed `wolfsentry_eventconfig_update_1()` to copy over all relevant elements.

Fixed and updated expression for `WOLFSENTRY_USER_DEFINED_TYPES`.

9.5 Self-Test Enhancements

`Makefile.analyzers`: Added targets `test_lwip`, `minimal-threaded-build-test`, `pahole-test`, `route-holes-test`, `benchmark-test`, `benchmark-singlethreaded-test`, and `doc-check`.

Implemented tripwires in `benchmark-test` and `benchmark-singlethreaded-test` for unexpectedly high cycles/call.

Enlarged coverage of target `notification-demo-build-test` to run the applications and check for expected and unexpected output.

`tests/unittests.c`:

- Add `test_lwip()` with associated helper functions;
- Add `WOLFSENTRY_UNITTEST_BENCHMARKS` sections in `test_static_routes()` and `test_json()`;

- Add to test_init() tests of wolfsentry_errcode_source_string() and wolfsentry_errcode_error_string();
- Add to test_static_routes() tests of wolfsentry_route_default_policy_set() and wolfsentry_get_object_type(), wolfsentry_object_checkout(), and wolfsentry_object_release().

10 wolfSentry Release 1.4.1 (July 20, 2023)

Release 1.4.1 of the wolfSentry embedded firewall/IDPS has bug fixes including:

10.1 Bug Fixes and Cleanups

Add inline implementations of `WOLFSENTRY_ERROR_DECODE_{ERROR_CODE,SOURCE_ID,LINE_NUMBER}()` for portable protection from multiple argument evaluation, and refactor `WOLFSENTRY_ERROR_ENCODE()` and `WOLFSENTRY_SUCCESS_ENCODE()` to avoid unnecessary dependence on non-portable (gnu-specific) construct.

Use a local stack variable in `WOLFSENTRY_ERROR_ENCODE_1()` to assure a single evaluation of the argument.

Add `-Wno-inline` to `CALL_TRACE` `CFLAGS`.

Correct the release date of 1.4.0 in ChangeLog.

10.2 Self-Test Enhancements

Add `CALL_TRACE-test` to `Makefile.analyzers`, and include it in the `check-extra` dep list.

11 wolfSentry Release 1.4.0 (July 19, 2023)

Release 1.4.0 of the wolfSentry embedded firewall/IDPS has bug fixes and improvements including:

11.1 New Features

Routes can now be configured to match traffic with designated `action_results` bit constraints, and can be configured to update `action_results` bits, by inserting the route with a parent event that has the desired configuration. Parent events can now also be configured to add or clear route flags for all routes inserted with that parent event.

Added new aux_event mechanism to facilitate distinct configurations for a static generator route and the narrower ephemeral routes dynamically created when it is matched.

Added a new built-in action, `"%track-peer-v1"`, that can be used in combination with the above new facilities to dynamically spawn ephemeral routes, allowing for automatic pinhole routes, automatic adversary tracking, and easy implementation of dynamic blocks and/or notifications for port scanning adversaries.

11.2 Noteworthy Changes and Additions

Added new APIs `wolfsentry_event_set_aux_event()` and `wolfsentry_event_get_aux_event()`.

Added flag filters and controls to `struct wolfsentry_eventconfig`, and added corresponding clauses to JSON `"config"` sections:

- `.action_res_filter_bits_set`, “action-res-filter-bits-set”
- `.action_res_filter_bits_unset`, `"action-res-filter-bits-unset"`
- `.action_res_bits_to_add`, `"action-res-bits-to-add"`
- `.action_res_bits_to_clear`, `"action-res-bits-to-clear"`
- `.route_flags_to_add_on_insert`, `"route-flags-to-add-on-insert"`
- `.route_flags_to_clear_on_insert`, `"route-flags-to-clear-on-insert"`

Added new `WOLFSENTRY_ACTION_RES_*` (action result) flags to support filtering matches by generic traffic type:

- `WOLFSENTRY_ACTION_RES_SENDING`
- `WOLFSENTRY_ACTION_RES_RECEIVED`
- `WOLFSENTRY_ACTION_RES_BINDING`
- `WOLFSENTRY_ACTION_RES_LISTENING`
- `WOLFSENTRY_ACTION_RES_STOPPED_LISTENING`
- `WOLFSENTRY_ACTION_RES_CONNECTING_OUT`
- `WOLFSENTRY_ACTION_RES_CLOSED`
- `WOLFSENTRY_ACTION_RES_UNREACHABLE`
- `WOLFSENTRY_ACTION_RES_SOCK_ERROR`

These flags are now passed by the lwIP integration code in `src/lwip/packet_filter_glue.c`. Detailed descriptions of these and other `_ACTION_RES_` bits are in `wolfsentry/wolfsentry.h`.

