POS52-C. Do not perform operations that can block while holding a POSIX lock

If a lock is being held and an operation that can block is performed, any other thread that needs to acquire that lock may also block. This condition can degrade the performance of a system or cause a deadlock to occur.

Blocking calls include, but are not limited to: network, file, and console I/O. This rule is a specific instance of CON05-C. Do not perform operations that can block while holding a lock using POSIX threads.

Noncompliant Code Example

This noncompliant code example demonstrates an occurrence of a blocking call that waits to receive data on a socket while a mutex is locked. The `recv()` call blocks until data arrives on the socket. While it is blocked, other threads that are waiting for the lock are also blocked.

Although this example is specific to network I/O, the `recv()` call could be replaced with any blocking call, and the same behavior would occur.

```
pthread_mutexattr_t attr;
pthread_mutex_t mutex;

void thread_foo(void *ptr) {
  uint32_t num;
  int result;
  int sock;

  /* sock is a connected TCP socket */

  if ((result = pthread_mutex_lock(&mutex)) != 0) {
    /* Handle Error */
  }

  if ((result = recv(sock, (void *)&num, sizeof(uint32_t), 0)) < 0) {
    /* Handle Error */
  }

  /* ... */

  if ((result = pthread_mutex_unlock(&mutex)) != 0) {
    /* Handle Error */
  }
}

int main() {
  pthread_t thread;
  int result;

  if ((result = pthread_mutexattr_settype(
      &mutex, PTHREAD_MUTEX_ERRORCHECK)) != 0) {
    /* Handle Error */
  }

  if ((result = pthread_mutex_init(&mutex, &attr)) != 0) {
    /* Handle Error */
  }

  if (pthread_create(&thread, NULL,(void *)& thread_foo, NULL) != 0) {
    /* Handle Error */
  }

  /* ... */

  pthread_join(thread, NULL);

  if ((result = pthread_mutex_destroy(&mutex)) != 0) {
    /* Handle Error */
  }

  return 0;
}
```

Compliant Solution (Block while Not Locked)

This compliant solution performs the `recv()` call when the lock has not been acquired. The blocking behavior consequently affects only the thread that called the blocking function.

```
void thread_foo(void *ptr) {
  uint32_t num;
  int result;
  int sock;

  /* sock is a connected TCP socket */

  if ((result = recv(sock, (void *)&num, sizeof(uint32_t), 0)) < 0) {
    /* Handle Error */
  }

  if ((result = pthread_mutex_lock(&mutex)) != 0) {
    /* Handle Error */
  }

  /* ... */

  if ((result = pthread_mutex_unlock(&mutex)) != 0) {
    /* Handle Error */
  }
}
```

Compliant Solution (Use a Nonblocking Call)

This compliant solution performs the `recv()` call with the parameter `o_nonblock`, which causes the call to fail if no messages are available on the socket:

```
void thread_foo(void *ptr) {
  uint32_t num;
  int result;

  /* sock is a connected TCP socket */

  if ((result = recv(sock, (void *)&num, sizeof(uint32_t), O_NONBLOCK)) < 0) {
    /* Handle Error */
  }

  if ((result = pthread_mutex_lock(&mutex)) != 0) {
    /* Handle Error */
  }

  /* ... */

  if ((result = pthread_mutex_unlock(&mutex)) != 0) {
    /* Handle Error */
  }
}
```

Exceptions

POS52-C-EX1: A thread may block while holding one or more locks and waiting to acquire another lock. When acquiring multiple locks, the order of locking must avoid deadlock, as specified in CON35-C. Avoid deadlock by locking in a predefined order.

Risk Assessment

Blocking or lengthy operations performed within synchronized regions could result in a deadlocked or an unresponsive system.

Rule	Severity	Likelihood	Remediation Cost	Priority	Level
POS52-C	Low	Probable	High	P2	L3

Automated Detection

Tool	Version	Checker	Description

CodeSonar	5.2p0	**CONCURRENCY.STARVE.BLOCKING**	Blocking in Critical Section
Klocwork	2018	**CONC.SLEEP**	
Parasoft C/C++test	10.4.2	**CERT_C-POS52-a**	Do not use blocking functions while holding a lock
Polyspace Bug Finder	R2019b	CERT C: Rule POS52-C	Checks for blocking operation while holding lock (rule partially covered)
PRQA QA-C	9.7	**4966, 4967**	

Related Guidelines

Key here (explains table format and definitions)

Taxonomy	Taxonomy item	Relationship
CERT C	LCK09-J. Do not perform operations that can block while holding a lock	Prior to 2018-01-12: CERT: Unspecified Relationship
CWE 2.11	CWE-557	2017-07-10: CERT: Rule subset of CWE

CERT-CWE Mapping Notes

Key here for mapping notes

CWE-557 and POS52-C

CWE-557 = Union(POS52-C, list) where list =

- Concurrency issues besides blocking while holding a POSIX lock

Bibliography

[Barney 2010]	POSIX Threads Programming
[Open Group]	pthread_cancel() recv()