

 \mathbb{X} Flex

-GO Go

HTML 5

Java JavaScript

Kotlin

Kubernetes

Objective C

PHP

PL/I

PL/SQL

Python

RPG

Ruby

Scala

Swift

Terraform

Text

TypeScript

T-SQL

VB.NET

VB6

XML

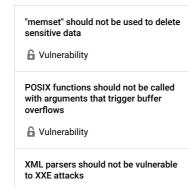


C static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your C code

⊗ Code O Quick 14 ΑII 311 Security 18 206 6 Vulnerability (13) ₩ Bug (74) rules Hotspot Smell

Tags



Vulnerability

Function-like macros should not be invoked without all of their arguments

₩ Bug

The address of an automatic object should not be assigned to another object that may persist after the first object has ceased to exist

👬 Bug

"pthread_mutex_t" should be unlocked in the reverse order they were locked

"pthread_mutex_t" should be properly initialized and destroyed

Bua

"pthread_mutex_t" should not be consecutively locked or unlocked

Bug

Functions with "noreturn" attribute should not return

₩ Bua

"memcmp" should only be called with pointers to trivially copyable types with no padding

🖷 Bug

Expanding archive files without controlling resource consumption is security-sensitive

Analyze your code



cwe cert owasp

Search by name.

Successful Zip Bomb attacks occur when an application expands untrusted archive files without controlling the size of the expanded data, which can lead to denial of service. A Zip bomb is usually a malicious archive file of a few kilobytes of compressed data but turned into gigabytes of uncompressed data. To achieve this extreme compression ratio, attackers will compress irrelevant data (eg. a long string of repeated bytes).

Ask Yourself Whether

Archives to expand are untrusted and:

- There is no validation of the number of entries in the archive.
- There is no validation of the total size of the uncompressed data.
- There is no validation of the ratio between the compressed and uncompressed archive entry.

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

Recommended Secure Coding Practices

- Define and control the threshold for maximum total size of the uncompressed
- Count the number of file entries extracted from the archive and abort the extraction if their number is greater than a predefined threshold, in particular it's not recommended to recursively expand archives (an entry of an archive could be

Sensitive Code Example

```
#include <archive.h>
#include <archive_entry.h>
void f(const char *filename, int flags) {
  struct archive_entry *entry;
  struct archive *a = archive_read_new();
  struct archive *ext = archive write disk new();
  archive write disk set options(ext, flags);
  archive_read_support_format_tar(a);
  if ((archive_read_open_filename(a, filename, 10240))) {
   return;
  }
  for (;;) {
   int r = archive read next header(a, &entry);
    if (r == ARCHIVE_EOF) {
     break;
   if (r != ARCHIVE_OK) {
     return;
```

Stack allocated memory and nonowned memory should not be freed

🕕 Bug

Closed resources should not be accessed

👬 Bug

Dynamically allocated memory should be released

Bug

Freed memory should not be used

```
archive_read_close(a);
archive_read_free(a);
archive_write_close(ext);
archive_write_free(ext);
```

Compliant Solution

```
#include <archive.h>
#include <archive_entry.h>
int f(const char *filename, int flags) {
 const int max_number_of_extraced_entries = 1000;
 const int64 t max file size = 1000000000; // 1 GB
  int number_of_extraced_entries = 0;
 int64_t total_file_size = 0;
 struct archive_entry *entry;
 struct archive *a = archive read new():
 struct archive *ext = archive_write_disk_new();
 archive_write_disk_set_options(ext, flags);
 archive_read_support_format_tar(a);
 int status = 0;
 if ((archive_read_open_filename(a, filename, 10240))) {
 }
  for (;;) {
   number_of_extraced_entries++;
   if (number_of_extraced_entries > max_number_of_extraced_e
     status = 1;
     break;
   int r = archive_read_next_header(a, &entry);
   if (r == ARCHIVE_EOF) {
     break;
   if (r != ARCHIVE_OK) {
     status = -1;
     break;
   int file_size = archive_entry_size(entry);
   total_file_size += file_size;
   if (total_file_size > max_file_size) {
     status = 1;
     break;
 }
 archive_read_close(a);
 archive_read_free(a);
 archive_write_close(ext);
 archive write free(ext);
 return status;
```

- OWASP Top 10 2021 Category A1 Broken Access Control
- OWASP Top 10 2021 Category A5 Security Misconfiguration
- OWASP Top 10 2017 Category A6 Security Misconfiguration
- \bullet MITRE, CWE-409 Improper Handling of Highly Compressed Data (Data Amplification)
- $\bullet \ \underline{\text{CERT, IDS04-J.}} \ \ \text{Safely extract files from ZipInputStream}$
- bamsoftware.com A better Zip Bomb

Available In:

sonarcloud 🚳 sonarqube Developer Edition

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.

Privacy Policy