Added `wolfsentry_addr_family_max_addr_bits()`, to allow programmatic determination of whether a given address is a prefix or fully specified.

Added a family of functions to let routes be inserted directly from a prepared `struct wolfsentry_route_exports`, and related helper functions to prepare it:

- `wolfsentry_route_insert_by_exports_into_table()`
- `wolfsentry_route_insert_by_exports()`
- `wolfsentry_route_insert_by_exports_into_table_and_check_out()`

- `wolfsentry_route_insert_by_exports_and_check_out()`
- `wolfsentry_route_reset_metadata_exports()`

Added convenience accessor/validator functions for routes:

- `wolfsentry_route_get_addrs()`
- `wolfsentry_route_check_flags_sensical()`

Refactored the event action list implementation so that the various action lists (`WOLFSENTRY_ACTION_TYPE_POST`, `_INSERT`, `_MATCH`, `_UPDATE`, `_DELETE`, and `_DECISION`) are represented directly in the `struct wolfsentry_event`, rather than through a "subevent". The related APIs (`wolfsentry_event_action_prepend()`, `wolfsentry_event_action_append()`, `wolfsentry_event_action_insert_after()`, `wolfsentry_event_action_delete()`, `wolfsentry_event_action_list_start()`) each gain an additional argument, `which_action_list`. The old JSON grammar is still supported via internal emulation (still tested by `test-config.json`). The JSON configuration for the new facility is `"post-actions"`, `"insert-actions"`, `"match-actions"`, `"update-actions"`, `"delete-actions"`, and `"decision-actions"`, each optional, and each expecting an array of zero or more actions.

Added a restriction that user-defined action and event labels can't start with "%", and correspondingly, all built-in actions and events have labels that start with "%". This can be overridden by predefining `WOLFSENTRY_BUILTIN_LABEL_PREFIX` in user settings.

Removed unused flag `WOLFSENTRY_ACTION_RES_CONTINUE`, as it was semantically redundant relative to `WOLFSENTRY_ACTION_RES_STOP`.

Removed flags `WOLFSENTRY_ACTION_RES_INSERT` and `WOLFSENTRY_ACTION_RES_DELETE`, as the former is superseded by the new builtin action facility, and the latter will be implemented later with another builtin action.

Added flag `WOLFSENTRY_ACTION_RES_INSERTED`, to indicate when a side-effect route insertion was performed. This flag is now always set by the route insert routines when they succeed. Action plugins must copy this flag as shown in the new `wolfsentry_builtin_action_track_peer()` to assure proper internal accounting.

Reduced number of available user-defined `_ACTION_RESULT_` bits from 16 to 8, to accommodate new generic traffic bits (see above).

In `struct wolfsentry_route_metadata_exports`, changed `.connection_count`, `.derogatory_count`, and `.commendable_count`, from `wolfsentry_hitcount_t` to `uint16_t`, to match internal representations. Similarly, in `struct wolfsentry_route_exports`, changed `.parent_event_label_len` from `size_t` to `int` to match `label_len` arg type.

Added `wolfsentry_table_ent_get_by_id()` to the public API.

Renamed public API `wolfsentry_action_res_decode()` as `wolfsentry_action_res_assoc_by_flag()` for clarity and consistency.

11.3 Bug Fixes and Cleanups

Consistently set the `WOLFSENTRY_ACTION_RES_FALLTHROUGH` flag in `action_results` when dispatch classification (`_ACCEPT`/`_REJECT`) was by fallthrough policy.

Refactored internal code to avoid function pointer casts, previously used to allow implementations with struct pointers where a handler pointer has a type that expects `void *`. The refactored code has shim implementations with fully conformant signatures, that cast the arguments to pass them to the actual implementations. This works around over-eager analysis by the `clang` UB sanitizer.

Fix missing default cases in non-enum `switch()` constructs.

11.4 Self-Test Enhancements

Added new clauses to `test-config*.json` for `wolfsentry_builtin_action_track_peer()` (events "ephemeral-pinhole-parent", "pinhole-generator-parent", "ephemeral-port-scanner-parent", "port-scanner-generator-parent", and related routes), and added full dynamic workout for them to `test_json()`.

