Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'OPENSSL_strdup.3ossl' command

$ man OPENSSL_strdup.3ossl

OPENSSL_MALLOC(3ossl) OpenSSL OPENSSL_MALLOC(3ossl)

NAME

OPENSSL_malloc_init, OPENSSL_malloc, OPENSSL_zalloc, OPENSSL_realloc,

OPENSSL_free, OPENSSL_clear_realloc, OPENSSL_clear_free,

OPENSSL_cleanse, CRYPTO_malloc, CRYPTO_zalloc, CRYPTO_realloc,

CRYPTO_free, OPENSSL_strdup, OPENSSL_strndup, OPENSSL_memdup,

OPENSSL_strlcpy, OPENSSL_strlcat, CRYPTO_strdup, CRYPTO_strndup,

OPENSSL_mem_debug_push, OPENSSL_mem_debug_pop, CRYPTO_mem_debug_push,

CRYPTO_mem_debug_pop, CRYPTO_clear_realloc, CRYPTO_clear_free,

CRYPTO_malloc_fn, CRYPTO_realloc_fn, CRYPTO_free_fn,

CRYPTO_get_mem_functions, CRYPTO_set_mem_functions,

CRYPTO_get_alloc_counts, CRYPTO_set_mem_debug, CRYPTO_mem_ctrl,

CRYPTO_mem_leaks, CRYPTO_mem_leaks_fp, CRYPTO_mem_leaks_cb,

OPENSSL_MALLOC_FAILURES, OPENSSL_MALLOC_FD - Memory allocation

functions

SYNOPSIS

```
#include <openssl/crypto.h>

int OPENSSL_malloc_init(void);

void *OPENSSL_malloc(size_t num);

void *OPENSSL_zalloc(size_t num);

void *OPENSSL_realloc(void *addr, size_t num);

void OPENSSL_free(void *addr);

char *OPENSSL_strdup(const char *str);
```

```
char *OPENSSL_strndup(const char *str, size_t s);
size_t OPENSSL_strlcat(char *dst, const char *src, size_t size);
size_t OPENSSL_strlcpy(char *dst, const char *src, size_t size);
void *OPENSSL_memdup(void *data, size_t s);
void *OPENSSL_clear_realloc(void *p, size_t old_len, size_t num);
void OPENSSL_clear_free(void *str, size_t num);
void OPENSSL_cleanse(void *ptr, size_t len);
void *CRYPTO_malloc(size_t num, const char *file, int line);
void *CRYPTO_zalloc(size_t num, const char *file, int line);
void *CRYPTO_realloc(void *p, size_t num, const char *file, int line);
void CRYPTO_free(void *str, const char *, int);
char *CRYPTO_strdup(const char *p, const char *file, int line);
char *CRYPTO_strndup(const char *p, size_t num, const char *file, int line);
void *CRYPTO_clear_realloc(void *p, size_t old_len, size_t num,
                           const char *file, int line);
void CRYPTO_clear_free(void *str, size_t num, const char *, int);
typedef void *(*CRYPTO_malloc_fn)(size_t num, const char *file, int line);
typedef void *(*CRYPTO_realloc_fn)(void *addr, size_t num, const char *file,
                                   int line);
typedef void (*CRYPTO_free_fn)(void *addr, const char *file, int line);
void CRYPTO_get_mem_functions(CRYPTO_malloc_fn *malloc_fn,
                              CRYPTO_realloc_fn *realloc_fn,
                              CRYPTO_free_fn *free_fn);
int CRYPTO_set_mem_functions(CRYPTO_malloc_fn malloc_fn,
                             CRYPTO_realloc_fn realloc_fn,
                             CRYPTO_free_fn free_fn);
void CRYPTO_get_alloc_counts(int *mcount, int *rcount, int *fcount);
env OPENSSL_MALLOC_FAILURES=... <application>
env OPENSSL_MALLOC_FD=... <application>
```

The following functions have been deprecated since OpenSSL 3.0, and can be hidden entirely by defining OPENSSL_API_COMPAT with a suitable version value, see openssl_user_macros(7):

```
int CRYPTO_mem_leaks(BIO *b);
```

```
int CRYPTO_mem_leaks_fp(FILE *fp);
int CRYPTO_mem_leaks_cb(int (*cb)(const char *str, size_t len, void *u),
                        void *u);
int CRYPTO_set_mem_debug(int onoff);
int CRYPTO_mem_ctrl(int mode);
int OPENSSL_mem_debug_push(const char *info);
int OPENSSL_mem_debug_pop(void);
int CRYPTO_mem_debug_push(const char *info, const char *file, int line);
int CRYPTO_mem_debug_pop(void);
```

DESCRIPTION

OpenSSL memory allocation is handled by the OPENSSL_xxx API. These are generally macro's that add the standard C __FILE__ and __LINE__ parameters and call a lower-level CRYPTO_xxx API. Some functions do not add those parameters, but exist for consistency.

OPENSSL_malloc_init() does nothing and does not need to be called. It is included for compatibility with older versions of OpenSSL.

