rules.sonarsource.com

C static code analysis: POSIX functions should not be called with arguments that trigger buffer overflows

2 minutes

Array overruns and buffer overflows happen when memory access accidentally goes beyond the boundary of the allocated array or buffer. These overreaching accesses cause some of the most damaging, and hard to track defects.

When the buffer overflow happens while reading a buffer, it can expose sensitive data that happens to be located next to the buffer in memory. When it happens while writing a buffer, it can be used to inject code or to wipe out sensitive memory.

This rule detects when a POSIX function takes one argument that is a buffer and another one that represents the size of the buffer, but the two arguments do not match.

Noncompliant Code Example

```
char array[10];
initialize(array);
void *pos = memchr(array, '@', 42); // Noncompliant, buffer overflow that could expose sensitive data
```

Compliant Solution

```
char array[10];
initialize(array);
void *pos = memchr(array, '@', 10);
```

Exceptions

Functions related to sockets using the type `socklen_t` are not checked. This is because these functions are using a C-style polymorphic pattern using `union`. It relies on a mismatch between allocated memory and sizes of structures and it creates false positives.

See

- OWASP Top 10 2017 Category A9 - Using Components with Known Vulnerabilities
- MITRE, CWE-119 - Improper Restriction of Operations within the Bounds of a Memory Buffer
- MITRE, CWE-131 - Incorrect Calculation of Buffer Size
- MITRE, CWE-788 - Access of Memory Location After End of Buffer
- CERT, ARR30-C. - Do not form or use out-of-bounds pointers or array subscripts
- CERT, STR50-CPP. - Guarantee that storage for strings has sufficient space for character data and the null terminator