Add unit test coverage:

- `wolfsentry_event_set_aux_event()`
- `wolfsentry_event_get_aux_event()`
- `wolfsentry_event_get_label()`
- `wolfsentry_addr_family_max_addr_bits()`

12 wolfSentry Release 1.3.1 (July 5, 2023)

Release 1.3.1 of the wolfSentry embedded firewall/IDPS has bug fixes and improvements including:

12.1 Bug Fixes and Cleanups

Updated lwIP patches to fix `packet_filter_event_t` checking on short-enum targets.

Fixed copying of route table header fields (table config) when cloning or rebuilding (preserve default policy etc when loading with `WOLFSENTRY_CONFIG_LOAD_FLAG_LOAD_THEN_COMMIT | WOLFSENTRY_CONFIG_LOAD_FLAG_NO_FLUSH` or `WOLFSENTRY_CONFIG_LOAD_FLAG_FLUSH_ONLY_ROUTES`).

Implemented proper locking in `wolfsentry_route_get_reference()`, and corresponding lock assertion in `wolfsentry_table_cursor_init()`.

Fixed logic in address matching to properly match zero-length addresses when peforming subnet matching, even if the corresponding `_ADDR_WILDCARD` flag bit is clear.

12.2 Self-Test Enhancements

`Makefile.analyzers`: add `-fshort-enums` variants to `sanitize-all` and `sanitize-all-gcc` recipes, and add `short-enums-test` recipe.

Added `wolfsentry_route_event_dispatch()` cases to `test_json()`.

Added unit test coverage to confirm correct copying of route table header fields when cloning.

13 wolfSentry Release 1.3 (May 19, 2023)

Release 1.3 of the wolfSentry embedded firewall/IDPS has bug fixes and improvements including:

13.1 New Features

13.1.1 Route dump to JSON

The route (rule) table can now be dumped in conformant JSON format to a byte stream, using wolfSentry intrinsics (no `stdio` dependencies), and subsequently reloaded.

- `wolfsentry_route_table_dump_json_start()`, `_next()`, `_end()`
- Byte streams using new `WOLFSENTRY_BYTE_STREAM_*()` macros, with stack and heap options.
- Retryable rendering on `_BUFFER_TOO_SMALL` error, by flushing the byte stream, calling `WOLFSENTRY_BYTE_STREAM_RESET()`, and retrying the `wolfsentry_route_table_dump_json_*()` call.
- New flag `WOLFSENTRY_CONFIG_LOAD_FLAG_FLUSH_ONLY_ROUTES`, to allow reloads that leave all event and key-value configuration intact, and only replace the routes.

13.2 Bug Fixes and Cleanups

- Non-threadsafe `get{proto,serv}by{name.number}()` calls (already configuration-gated) have been replaced by their `_r()` counterparts, and gated on compatible glibc.
- Fixed an underread bug in `convert_hex_byte()` that affected parsing of MAC addresses.

13.3 Self-Test Enhancements

- Added `__wolfsentry_wur` to `WOLFSENTRY_LOCAL`.
- Added new clauses in `test_json()` to verify bitwise idempotency of route table export-ingest cycles to/from JSON.
- Added new target `notification-demo-build-test`.

14 wolfSentry Release 1.2.2 (May 4, 2023)

Release 1.2.2 of the wolfSentry embedded firewall/IDPS has bug fixes and improvements including:

14.1 Noteworthy Changes and Additions

Added C89 pedantic compatibility in core codebase, including unit tests, via `-DWOLFSENTRY_C89`.

Added error code `IO_FAILED`, returned for various stdio failures that previously returned `SYS_OP_FAILED` or went undetected.

Refined `wolfsentry_lock_unlock()` so that final unlock while holding a promotion reservation is not an error and implicitly drops the reservation.

14.2 Bug Fixes and Cleanups

Cleanups guided by `clang-tidy` and `cppcheck`: fixed a misused retval from `posix_memalign()`, fixed overwritten retvals in `wolfsentry_lock_unlock()`, and effected myriad cleanups to improve clarity and portability.

Fixed missing assignment of `new->prev` in `wolfsentry_table_clone()`.

Fixed route metadata coherency in transactional configuration updates: add `wolfsentry_route_copy_metadata()` and call it from `wolfsentry_context_exchange()`.

When `wolfsentry_route_event_dispatch*()` results in a default policy fallback, return `USED_FALLBACK` success code.

Properly release lock promotion reservation in `wolfsentry_config_json_init_ex()` if obtained.

Fixed several accounting bugs in the lock kernel related to promotion reservations.

Copy `fallthrough_route` pointer in `wolfsentry_route_table_clone_header()`, rather than improperly trying to clone the fallthrough route.

