# CWE Detail – CWE-252

## Description

The product does not check the return value from a method or function, which can prevent it from detecting unexpected states and conditions.

## Extended Description

Two common programmer assumptions are "this function call can never fail" and "it doesn't matter if this function call fails". If an attacker can force the function to fail or otherwise return a value that is not expected, then the subsequent program logic could lead to a vulnerability, because the product is not in a state that the programmer assumes. For example, if the program calls a function to drop privileges but does not check the return code to ensure that privileges were successfully dropped, then the program will continue to operate with the higher privileges.

## Threat-Mapped Scoring

Score: 1.8

Priority: P4 - Informational (Low)

## Observed Examples (CVEs)

**•** CVE-2020-17533: Chain: unchecked return value (CWE-252) of some functions for policy enforcement leads to authorization bypass (CWE-862)

**•** CVE-2020-6078: Chain: The return value of a function returning a pointer is not checked for success (CWE-252) resulting in the later use of an uninitialized variable (CWE-456) and a null pointer dereference (CWE-476)

**•** CVE-2019-15900: Chain: sscanf() call is used to check if a username and group exists, but the return value of sscanf() call is not checked (CWE-252), causing an uninitialized variable to be checked (CWE-457), returning success to allow authorization bypass for executing a privileged (CWE-863).

**•** CVE-2007-3798: Unchecked return value leads to resultant integer overflow and code execution.

**•** CVE-2006-4447: Program does not check return value when invoking functions to drop privileges, which could leave users with higher privileges than expected by forcing those functions to fail.

**•** CVE-2006-2916: Program does not check return value when invoking functions to drop privileges, which could leave users with higher privileges than expected by forcing those functions to fail.

**•** CVE-2008-5183: chain: unchecked return value can lead to NULL dereference

**•** CVE-2010-0211: chain: unchecked return value (CWE-252) leads to free of invalid, uninitialized pointer (CWE-824).

**•** CVE-2017-6964: Linux-based device mapper encryption program does not check the return value of setuid and setgid allowing attackers to execute code with unintended privileges.

**•** CVE-2002-1372: Chain: Return values of file/socket operations are not checked (CWE-252), allowing resultant consumption of file descriptors (CWE-772).

## Modes of Introduction

**•** Implementation: N/A

## Common Consequences

**•** Impact: Unexpected State, DoS: Crash, Exit, or Restart — Notes: An unexpected return value could place the system in a state that could lead to a crash or other unintended behaviors.

## Potential Mitigations

**•** Implementation: Check the results of all functions that return a value and verify that the value is expected. (Effectiveness: High)

**•** Implementation: Ensure that you account for all possible return values from the function. (Effectiveness: N/A)

**•** Implementation: When designing a function, make sure you return a value or throw an exception in case of an error. (Effectiveness: N/A)

## Applicable Platforms

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

## Demonstrative Examples

**•** The programmer expects that when fgets() returns, buf will contain a null-terminated string of length 9 or less. But if an I/O error occurs, fgets() will not null-terminate buf. Furthermore, if the end of the file is reached before any characters are read, fgets() returns without writing anything to buf. In both of these situations, fgets() signals that something unusual has happened by returning NULL, but in this code, the warning will not be noticed. The lack of a null terminator in buf can result in a buffer overflow in the subsequent call to strcpy().

**•** If returnChunkSize() happens to encounter an error it will return -1. Notice that the return value is not checked before the memcpy operation (CWE-252), so -1 can be passed as the size argument to memcpy() (CWE-805). Because memcpy() assumes that the value is unsigned, it will be interpreted as MAXINT-1 (CWE-195), and therefore will copy far more memory than is likely available to the destination buffer (CWE-787, CWE-788).

**•** The traditional defense of this coding error is: "If my program runs out of memory, it will fail. It doesn't matter whether I handle the error or allow the program to die with a segmentation fault when it tries to dereference the null pointer." This argument ignores three important considerations:

**•** The code loops through a set of users, reading a private data file for each user. The programmer assumes that the files are always 1 kilobyte in size and therefore ignores the return value from Read(). If an attacker can create a smaller file, the program will recycle the remainder of the data from the previous user and treat it as though it belongs to the attacker.

**•** The following code does not check to see if the string returned by the Item property is null before calling the member function Equals(), potentially causing a NULL dereference.

**•** The traditional defense of this coding error is: "I know the requested value will always exist because.... If it does not exist, the program cannot perform the desired behavior so it doesn't matter whether I handle the error or allow the program to die dereferencing a null value." But attackers are skilled at finding unexpected paths through programs, particularly when exceptions are involved.

**•** In .NET, it is not uncommon for programmers to misunderstand Read() and related methods that are part of many System.IO classes. The stream and reader classes do not consider it to be unusual or exceptional if only a small amount of data becomes available. These classes simply add the small amount of data to the return buffer, and set the return value to the number of bytes or characters read. There is no guarantee that the amount of data returned is equal to the amount of data requested.

**•** N/A

**•** If an attacker provides an address that appears to be well-formed, but the address does not resolve to a hostname, then the call to gethostbyaddr() will return NULL. Since the code does not check the return value from gethostbyaddr (CWE-252), a NULL pointer dereference
 (CWE-476) would then occur in the call to strcpy().

**•** However, the code does not check the value returned by pthread\_mutex\_lock() for errors. If pthread\_mutex\_lock() cannot acquire the mutex for any reason, the function may introduce a race condition into the program and result in undefined behavior.