# PHP Lab 3 Troubleshooting PHP Scripts

The biggest problem my students have had is that they write a PHP script, they open it from their browser, and nothing happens. No errors, nothing!

When a user hits your PHP page, if there are errors in the PHP the user will see a blank page. You don’t want users to see error messages, as they could give valuable information to an attacker. It's good to be able to run the PHP script ahead of time to remove errors before the script is part of your site.

This lab will demonstrate two command line methods to test your php code so you can see errors they generate. Then you will try to display a php page with errors through a web browser, and then track down the errors it generates in the web server log files.

**Please read** the CyberAces module, "PHP Variables and Syntax".

2. Enter the script from the Cyber Aces module PHP Syntax & Variables module, Variable Example, on slide 10. Type the commands one at a time. (You don't need to enter <?php because you're already in PHP.) Make some typos so you see what errors look like.

## Method 1 Command Line

This method runs a PHP script that you've typed into a file. In this case, the command is just:

[john@localhost ~]$ php filename.php.

This is covered in slide 23 of the PHP Flow Control module.

This is the review question from slide 15 of PHP Syntax & Variables. Paste it into a file in your home directory, save it with the extension .php, and run it with php filename.php. Figure out what it will do before you run it.

<?php

$name = "Charles";

$quote = "Aces, $name";

echo "Chuck's dad always used to say \"$quote\" to him as a kid.";

Put an error into the file (removing a ";" will work) and run the file again. Note the error that you get. PHP error messages aren't very enlightening...

## Method 2 error.log file in your web server

1. Copy the file you made in Method 1 (with the error) to the document root of your server, /var/www/html. Attempt to run the script by pointing the VM's browser to:

http://localhost/filename.php

If your script still has the error, your browser won't show anything. To see what happened, look at the error log in /var/log/apache2/error.log.

2. Fix the error in /var/www/html/filename.php and point your browser to it again. The browser should show output this time.

3. To see who has accessed your server, look at /var/log/apache2/access\_log. Can you tell which accesses had errors and which worked? (Hint: The HTTP status code is the second item in each line, after the URI that was accessed.)

## Method 3 Allow PHP error messages

This method should only be used while you are testing a website **before** you make it available on the Internet. It should be turned off once the site is available to the Internet (jargon: In Production.) It gives attackers a huge advantage if you let them see the error messages they generate when they attack your site.

### Step 1: Find the php.ini file

The Ubuntu configuration of Apache/PHP puts the php initialization file, php.ini, in:  
/etc/php/<php version number>/apache2/php.ini.

We need to find what to use for <php version number>, so change directory (cd) to /etc/php and list (ls) the directory. You should see a directory that is just a number. Currently it is 8.1. So the path for php.ini would be:  
/etc/php/8.1/apache2/php.ini

### Step 2: Find the setting display\_errors

Open php.ini in the text editor of your choice. You will need to use sudo, so do one of these from a terminal:  
sudo gedit /etc/php/8.1/apache2/php.ini  
sudo nano /etc/php/8.1/apache2/php.ini  
sudo vim /etc/php/8.1/apache2/php.ini  
Remember that your version number may be different from 8.1.

Find the setting for display\_errors. Note that any line that starts with a semicolon (;) is a comment explaining how the settings work and does not change the setting. Find the line  
display\_errors = Off  
and change it to  
display\_errors = On

The real display\_errors setting is in the section that looks like this:  
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;  
; Error handling and logging ;  
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Save the file. (You did remember to open your editor with sudo, didn't you?)

### Step 3: Restart Apache

Issue the command  
sudo systemctl restart apache2

### Step 4: Test

Use your browser to visit a php file that has an error. You can remove a semicolon from a working file to generate an error if necessary. You should see the error message instead of a blank page.

## Practice

Test the examples in the PHP Variables & Syntax module using the three methods above.

# Hand in

Hand in a screenshot of browsing to your filename.php.