Computer Science & Engineering
College of Engineering
University of Nevada, Reno

Internet Security Final Project

Social Media Phisher

CS 445

Ajani Burgos
Lan Le

Instructor: Iman Vakilinia

May 10th, 2019

Introduction

Phishing is an issue that is of major concern for those involved in internet security. This is due to how maintaining privacy and ensuring that login information is only known by the user is crucial to maintain a good relationship between a website and its users. There are several methods of phishing that rely on either cracks in security or simply tricking the user to believe that a page isn't fraudulent. Such methods of attack that exploit breaches in security are things like session hijacking, trojans, or keyloggers. Tricking the user is another method of stealing information, as the easiest way to hack your way into anything is to fool the user. Our project focuses on implementing one such method of phishing, which is to simply fool the user.

The phisher that we have implemented is designed to act as a general man-in-the-middle attack, with us pulling a website's login page and hosting it on our own network. From there we can steal a user's login info and then redirect them to the original site that they were going on. This script is designed to download the login page and then to upload it to your own server. The primary technology being used is Python for the programming language of the script, as the plethora of libraries available to use to help automate the process makes it ideal, especially when working with subprocessing calls like general bash commands.

The Cloning Process

We designed this script as described earlier to pull a login page from a website. This is done by simply doing a "wget" subprocess call. Before we can wget anything though we'll first need to test if a connection can be established in the first place for consistency's sake. From there the script will terminate and the user may check that a new directory is made based on the website. For our example we will work on spoofing the Twitter login page.

Figure 1: Initial functionality of the script, the clone command steals a login page

Now it may be noted that you may need to cancel the script when you notice it hanging. This is due to how it's attempting to log the wget transaction, but for our purposes we simply care about stealing the index.html file from Twitter.

The Phishing Enabling

Once the website is stolen to your local machine, the user simply uses the script again and it will scp everything for the user. To do this we imported the paramiko and scp modules in order to SSH and transfer everything via secure shell. We will have a list of modules that you may need to install in the next section.

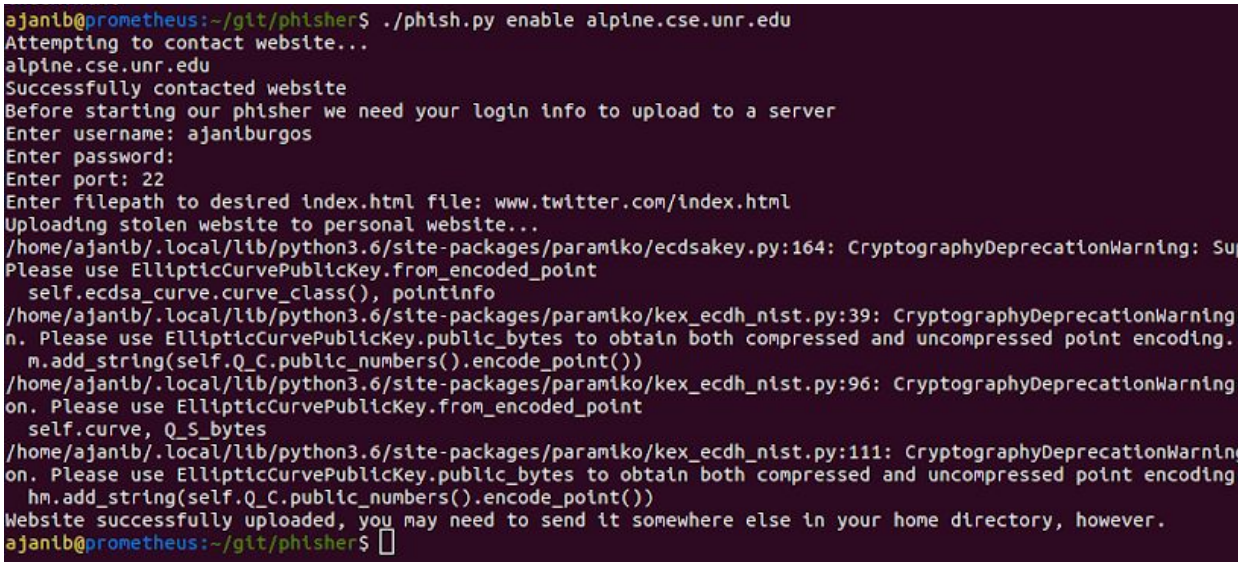

Figure 2: Website uploading functionality, the enable command simply uploads it and prompts the user for any necessary information

Here we see that like in the clone function, the enable function also tests to see if there is a valid connection to a webserver. For our purposes, we simply used alpine.cse.unr.edu, a public-facing server utilized by the CSE Department that also allows users to hosts websites in their public_html directory (hence, why the last message is presented to the user after running).

