Secrets
ABAP
Apex
C
C++
CloudFormation
COBOL
C#
CSS
Flex
Go
HTML
Java
JavaScript
Kotlin
Objective C
PHP
PL/I
PL/SQL
Python
RPG
Ruby
Scala
Swift
Terraform
Text
TypeScript
T-SQL
VB.NET
VB6
XML

PHP static code analysis

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

All rules 268 | Vulnerability 40 | Bug 51 | Security Hotspot 33 | Code Smell 144

Tags

Search by name...

Bug

All branches in a conditional structure should not have exactly the same implementation

Bug

The output of functions that don't return anything should not be used

Bug

Unary prefix operators should not be repeated

Bug

"=+" should not be used instead of "+="

Bug

A "for" loop update clause should move the counter in the right direction

Bug

Return values from functions without side effects should not be ignored

Bug

Values should not be uselessly incremented

Bug

Related "if/else if" statements and "cases" in a "switch" should not have the same condition

Bug

Objects should not be created to be dropped immediately without being used

Bug

Identical expressions should not be used on both sides of a binary operator

Bug

All code should be reachable

Bug

Expanding archive files without controlling resource consumption is security-sensitive

Analyze your code

Security Hotspot | Critical | cwe owasp

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 ratio between compressed and uncompressed data, in general the data compression ratio for most of the legit archives is 1 to 3.
- Define and control the threshold for maximum total size of the uncompressed data.
- 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 also an archive).

Sensitive Code Example

For ZipArchive module:

```
$zip = new ZipArchive();
if ($zip->open($file) === true) {
    $zip->extractTo('.'); // Sensitive
    $zip->close();
}
```

For Zip module:

```
$zip = zip_open($file);
while ($file = zip_read($zip)) {
    $filename = zip_entry_name($file);
    $size = zip_entry_filesize($file);

    if (substr($filename, -1) !== '/') {
        $content = zip_entry_read($file, zip_entry_filesize(
        file_put_contents($filename, $content);
    } else {
        mkdir($filename);
```

Loops with at most one iteration should be refactored

Bug

Short-circuit logic should be used to prevent null pointer dereferences in conditionals

Bug

Variables should not be self-assigned

Bug

Useless "if(true) {...}" and "if(false){...}" blocks should be removed

```
    }
}
zip_close($zip);
```

Compliant Solution

For ZipArchive module:

```
define('MAX_FILES', 10000);
define('MAX_SIZE', 1000000000); // 1 GB
define('MAX_RATIO', 10);
define('READ_LENGTH', 1024);

$fileCount = 0;
$totalSize = 0;

$zip = new ZipArchive();
if ($zip->open($file) === true) {
    for ($i = 0; $i < $zip->numFiles; $i++) {
        $filename = $zip->getNameIndex($i);
        $stats = $zip->statIndex($i);

        // Prevent ZipSlip path traversal (S6096)
        if (strpos($filename, '../') !== false || substr($fi
            throw new Exception();
        }

        if (substr($filename, -1) !== '/') {
            $fileCount++;
            if ($fileCount > MAX_FILES) {
                // Reached max. number of files
                throw new Exception();
            }

            $fp = $zip->getStream($filename); // Compliant
            $currentSize = 0;
            while (!feof($fp)) {
                $currentSize += READ_LENGTH;
                $totalSize += READ_LENGTH;

                if ($totalSize > MAX_SIZE) {
                    // Reached max. size
                    throw new Exception();
                }

                // Additional protection: check compression
                if ($stats['comp_size'] > 0) {
                    $ratio = $currentSize / $stats['comp_siz
                    if ($ratio > MAX_RATIO) {
                        // Reached max. compression ratio
                        throw new Exception();
                    }
                }

                file_put_contents($filename, fread($fp, READ
            }

            fclose($fp);
        } else {
            mkdir($filename);
        }
    }
    $zip->close();
}
```

For Zip module:

```
define('MAX_FILES', 10000);
define('MAX_SIZE', 1000000000); // 1 GB
define('MAX_RATIO', 10);
define('READ_LENGTH', 1024);

$fileCount = 0;
$totalSize = 0;

$zip = zip_open($file);
while ($file = zip_read($zip)) {
    $filename = zip_entry_name($file);
```

```
    // Prevent ZipSlip path traversal (S6096)
    if (strpos($filename, '../') !== false || substr($filena
        throw new Exception();
    }

    if (substr($filename, -1) !== '/') {
        $fileCount++;
        if ($fileCount > MAX_FILES) {
            // Reached max. number of files
            throw new Exception();
        }

        $currentSize = 0;
        while ($data = zip_entry_read($file, READ_LENGTH)) {
            $currentSize += READ_LENGTH;
            $totalSize += READ_LENGTH;

            if ($totalSize > MAX_SIZE) {
                // Reached max. size
                throw new Exception();
            }

            // Additional protection: check compression rati
            if (zip_entry_compressedsize($file) > 0) {
                $ratio = $currentSize / zip_entry_compressed
                if ($ratio > MAX_RATIO) {
                    // Reached max. compression ratio
                    throw new Exception();
                }
            }

            file_put_contents($filename, $data, FILE_APPEND)
        }
    } else {
        mkdir($filename);
    }
}
zip_close($zip);
```

See

- 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
- MITRE, CWE-409 - Improper Handling of Highly Compressed Data (Data Amplification)
- bamsoftware.com - A better Zip Bomb

Available In:

sonarcloud | sonarqube