Secrets
ABAP
Apex
C
C++
CloudFormation
COBOL
C#
CSS
Flex
Go
HTML
Java
JavaScript
Kotlin
Kubernetes
Objective C
PHP
PL/I
PL/SQL
Python
RPG
Ruby
Scala
Swift
Terraform
Text
TypeScript
T-SQL
VB.NET
VB6
XML

Objective C static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your OBJECTIVE C code

All rules 315 | Vulnerability 10 | Bug 75 | Security Hotspot 18 | Code Smell 212 | Quick Fix 13

Tags

Search by name...

"memset" should not be used to delete sensitive data
Vulnerability

POSIX functions should not be called with arguments that trigger buffer overflows
Vulnerability

Function-like macros should not be invoked without all of their arguments
Bug

The address of an automatic object should not be assigned to another object that may persist after the first object has ceased to exist
Bug

"pthread_mutex_t" should be unlocked in the reverse order they were locked
Bug

"pthread_mutex_t" should be properly initialized and destroyed
Bug

"pthread_mutex_t" should not be consecutively locked or unlocked twice
Bug

Functions with "noreturn" attribute should not return
Bug

"memcmp" should only be called with pointers to trivially copyable types with no padding
Bug

Stack allocated memory and non-owned memory should not be freed
Bug

Closed resources should not be accessed
Bug

Dynamically allocated memory should be released
Bug

Parameter values should be appropriate

Analyze your code

Bug | Critical | symbolic-execution

The standard C library includes a number of functions for string and memory manipulation. They take pointers and lengths as parameters. Passing `NULL` for the pointers will at best do nothing and at worst crash the application.

This rule raises an issue when the pointer passed to any of the following functions is `NULL`:

- `int aio_suspend(const struct aiocb* const aiocb_list[], int nitems, const struct timespec* timeout);`
- `void* bsearch(const void* key, const void* base, size_t nmemb, size_t size, int (*compar)(const void *, const void *));`
- `void encrypt(char block[64], int edflag);`
- `double erand48(unsigned short xsubi[3]);`
- `int fgetpwent_r(FILE* fp, struct passwd* pwbuf, char* buf, size_t buflen, struct passwd** pwbufp);`
- `char* fgets(char* str, int size, FILE* stream);`
- `wchar_t* fgetws(wchar_t* restrict ws, int n, FILE* restrict fp);`
- `int getgrgid_r(gid_t gid, struct group* grp, char* buffer, size_t bufsize, struct group** result);`
- `int getgrnam_r(const char* name, struct group* grp, char* buffer, size_t bufsize, struct group** result);`
- `int gethostbyaddr_r(const void* addr, socklen_t len, int type, struct hostent* ret, char* buf, size_t buflen, struct hostent** result, int* h_errnop);`
- `int gethostbyname_r(const char* name, struct hostent* ret, char* buf, size_t buflen, struct hostent** result, int* h_errnop);`
- `int gethostbyname2_r(const char* name, int af, struct hostent* ret, char* buf, size_t buflen, struct hostent** result, int* h_errnop);`
- `int gethostent_r(struct hostent* ret, char* buf, size_t buflen, struct hostent** result, int* h_errnop);`
- `int gethostname(char* name, size_t len);`
- `ssize_t getline(char** restrict linep, size_t* restrict linecapp, FILE* restrict stream);`
- `int getlogin_r(char* name, int len);`
- `int getnetbyaddr_r(uint32_t net, int type, struct netent* result_buf, char* buf, size_t buflen, struct netent** result, int* h_errnop);`
- `int getnetbyname_r(const char* name, struct netent* result_buf, char* buf, size_t buflen, struct netent** result, int* h_errnop);`
- `int getnetent_r(struct netent* result_buf, char* buf, size_t buflen, struct netent** result, int* h_errnop);`
- `int getnetgrent_r(char** host, char** user, char** domain, char* buf, int buflen);`
- `int getopt(int argc, char* const argv[], const char* optstring);`
- `int getopt_long(int argc, char* const* argv, const char* optstring, const struct option* longopts, int* longindex);`
- `int getopt_long_only(int argc, char* const* argv, const char* optstring, const struct option* longopts, int* longindex);`
- `int getprotobyname_r(const char* name, struct protoent* result_buf, char* buf, size_t buflen, struct protoent**`

