# CWE Detail – CWE-379

## Description

The product creates a temporary file in a directory whose permissions allow unintended actors to determine the file's existence or otherwise access that file.

## Extended Description

On some operating systems, the fact that the temporary file exists may be apparent to any user with sufficient privileges to access that directory. Since the file is visible, the application that is using the temporary file could be known. If one has access to list the processes on the system, the attacker has gained information about what the user is doing at that time. By correlating this with the applications the user is running, an attacker could potentially discover what a user's actions are. From this, higher levels of security could be breached.

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Observed Examples (CVEs)

**•** CVE-2022-27818: A hotkey daemon written in Rust creates a domain socket file underneath /tmp, which is accessible by any user.

**•** CVE-2021-21290: A Java-based application for a rapid-development framework uses File.createTempFile() to create a random temporary file with insecure default permissions.

## Modes of Introduction

**•** Implementation: N/A

## Common Consequences

**•** Impact: Read Application Data — Notes: Since the file is visible and the application which is using the temp file could be known, the attacker has gained information about what the user is doing at that time.

## Potential Mitigations

**•** Requirements: Many contemporary languages have functions which properly handle this condition. Older C temp file functions are especially susceptible. (Effectiveness: N/A)

**•** Implementation: Try to store sensitive tempfiles in a directory which is not world readable -- i.e., per-user directories. (Effectiveness: N/A)

**•** Implementation: Avoid using vulnerable temp file functions. (Effectiveness: N/A)

## Applicable Platforms

**•** None (Class: Not Language-Specific, Prevalence: Undetermined)

## Demonstrative Examples

**•** However, within this C/C++ code the method tmpfile() is used to create and open the temp file. The tmpfile() method works the same way as the fopen() method would with read/write permission, allowing attackers to read potentially sensitive information contained in the temp file or modify the contents of the file.