14.3 Self-Test Enhancements

Added new global compiler warnings to `Makefile`:

- `-Wmissing-prototypes`
- `-Wdeclaration-after-statement`
- `-Wnested-externs`
- `-Wlogical-not-parentheses`
- `-Wpacked-not-aligned`

Added new targets to `Makefile.analyzers`:

- `clang-tidy-build-test`
- `cppcheck-analyze`
- `c89-test`
- `m32-c89-test`
- `freertos-arm32-c89-build-test`
- `freertos-arm32-singlethreaded-build-test`
- `sanitize-aarch64-be-test`
- `sanitize-all-no-inline-gcc`
- `no-inline-test`
- `no-alloca-test`
- `release-check`

Added `WOLFSENTRY_CONFIG_LOAD_FLAG_NO_FLUSH` coverage and an array of should-fail JSON objects to `unittests.c:test_json()`.

Added more arg-not-null and thread-inited checks to thread/lock routines in `src/wolfsentry_util.c`, and corresponding unit test coverage for all null/uninited arg permutations.

Added assert in release recipe to assure that wolfsentry.h has a version that matches the tagged version.

15 wolfSentry Release 1.2.1 (Apr 5, 2023)

Release 1.2.1 of the wolfSentry embedded firewall/IDPS has bug fixes and improvements including:

15.1 Noteworthy Changes and Additions

Added API `wolfsentry_route_render_flags()`, now used in `wolfsentry_route_render()` and `wolfsentry_route_exports_render()`.

Refactored `wolfsentry_route_lookup_0()` to consistently return the highest-priority matching route, breaking ties using `compare_match_exactness()`.

Added `DEBUG_ROUTE_LOOKUP` code paths in `wolfsentry_route_lookup_0()`, for verbose troubleshooting of configurations and internal logic.

Added to `convert_hex_byte()` (and therefore to MAC address parsing) tolerance for single-hex-digit byte values, as in `a:b:c:1:2:3`.

15.2 Bug Fixes

Removed several inappropriate wildcard flags on queries in lwIP event handlers, particularly `_SA_LOCAL_PORT_WILDCARD` for `FILT_PORT_UNREACHABLE` and `*_INTERFACE_WILDCARD` for `FILT_BINDING`/`FILT_LISTENING`/`FILT_STOP_LISTENING` and when `event->netif` is null.

Added nullness checks for `laddr` and `raddr` in lwIP event handlers, and if null, set all-zeros address.

Refactored wildcard handling in `wolfsentry_route_init()`, `wolfsentry_route_new()`, and `wolfsentry_route_insert_1()`, to zero out wildcard fields at insert time, rather than at init time, so that routes used as targets contain accurate information for `compare_match_exactness()`, regardless of wildcard bits.

Fixed `WOLFSENTRY_VERSION_*` values, which were inadvertently swapped in release 1.2.0.

16 wolfSentry Release 1.2.0 (Mar 24, 2023)

Production Release 1.2.0 of the wolfSentry embedded firewall/IDPS has bug fixes and improvements including:

16.1 New Features

16.1.1 lwIP full firewall integration

When wolfSentry is built with make options `LWIP=1 LWIP_TOP=<path-to-lwIP-source>`, the library is built with new APIs `wolfsentry_install_lwip_filter_ethernet_callback()`, `wolfsentry_install_lwip_filter_ip_callbacks()`, `wolfsentry_install_lwip_filter_icmp_callbacks()`, `wolfsentry_install_lwip_filter_tcp_callback()`, `wolfsentry_install_lwip_filter_udp_callback()` and the all-on-one `wolfsentry_install_lwip_filter_callbacks()`. For each layer/protocol, a simple bitmask, of type `packet_filter_event_mask_t`, allows events to be selectively filtered, with other traffic passed with negligible overhead. For example, TCP connection requests can be fully evaluated by wolfSentry, while traffic within established TCP connections can pass freely.

`wolfSentry LWIP=1` relies on a patchset to lwIP, gated on the macro `LWIP_PACKET_FILTER_API`, that adds generic filter callback APIs to each layer and protocol. See `lwip/README.md` for details.

In addition to `LWIP_DEBUG` instrumentation, the new integration supports `WOLFSENTRY_DEBUG_PACKET_FILTER`, which renders the key attributes and outcome for all callout events.

16.2 Noteworthy Changes and Additions

Routes and default actions can now be annotated to return `WOLFSENTRY_ACTION_RES_PORT_RESET` in their `action_results`. This is used in the new lwIP integration to control whether TCP reset and ICMP port-unreachable packets are sent (versus dropping the rejected packet unacknowledged).

A new `ports/` tree is added, and the former FreeRTOS/ tree is moved to `ports/FreeRTOS-lwIP`.

