rules.sonarsource.com

C++ static code analysis: Using "tmpnam", "tmpnam_s" or "tmpnam_r" is security-sensitive

3-4 minutes

The functions "tmpnam", "tmpnam_s" and "tmpnam_r" are all used to return a file name that does not match an existing file, in order for the application to create a temporary file. However, even if the file did not exist at the time those functions were called, it might exist by the time the application tries to use the file name to create the files. This has been used by hackers to gain access to files that the application believed were trustworthy.

There are alternative functions that, in addition to creating a suitable file name, create and open the file and return the file handler. Such functions are protected from this attack vector and should be preferred. About the only reason to use these functions would be to create a temporary folder, not a temporary file.

Additionally, these functions might not be thread-safe, and if you don’t provide them buffers of sufficient size, you will have a buffer overflow.

Ask Yourself Whether

- There is a possibility that several threads call any of these

functions simultaneously

- There is a possibility that the resulting file is opened without forcing its creation, meaning that it might have unexpected access rights
- The buffers passed to these functions are respectively smaller than
- `L_tmpnam` for `tmpnam`
- `L_tmpnam_s` for `tmpnam_s`
- `L_tmpnam` for `tmpnam_r`

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

- Use a function that directly opens the temporary file, such a `tmpfile`, `tmpfile_s`, `mkstemp` or `mkstemps` (the last two allow more accurate control of the file name).
- If you can’t get rid of these functions, when using the generated name to open the file, use a function that forces the creation of the file and fails if the file already exists.

Sensitive Code Example

```
int f(char *tempData) {
  char *path = tmpnam(NULL); // Sensitive
  FILE* f = fopen(tmpnam, "w");
  fputs(tempData, f);
  fclose(f);
}
```

Compliant Solution

```
int f(char *tempData) {
  // The file will be opened in "wb+" mode, and will be
automatically removed on normal program exit
  FILE* f = tmpfile(); // Compliant
  fputs(tempData, f);
  fclose(f);
}
```

See

- OWASP Top 10 2021 Category A1 - Broken Access Control
- OWASP Top 10 2021 Category A6 - Vulnerable and Outdated Components
- OWASP Top 10 2017 Category A9 - Using Components with Known Vulnerabilities
- MITRE, CWE-377 - Insecure Temporary File
- CERT, CON33-C. - Avoid race conditions when using library functions
- CERT, FIO21-C. - Do not create temporary files in shared directories

Available In: