**Challenge URL:** /dvwa/vulnerabilities/exec/
**Objective:** Remotely, find out the user of the web service on the OS, as well as the machine's hostname via RCE.
**Tools needed:** None

*Did you remember to read this section's README?*

## The Guide

### Probing the Code

When we start the challenge, we see what looks to be a ping tool:

Seems simple enough. Let's quickly test it with a valid IP.

And with an invalid input:

So it seems that the code takes our input, passes it as an argument to the command `ping`, and executes the string "`ping [input]`". If it works, we'll see the unformatted output of the command. If it doesn't, we'll see nothing. That's useful information, since we can easily tell if our future command injection is working sucessfully based on the output we see.

Now that we know how the code works, we need to figure out two things: which commands to execute, and how to chain them together in a way that all commands execute and display their output successfully.

### Which Commands to Execute

We have two pieces of information to collect: the name of the current user of the web service, and the machine's hostname. We can Google this info rather easily, or test it on our Kali box. For the user, we can run the command `whoami` without arguments. For the hostname, we use the appropriately-named `hostname` without arguments.

Even more conveniently, we also learn that the `whoami` and `hostname` commands are the exact same in both Windows and Linux! If that wasn't true and we were doing a black-box pentest, we would need to figure out what OS the web server uses. We could run a command unique to each OS and examine any output to see what fails. Good examples of commands to test with include `powershell` (works on Windows, not Linux) or `ifconfig` (works on Linux, not Windows). There are other ways to find the OS as well; play around and see what works for you.

### How to Chain Commands

Again using our good friend Google, we learn we can chain commands in multiple ways. In Linux, we can use characters like `;` for standard chaining, `&&` if we want the second command to only run if the first worked, or `&` to run the first command in the background while the second runs. In Windows, we usually use `&`. If this was a black-box test, we could iterate through all options to see what selection returns output and determine the OS that way. Or we could just use `&` since it works on both OSes. We do run the risk of having command output returning in odd orders if we have a Linux OS, due to how threads terminate. Personally, as long as we get all required data, I'm okay with that. So let's start with `&`.

## The Attack

Let's take what we learned and putting it into our actual attack string. We know we need:

- A valid IP address to close out the `ping` command built into the code;
- The command `whoami` to find out the user;
- The command `hostname` to find out the hostname; and
- The chaining character `&` to combine the above.

Putting that together, we get our attack string:

```
127.0.0.1 & whoami & hostname
```

Let's try it out.

Success! Line 1 shows the output of `whoami`, "daemon". Line 4 shows the output of `hostname`, which I had specified as "ubuntuwebtest". Our hostname also leaks some useful information, that we're running Ubuntu on our web server. You can test this result as you see fit, but I won't show that here.

If we wanted cleaner output, we could change how we chain the commands:

```
127.0.0.1; echo "\nUser: $(whoami)"; echo "Hostname: $(hostname)"
```

If we want to be thorough, we could verify our answer on the test web server itself:

We did it! Challenge complete.