New helper macros are added for managing thread state: `WOLFSENTRY_THREAD_HEADER_DECLS`, `WOLFSENTRY_THREAD_HEADER_INIT()`, `WOLFSENTRY_THREAD_HEADER_INIT_CHECKED()`.

New flags `WOLFSENTRY_ROUTE_FLAG_PORT_RESET` and `WOLFSENTRY_ACTION_RES_EXCLUDE_REJECT_ROUTES` to support firewall functionalities.

16.3 Bug Fixes

Wildcard matching in the routes/rules table now works correctly even for non-contiguous wildcard matching.

`struct wolfsentry_sockaddr` now aligns its `addr` member to a 4 byte boundary, for safe casting to `(int *)`, using a new `attr_align_to()` macro.

The route lookup algorithm has been improved for correct results with non-contiguous wildcards, to correctly break ties using the new `compare_match_exactness()`, and to correctly give priority to routes with a matching event.

When matching target routes (e.g. with `wolfsentry_route_event_dispatch()`), ignore failure in `wolfsentry_event_get_reference()` if `WOLFSENTRY_ROUTE_FLAG_PARENT_EVENT_WILDCARD` is set in the `flags`.

17 wolfSentry Release 1.1.0 (Feb 23, 2023)

Production Release 1.1.0 of the wolfSentry embedded firewall/IDPS has bug fixes and improvements including:

17.1 New Features

Internal settings, types, alignments, constants, a complete set of internal shims, and Makefile clauses, for portability to native FreeRTOS with threads on 32 bit gcc targets.

17.2 Noteworthy Changes and Additions

rwlock control contexts can now be allocated inside interrupt handlers, and `WOLFSENTRY_LOCK_FLAG_RETAIN_SEMAP` can be supplied to the new `wolfsentry_context_lock_mutex_timed_ex()`, allowing safe trylock followed by automatic lock recursion.

API routines are now marked warn-unused-return by default, subject to user-defined override. This new default warns on untrapped errors, to aid preventing undefined behavior.

API arguments previously accepting “long” ints for counts of seconds now expect `time_t`, for portability to ARM32 and FreeRTOS.

New unit test: `test_json_corpus`, for highly configurable bulk trial runs of the JSON processing subsystem.

New tests in `Makefile.analyzers`: `no-getprotoby-test`, `freertos-arm32-build-test`.

A new guard macro, `WOLFSENTRY_NO_GETPROTOBY`, allows narrow elimination of dependencies on `getprotobyname()` and `getprotobynumber()`.

Recursive JSON DOM tree processing logic was refactored to greatly reduce stack burden.

Substantial enlargement of code coverage by unit tests, guided by `gcov`.

New convenience macros for typical threaded state tracking wrappers: `WOLFSENTRY_THREAD_HEADER_CHECKED()` and `WOLFSENTRY_THREAD_TAILER_CHECKED()`.

17.3 Bug Fixes

Cloning of user-defined deep JSON objects is now implemented, as needed for configuration load dry runs and load-then-commit semantics.

JSON processing of UTF-8 surrogate pairs is now fixed.

Fixed retval testing in `wolfsentry_action_list_{append,prepend,insert}_1()`, and added missing `point_action` lookup in `wolfsentry_action_list_insert_after()`.

Fixed potential use-after-free defect in `wolfsentry_event_delete()`.

18 wolfSentry Release 1.0.0 (Jan 18, 2023)

Production Release 1.0.0 of the wolfSentry embedded firewall/IDPS has bug fixes and improvements including:

18.1 Noteworthy Changes and Additions

- Makefile improvements around `wolfsentry_options.h`, and a new com-bundle rule.
- A new macro `WOLFSENTRY_USE_NONPOSIX_THREADS`, separated from `WOLFSENTRY_USE_NONPOSIX_SEMAPHORE` supporting mixed-model targets, e.g. Mac OS X.

18.2 Bug Fixes

- In `examples/notification-demo/log_server/log_server.c`, in `main()`, properly reset `transaction_successful` at top of the accept loop.

19 wolfSentry Release 0.8.0 (Jan 6, 2023)

Preview Release 0.8.0 of the wolfSentry embedded firewall/IDPS has bug fixes and new features including:

19.1 New Features

19.1.1 Multithreaded application support

- Automatic locking on API entry, using a high performance, highly portable semaphore-based readwrite lock facility, with error checking and opportunistic lock sharing.
- Thread-specific deadlines set by the caller, limiting waits for lock acquisition as needed for real-time applications.
- A mechanism for per-thread private data, accessible to user plugins.
- No dependencies on platform-supplied thread-local storage.

