**Command Injection**

Command injection attacks are possible when an application passes unsafe user supplied data via forms, cookies, HTTP headers to a system shell.

The following code prompts a user to enter an IP address in which the underlying shell performs the ping and returns the result back to the end user.

The following code allows subsequent commands to be appended to the input via the use of the shell\_exec API.

<html lang="en">

<head></head>

<body>

<p>Enter your IP/host to ping.

<form method='get' action=''>

<div class="form-group">

<label></label>

<input class="form-control" width="50%" placeholder="" name="target"></input> <br>

<div align="left"> <button class="btn btn-default" type="submit">Submit Button</button></div>

</div>

</form>

</p>

<?php

if (isset($\_REQUEST['target'])) {

$target = $\_REQUEST['target'];

//echo &target;

if($target){

if (stristr(php\_uname('s'), 'Windows NT')) {

$cmd = shell\_exec( 'ping ' . $target );

echo '<pre>'.$cmd.'</pre>';

} else {

$cmd = shell\_exec( 'ping -c 3 ' . $target );

echo '<pre>'.$cmd.'</pre>';

}

}

}

?>

</body>

</html>

**Tasks**

A cheat sheet for the tasks below can be located at:

https://hackersonlineclub.com/command-injection-cheatsheet/

1. View the contents of the directory in which the above php file is located.
2. Echo the contents of the above php in the browser
3. Echo JavaScript so it will execute in the victims browser
4. View the network configuration of the webserver
5. Write a .php file to the server. The .php file will take an input from the user (a location on the server). The .php file will return the contents of the supplied directory.

Fix the vulnerable code