OPENSSL_malloc(), OPENSSL_realloc(), and OPENSSL_free() are like the C malloc(), realloc(), and free() functions. OPENSSL_zalloc() calls memset() to zero the memory before returning.

OPENSSL_clear_realloc() and OPENSSL_clear_free() should be used when the buffer at addr holds sensitive information. The old buffer is filled with zero's by calling OPENSSL_cleanse() before ultimately calling OPENSSL_free().

OPENSSL_cleanse() fills ptr of size len with a string of 0's. Use OPENSSL_cleanse() with care if the memory is a mapping of a file. If the storage controller uses write compression, then it's possible that sensitive tail bytes will survive zeroization because the block of zeros will be compressed. If the storage controller uses wear leveling, then the old sensitive data will not be overwritten; rather, a block of 0's will be written at a new physical location.

OPENSSL_strdup(), OPENSSL_strndup() and OPENSSL_memdup() are like the equivalent C functions, except that memory is allocated by calling the OPENSSL_malloc() and should be released by calling OPENSSL_free().

OPENSSL_strlcpy(), OPENSSL_strlcat() and OPENSSL_strnlen() are equivalents of the common C library functions and are provided for portability.

If no allocations have been done, it is possible to "swap out" the default implementations for OPENSSL_malloc(), OPENSSL_realloc() and OPENSSL_free() and replace them with alternate versions.

CRYPTO_get_mem_functions() function fills in the given arguments with the function pointers for the current implementations. With CRYPTO_set_mem_functions(), you can specify a different set of functions. If any of malloc_fn, realloc_fn, or free_fn are NULL, then the function is not changed. While it's permitted to swap out only a few and not all the functions with CRYPTO_set_mem_functions(), it's recommended to swap them all out at once.

If the library is built with the "crypto-mdebug" option, then one function, CRYPTO_get_alloc_counts(), and two additional environment variables, OPENSSL_MALLOC_FAILURES and OPENSSL_MALLOC_FD, are available.

The function CRYPTO_get_alloc_counts() fills in the number of times each of CRYPTO_malloc(), CRYPTO_realloc(), and CRYPTO_free() have been called, into the values pointed to by mcount, rcount, and fcount, respectively. If a pointer is NULL, then the corresponding count is not stored.

The variable OPENSSL_MALLOC_FAILURES controls how often allocations should fail. It is a set of fields separated by semicolons, which each field is a count (defaulting to zero) and an optional atsign and percentage (defaulting to 100). If the count is zero, then it lasts forever. For example, "100;@25" or "100@0;0@25" means the first 100 allocations pass, then all other allocations (until the program exits or crashes) have a 25% chance of failing.

If the variable OPENSSL_MALLOC_FD is parsed as a positive integer, then it is taken as an open file descriptor. This is used in conjunction with OPENSSL_MALLOC_FAILURES described above. For every allocation it will log details about how many allocations there have been so far,

what percentage chance there is for this allocation failing, and whether it has actually failed. The following example in classic shell syntax shows how to use this (will not work on all platforms):

```
OPENSSL_MALLOC_FAILURES='200;@10'
export OPENSSL_MALLOC_FAILURES
OPENSSL_MALLOC_FD=3
export OPENSSL_MALLOC_FD
...app invocation... 3>/tmp/log$$
```

RETURN VALUES

OPENSSL_malloc_init(), OPENSSL_free(), OPENSSL_clear_free() CRYPTO_free(), CRYPTO_clear_free() and CRYPTO_get_mem_functions() return no value.

OPENSSL_malloc(), OPENSSL_zalloc(), OPENSSL_realloc(), OPENSSL_clear_realloc(), CRYPTO_malloc(), CRYPTO_zalloc(), CRYPTO_realloc(), CRYPTO_clear_realloc(), OPENSSL_strdup(), and OPENSSL_strndup() return a pointer to allocated memory or NULL on error.

CRYPTO_set_mem_functions() returns 1 on success or 0 on failure (almost always because allocations have already happened).

CRYPTO_mem_leaks(), CRYPTO_mem_leaks_fp(), CRYPTO_mem_leaks_cb(), CRYPTO_set_mem_debug(), and CRYPTO_mem_ctrl() are deprecated and are no-ops that always return -1. OPENSSL_mem_debug_push(), OPENSSL_mem_debug_pop(), CRYPTO_mem_debug_push(), and CRYPTO_mem_debug_pop() are deprecated and are no-ops that always return 0.

HISTORY

OPENSSL_mem_debug_push(), OPENSSL_mem_debug_pop(), CRYPTO_mem_debug_push(), CRYPTO_mem_debug_pop(), CRYPTO_mem_leaks(), CRYPTO_mem_leaks_fp(), CRYPTO_mem_leaks_cb(), CRYPTO_set_mem_debug(), CRYPTO_mem_ctrl() were deprecated in OpenSSL 3.0. The memory-leak checking has been deprecated in OpenSSL 3.0 in favor of clang's memory and leak sanitizer.

COPYRIGHT