19.2 Updated Examples

19.2.1 examples/notification-demo

- Add interrupt handling for clean error-checked shutdown in `log_server`.
- Add `/kill-server` admin command to `log_server`.
- Reduce penalty-box-duration in `notify-config.{json,h}` to 10s for demo convenience.

19.3 Noteworthy Changes and Additions

- A new first argument to `wolfsentry_init_ex()` and `wolfsentry_init()`, `caller_build_settings`, for runtime error-checking of application/library compatibility. This mechanism will also allow future library changes to be conditionalized on caller version and/or configuration expectations as needed, often avoiding the need for application recompilation.
- `src/util.c` was renamed to `src/wolfsentry_util.c`.
- `wolfsentry/wolfsentry_settings.h` was added, containing setup code previously in `wolfsentry/wolfsentry.h`.
- Error IDs in `enum wolfsentry_error_id` are all now negative, and a new `WOLFSENTRY_SUCCESS_ID_*` namespace was added, with positive values and supporting macros.

19.3.1 New public utility APIs, macros, types, etc.

- `WOLFSENTRY_VERSION_*` macros, for version testing
- `wolfsentry_init_thread_context()`, `wolfsentry_alloc_thread_context()`, `wolfsentry_get_thread_id()`, `wolfsentry_get_thread_user_context()`, `wolfsentry_get_thread_deadline()`, `wolfsentry_get_thread_flags()`, `wolfsentry_destroy_thread_conte` `wolfsentry_free_thread_context()`, `wolfsentry_set_deadline_rel_usecs()`, `wolfsentry_set_deadline_abs()`, `wolfsentry_clear_deadline()`, `wolfsentry_set_thread_readonly()`, `wolfsentry_set_thread_readwrite()`
- `WOLFSENTRY_DEADLINE_NEVER` and `WOLFSENTRY_DEADLINE_NOW`, used internally and for testing values returned by `wolfsentry_get_thread_deadline()`
- Many new values in the `WOLFSENTRY_LOCK_FLAG_*` set.

- `wolfsentry_lock_*()` APIs now firmed, and new `wolfsentry_context_lock_shared_with_reservation`
- `WOLFSENTRY_CONTEXT_*` helper macros.
- `WOLFSENTRY_UNLOCK_*()`, `WOLFSENTRY_SHARED_*()`, `WOLFSENTRY_MUTEX_*()`, and `WOLFSENTRY_PROMOTABLE_*()` helper macros
- `WOLFSENTRY_ERROR_UNLOCK_AND_RETURN()`, `WOLFSENTRY_SUCCESS_UNLOCK_AND_RETURN()`, and related helper macros.

19.4 Bug Fixes

- Various fixes, and additional hardening and cleanup, in the readwrite lock kernel.
- Various fixes in `Makefile`, for proper handling and installation of `wolfsentry_options.h`.

20 wolfSentry Release 0.7.0 (Nov 7, 2022)

Preview Release 0.7.0 of the wolfSentry embedded firewall/IDPS has bug fixes and new features including:

20.1 New Features

20.1.1 Support for freeform user-defined JSON objects in the "user-values" (key-value pair) section of the config package.

- Uses syntax `"key" : { "json" : x }` where x is any valid standalone JSON expression.
- Key length limited to `WOLFSENTRY_MAX_LABEL_BYTES` by default.
- String length limited to `WOLFSENTRY_KV_MAX_VALUE_BYTES` by default.
- JSON tree depth limited to `WOLFSENTRY_MAX_JSON_NESTING` by default.
- All default limits subject to caller runtime override using the `json_config` arg to the new APIs `wolfsentry_config_json_init_ex()` and `wolfsentry_config_json_oneshot_ex()`, accepting a `JSON_CONFIG *` (accepted as `const`).

20.1.1.1 New APIs for JSON KVs

- `wolfsentry_user_value_store_json()`
- `wolfsentry_user_value_get_json()`
- `WOLFSENTRY_KV_V_JSON()`
- `wolfsentry_config_json_init_ex()`
- `wolfsentry_config_json_oneshot_ex()`

20.1.1.2 New config load flags controlling JSON KV parsing

- `WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_DUPKEY_ABORT`
- `WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_DUPKEY_USEFIRST`
- `WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_DUPKEY_USELAST`
- `WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_MAINTAINDICTORDER`

20.1.2 Support for setting a user KV as read-only.

- Read-only KVs can't be deleted or overwritten without first setting them read-write.
- Mechanism can be used to protect user-configured data from dynamic changes by JSON configuration package – JSON cannot change or override the read-only bit.

