**CEG 4430/6430 Cyber Network Security**

**Project 3**

**(20 Points)**

Through this project, students will learn **unrestricted file uploading vulnerability**, launching attacks against this vulnerability, and perform mitigation (graduate students).

**Submission**

1. Undergraduate:
   1. A report of your answers.
   2. Each team submits one report.
   3. Each team member needs to submit a list of all team members.
2. Graduates:
   1. A report of your answers.
   2. The patched “profile\_image.php” with fixed sanitization function.
   3. Each team submits one report.
   4. Each team member needs to submit a list of all team members.

**Environment**

The environment is simply a PHP-based web environment. You have two options to set it up.

**Option 1**Download and install XAMPP <https://en.wikipedia.org/wiki/XAMPP> OR

**Option 2** We have built the environment inside a virtual machine image. You can access this image by creating a virtual machine. Instructions on using the VCSL for creating the virtual machine are given in this link:  <https://engineering-computer-science.wright.edu/computer-science-and-engineering/virtual-cyber-security-lab> Please follow instructions to login and create an instance by choosing CEG6430/4430.

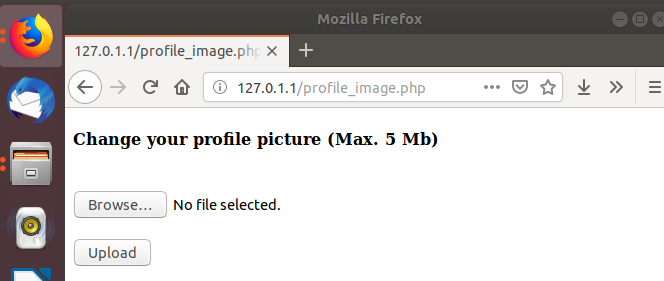
**Option 1** is easy to install. You can get the system up and running in a few minutes. Access virtual machines created using Option 2 from home requires you to use WSU VPN

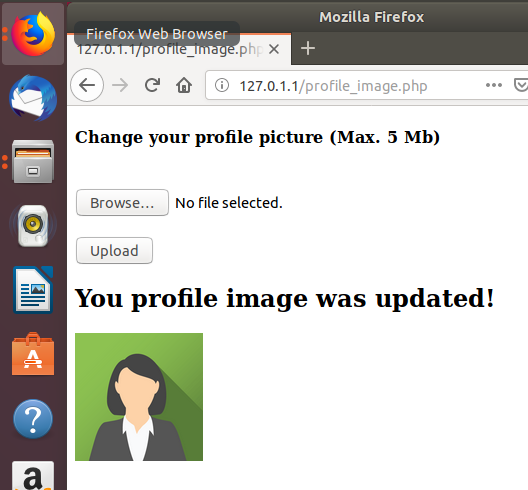
<https://www.wright.edu/information-technology/security/virtual-private-networks-software-overview>.

**After you have installed XAMPP,** copy profile\_image.php into directory c:\xampp\htdocs (assuming you installed XAMP to c:\xampp). Use your web browser, enter URL <http://localhost/profile_image.php>. You will see screens on the next page.

**A Vulnerable PHP Application** (The following steps are for option 2)

1. You will find an PHP application namely “profile\_image.php” in the directory of “/var/www/html/”. You can use text editor to open and edit this file.
2. You can use your browser to visit this application. Specifically, you can use http://127.0.0.1/profile\_image.php to access this file.
3. You should be able to see the interface as follows. This webpage is intended to accept an uploaded image from a user. The following two screenshots show the web interface and the interface after a photo is uploaded.
4. Unfortunately, this PHP script is vulnerable.





**Questions**

Assume that you are an attacker and you have access to the source code of the PHP file (you do not have access to the server).

1. What is the directory (or the path) in the server that stores the uploaded image? (1 points) and Justify your answer by analyzing the source code. (2 points)
2. Suppose the uploaded image has a name called “myphoto.jpg”. How can you directly access this image in your browser? Show the path. (3 points)
3. Can you upload files that are not images to the server and directly get access to uploaded files? Justify your answer using both testing (2 points) and code analysis. (2 points)
4. Create a PHP file and upload it to the server. This PHP file will allow the attacker to remotely execute arbitrary commands in the server. (hint: see “system()” API in PHP). (5 points)
5. Mitigation (5 points)
   1. **Undergraduates**: Briefly but precisely explain how you can expand the sanitization function in the PHP script to mitigate this vulnerability.
   2. **Graduates**: Expand the sanitization function in the PHP script to mitigate this vulnerability. You need to submit the patched “profile\_image.php”.

**References**

It may be helpful for the project to read the following articles.

1. <https://cwe.mitre.org/data/definitions/434.html>
2. PHP $\_FILES: <http://php.net/manual/en/reserved.variables.files.php>