Freed memory should not be used

Bug

Memory locations should not be released more than once

Bug

Memory access should be explicitly bounded to prevent buffer overflows

Bug

Printf-style format strings should not lead to unexpected behavior at runtime

Bug

Recursion should not be infinite

Bug

Resources should be closed

Bug

Hard-coded credentials are security-sensitive

Security Hotspot

"goto" should jump to labels declared later in the same function

Code Smell

Only standard forms of the "defined" directive should be used

Code Smell

Switch labels should not be nested inside non-switch blocks

Code Smell

result);

- `int getprotobynumber_r(int proto, struct protoent* result_buf, char* buf, size_t buflen, struct protoent** result);`
- `int getprotoent_r(struct protoent* result_buf, char* buf, size_t buflen, struct protoent** result);`
- `int getpwent_r(struct passwd* pwbuf, char* buf, size_t buflen, struct passwd** pwbufp);`
- `int getpwnam_r(const char* name, struct passwd* pwd, char* buf, size_t buflen, struct passwd** result);`
- `int getpwuid_r(uid_t uid, struct passwd* pwd, char* buf, size_t buflen, struct passwd** result);`
- `int getservbyname_r(const char* name, const char* proto, struct servent* result_buf, char* buf, size_t buflen, struct servent** result);`
- `int getservbyport_r(int port, const char* proto, struct servent* result_buf, char* buf, size_t buflen, struct servent** result);`
- `int getservent_r(struct servent* result_buf, char* buf, size_t buflen, struct servent** result);`
- `char* initstate(unsigned long seed, char* state, long n);`
- `long jrand48(unsigned short xseed[3]);`
- `void lcong48(unsigned short p[7]);`
- `void* lfind(const void* key, const void* base, size_t* nelp, size_t width, int (*compar)(const void *, const void *));`
- `int lio_listio(int mode, struct aiocb* const aiocb_list[], int nitems, struct sigevent* sevp);`
- `void* lsearch(const void* key, void* base, size_t* nelp, size_t width, int (*compar)(const void *, const void *));`
- `int mblen(const char* mbchar, size_t nbytes);`
- `size_t mbsnrtowcs(wchar_t* restrict dst, const char** restrict src, size_t nms, size_t len, mbstate_t* restrict ps);`
- `ssize_t mq_receive(mqd_t mqdes, char* msg_ptr, size_t msg_len, unsigned* msg_prio);`
- `int mq_send(mqd_t mqdes, const char* msg_ptr, size_t msg_len, unsigned msg_prio);`
- `ssize_t mq_timedreceive(mqd_t mqdes, char* msg_ptr, size_t msg_len, unsigned* msg_prio, const struct timespec* abs_timeout);`
- `int mq_timedsend(mqd_t mqdes, const char* msg_ptr, size_t msg_len, unsigned msg_prio, const struct timespec* abs_timeout);`
- `long nrand48(unsigned short xseed[3]);`
- `void posix_trace_event(trace_event_id_t event_id, const void* restrictdata_ptr, size_t data_len);`
- `int posix_trace_trygetnext_event(trace_id_t trid, struct posix_trace_event_info* restrict event, void* restrict data, size_t num_bytes, size_t* restrict data_len, int* restrict unavailable);`
- `ssize_t pread(int fd, void* buf, size_t nbytes, off_t offset);`
- `ssize_t preadv(int fd, const struct iovec* iov, int iovcnt, off_t offset);`
- `ssize_t preadv2(int fd, const struct iovec* iov, int iovcnt, off_t offset, int flags);`
- `int pthread_attr_setstack(pthread_attr_t* attr, void* stackaddr, size_t stacksize);`
- `ssize_t pwrite(int fd, const void* buf, size_t count, off_t offset);`
- `ssize_t pwritev(int fd, const struct iovec* iov, int iovcnt, off_t offset);`
- `ssize_t pwritev2(int fd, const struct iovec* iov, int iovcnt, off_t offset, int flags);`
- `void qsort(void* base, size_t nmemb, size_t size, int (*compar)(const void *, const void *));`
- `void qsort_r(void* base, size_t nmemb, size_t size, void* thunk, int (*compar)(void *, const void *, const void *));`
- `ssize_t read(int fildes, void* buf, size_t nbyte);`
- `ssize_t readlink(const char* restrict path, char* restrict buf, size_t bufsize);`
- `int readlinkat(int dirfd, const char* pathname, char* buf, size_t bufsiz);`
- `ssize_t readv(int fd, const struct iovec* iov, int iovcnt);`
- `ssize_t recv(int s, void* buf, size_t len, int flags);`
- `ssize_t recvfrom(int s, void* buf, size_t len, int flags, struct sockaddr* restrict from, socklen_t* restrict fromlen);`
- `unsigned short* seed48(unsigned short xseed[3]);`
- `int semop(int semid, struct sembuf* array, size_t nops);`
- `int semtimedop(int semid, struct sembuf* sops, unsigned nsops, struct timespec* timeout);`