20.1.2.1 KV mutability APIs:

- `wolfsentry_user_value_set_mutability()`
- `wolfsentry_user_value_get_mutability()`

20.2 Updated Examples

20.2.1 examples/notification-demo

- Update and clean up udp_to_dbus, and add `--kv-string` and `--kv-int` command line args for runtime ad hoc config overrides.

- Rename config node controlling the udp_to_dbus listen address from “notification-dest-addr” to “notification-listen-addr”.

20.2.1.1 Added examples/notification-demo/log_server

- Toy embedded web server demonstrating HTTPS with dynamic insertion of limited-lifespan wolfSentry rules blocking (penalty boxing) abusive peers.
- Demonstrates mutual authentication using TLS, and role-based authorizations pivoting on client certificate issuer (certificate authority).

20.3 Noteworthy Changes and Additions

- JSON strings (natively UTF-8) are now consistently passed in and out with unsigned char pointers.
- wolfsentry_kv_render_value() now has a struct wolfsentry_context * as its first argument (necessitated by addition of freeform JSON rendering).
- Added new API routine wolfsentry_centijson_errcode_translate(), allowing conversion of all CentiJSON return codes (e.g. from json_dom_parse(), json_value_path(), and json_value_build_path()) from native CentiJSON to roughly-corresponding native wolfSentry codes.

20.3.1 Cleanup of JSON DOM implementation

- Added json_ prefix to all JSON functions and types.
- CentiJSON now uses wolfSentry configured allocator for all heap operations.

20.3.2 New utility APIs

- wolfsentry_get_allocator()
- wolfsentry_get_timecbs()

20.4 Bug Fixes

- Fix error-path memory leak in JSON KV handling.
- Fix “echo: write error: Broken pipe” condition in recipe for rule “force”
- Various minor portability fixes.
- Enlarged scope for build-time pedantic warnings – now includes all of CentiJSON.

21 wolfSentry Release 0.6.0 (Sep 30, 2022)

Preview Release 0.6.0 of the wolfSentry embedded firewall/IDPS has bug fixes and new features including:

21.1 New Features

21.1.1 Core support for automatic penalty boxing, with configurable threshold when derogatory count reaches threshold

21.1.1.1 New APIs for manipulating route derogatory/commendable counts from application/-plugin code:

- `wolfsentry_route_increment_derogatory_count()`
- `wolfsentry_route_increment_commendable_count()`
- `wolfsentry_route_reset_derogatory_count()`
- `wolfsentry_route_reset_commendable_count()`

21.1.1.2 New JSON config nodes:

- `derog-thresh-for-penalty-boxing`
- `derog-thresh-ignore-commendable`
- `commendable-clears-derogatory`

21.1.1.3 Automatic purging of expired routes:

- constant time garbage collection
- `wolfsentry_route_table_max_purgeable_routes_get()`
- `wolfsentry_route_table_max_purgeable_routes_set()`
- `wolfsentry_route_stale_purge_one()`

21.2 Noteworthy Changes and Additions

- New API `wolfsentry_route_insert_and_check_out()`, allowing efficient update of route state after insert; also related new API `wolfsentry_object_checkout()`.
- New APIs `wolfsentry_route_event_dispatch_by_route()` and `wolfsentry_route_event_dispatch_by` analogous to the `_by_id()` variants, but accepting a struct wolfsentry_route pointer directly.
- `wolfsentry_route_init()` and `wolfsentry_route_new()` now allow (and ignore) nonzero supplied values in wildcarded wolfsentry_sockaddr members.
- New debugging aid, make CALL_TRACE=1, gives full call stack trace with codepoints and error codes, to aid debugging of library, plugins, and configurations.

21.3 Bug Fixes

- src/internal.c: fix wrong constant of iteration in `wolfsentry_table_ent_get_by_id()`.

22 wolfSentry Release 0.5.0 (Aug 1, 2022)

Preview Release 0.5.0 of the wolfSentry embedded firewall/IDPS has bug fixes and new features including:

22.1 New Example

22.1.0.1 examples/notification-demo Added examples/notification-demo, demonstrating plugin actions, JSON event representation, and pop-up messages using the D-Bus notification facility and a middleware translation daemon.

22.2 Noteworthy Changes

- Added new API `wolfsentry_init_ex()` with `wolfsentry_init_flags_t` argument.
- Added runtime error-checking on lock facility.

22.3 Bug Fixes

Fix missing assignment in `wolfsentry_list_ent_insert_after()`.

23 wolfSentry Release 0.4.0 (May 27, 2022)

Preview Release 0.4.0 of the wolfSentry embedded firewall/IDPS has bug fixes and new features including:

23.1 New Features

- User-defined key-value pairs in JSON configuration: allows user plugins to access custom config parameters in the wolfSentry config using the new `wolfsentry_user_value_*()` family of API functions. Binary configuration data can be supplied in the configuration using base64 encoding, and are decoded at parse time and directly available to user plugins in the original raw binary form. The key-value facility also supports a custom validator callback to enforce constraints on user-defined config params in the JSON.
- User-defined address families: allows user plugins for custom address families and formats, using new `wolfsentry_addr_family_*()` API routines. This allows idiomatic formats for non-Internet addresses in the JSON config, useful for various buses and device namespaces.
- Formalization of the concepts of default events and fallthrough rules in the route tables.
- A new subevent action list facility to support logging and notifications around the final decisions of the rule engine, alongside the existing subevents for rule insertions, matches, and deletions.
- The main plugin interface (`wolfsentry_action_callback_t`) now passes two separate routes, a "`trigger_route`" with full attributes of the instant traffic, and a "`rule_route`" that matches that traffic. In dynamic rule scenarios, plugins can manipulate the passed `rule_route` and set the `WOLFSENTRY_ACTION_RES_INSERT` bit in the to define a new rule that will match the traffic thereafter. All actions in the chain retain readonly access to the unmodified trigger route for informational purposes.
- The JSON DOM facility from CentiJSON is now included in the library by default (disabled by make `NO_JSON_DOM=1`), layered on the SAX facility used directly by the wolfSentry core to process the JSON config package. The DOM facility can be used as a helper in user plugins and applications, for convenient JSON parsing, random access, and production.

23.2 Noteworthy Changes

- In the JSON config, non-event-specific members of top level node "config-update" node have been moved to the new top level node "default-policies", which must appear after "event-insert". "default-policies" members are "default-policy-static", "default-policy-dynamic", "default-event-static", and "default-event-dynamic".

23.3 Bug Fixes

- In `wolfsentry_config_json_init()`, properly copy the load_flags from the caller into the `_json_process_state`.
- The JSON SAX API routines (`wolfsentry/centijson_sax.h`) are now properly exported.

24 wolfSentry Release 0.3.0 (Dec 30, 2021)

Preview Release 0.3.0 of the wolfSentry embedded firewall/IDPS has bug fixes and new features including:

24.1 New Ports and Examples

24.1.0.1 examples/Linux-LWIP This demo uses Linux-hosted LWIP in Docker containers to show packet-level and connection-level filtering using wolfSentry. Filtering can be by MAC, IPv4, or IPv6 address. Demos include pre-accept TCP filtering, and filtering of ICMP packets.

See examples/Linux-LWIP/README.md for the installation and usage guide, and examples/Linux-LWIP/echo-config.json for the associated wolfSentry configuration.

24.1.0.2 FreeRTOS with LWIP on STM32 This demo is similar to Linux-LWIP, but targets the STM32 ARM core and the STM32CubeMX or STM32CubeIDE toolchain, with a FreeRTOS+LWIP runtime. It shows wolfSentry functionality in a fully embedded (bare metal) application.

See examples/STM32/README.md for the installation and usage guide, and examples/STM32/Src/sentry.c for the compiled-in wolfSentry configuration.

24.2 New Features

- Autogeneration and inclusion of `wolfsentry_options.h`, synchronizing applications with wolfSentry library options as built.
- New APIs `wolfsentry_route_event_dispatch_[by_id]with_inited_result()`, for easy caller designation of known traffic attributes, e.g. `WOLFSENTRY_ACTION_RES_CONNECT` or `WOLFSENTRY_ACTION_RES_DISCONNECT`.
- Efficient support for aligned heap allocations on targets that don't have a native aligned allocation API: `wolfsentry_free_aligned_cb_t`, `wolfsentry_allocator.free_aligned`, `wolfsentry_builtin_free_aligned()`, `wolfsentry_free_aligned()`, and `WOLFSENTRY_FREE_ALIGNED()`.
- Semaphore wrappers for FreeRTOS, for use by the `wolfsentry_lock_*()` shareable-upgradeable lock facility.

24.3 Bug Fixes

- `wolfsentry_route_event_dispatch_1()`: don't impose `config.penaltybox_duration` on routes with `route->meta.last_penaltybox_time == 0`.
- trivial fixes for backward compat with gcc-5.4.0, re `-Wconversion` and `-Winline`.

Please send questions or comments to douzzer@wolfssl.com