- ssize_t send(int socket, const void* buffer, size_t length, int flags);
- ssize_t sendto(int socket, const void* message, size_t length, int flags, const struct sockaddr* dest_addr, socklen_t dest_len);
- void setbuf(FILE* restrict stream, char* restrict buf);
- void setbufer(FILE* restrict stream, char* restrict buf, size_t size);
- int socketpair(int domain, int type, int protocol, int* sv);
- size_t strftime(char* restrict buf, size_t maxsize, const char* restrict format, const struct tm* restrict timeptr);
- void swab(const void* restrict src, void* restrict dst, ssize_t len);
- int ttyname_r(int fd, char* buf, size_t len);
- int utimes(const char* path, const struct timeval* times);
- int vswprintf(wchar_t* restrict ws, size_t n, const wchar_t* restrict format, va_list ap);
- wchar_t* wcpncpy(wchar_t* s1, wchar_t* s2, size_t n);
- size_t wcsftime(wchar_t* restrict wcs, size_t maxsize, const wchar_t* restrict format, const struct tm* restrict timeptr);
- int wcsncasecmp(const wchar_t* s1, const wchar_t* s2, size_t n);
- int wcsncmp(const wchar_t* s1, const wchar_t* s2, size_t n);
- wchar_t* wcsncpy(wchar_t* restrict s1, const wchar_t* restrict s2, size_t n);
- size_t wcsnlen(const wchar_t* s, size_t maxlen);
- size_t wcsnrtombs(char* dest, const wchar_t** src, size_t nwc, size_t len, mbstate_t* ps);
- int wcswidth(const wchar_t* s, size_t n);
- size_t wcsxfrm(wchar_t* restrict ws1, const wchar_t* restrict ws2, size_t n);
- int wmemcmp(const wchar_t* s1, const wchar_t* s2, size_t n);
- wchar_t* wmemcpy(wchar_t* restrict s1, const wchar_t* restrict s2, size_t n);
- wchar_t* wmemmove(wchar_t* s1, const wchar_t* s2, size_t n);
- wchar_t* wmemset(wchar_t* s, wchar_t c, size_t n);
- ssize_t writev(int fd, const struct iovec* iov, int iovcnt);
- void *memcpy(void *dest, const void *src, size_t n);
- void *memmove(void *dest, const void *src, size_t n);
- void *memccpy(void *dest, const void *src, int c, size_t n);
- void *memset(void *s, int c, size_t n);
- int memcmp(const void *s1, const void *s2, size_t n);
- char *strcpy(char *dest, const char *src);
- char *strncpy(char *dest, const char *src, size_t n);
- char *strcat(char *dest, const char *src);
- char *strncat(char *dest, const char *src, size_t n);
- int strcmp(const char *s1, const char *s2);
- int strncmp(const char *s1, const char *s2, size_t n);
- void *mempcpy(void *dest, const void *src, size_t n);
- size_t strlen(const char *s);
- size_t strnlen(const char *s, size_t maxlen);
- void bcopy(const void *src, void *dest, size_t n);
- void bzero(void *s, size_t n);
- int bcmp(const void *s1, const void *s2, size_t n);
- int strcasecmp(const char *s1, const char *s2);
- int strncasecmp(const char *s1, const char *s2, size_t n);
- char *strsep(char **stringp, const char *delim);
- char *stpcpy(char *dest, const char *src);

Noncompliant Code Example

```
memcpy(NULL, src, 10); // Noncompliant, null pointer
```

Available In:

sonarcloud | sonarqube Developer Edition