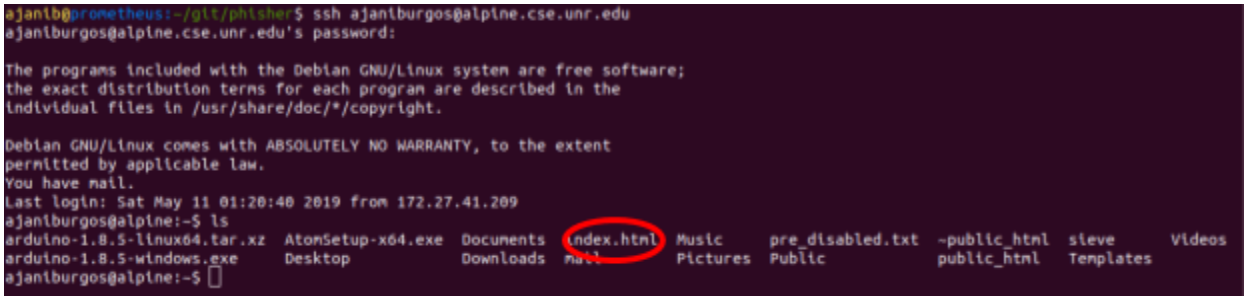

Figure 3: Demonstrating that the index.html file has been moved successfully

Here we see that the index.html file was uploaded successfully to the home directory of the attacker. From this point we can simply move the newly-cloned index.html file to our public_html directory, although the method of hosting may differ between different users based on what webhost they're using.

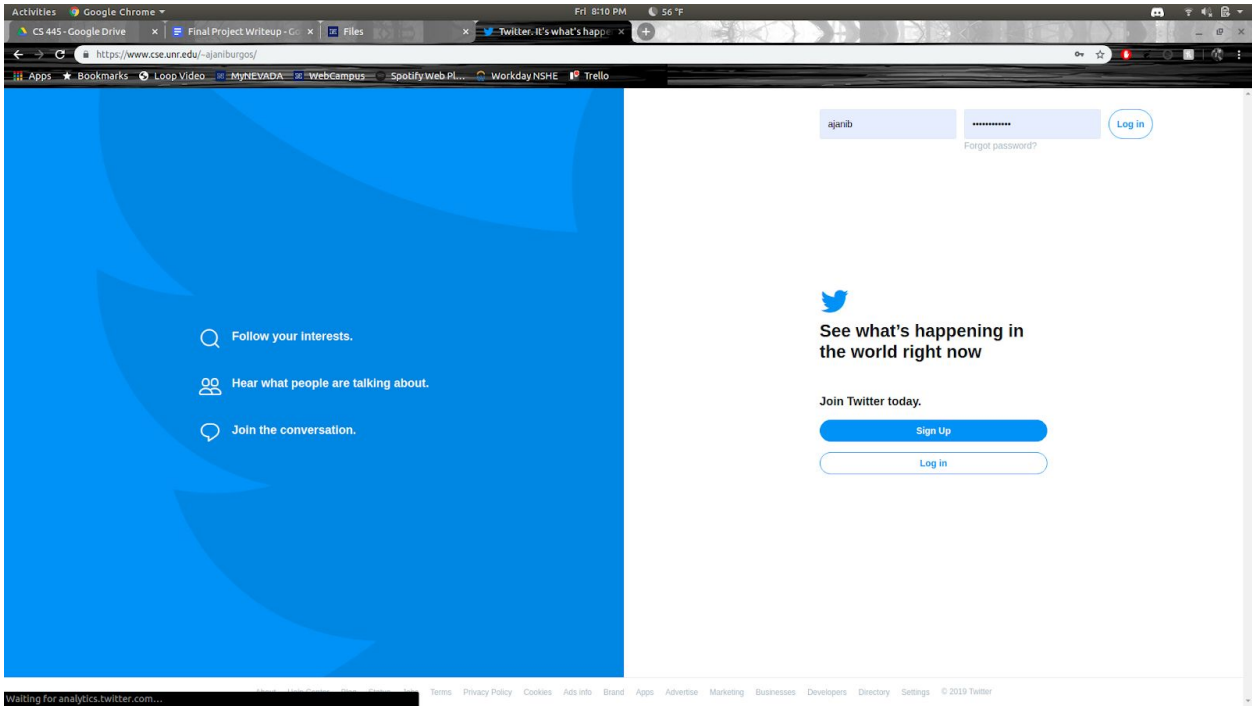

Figure 4: Demonstrating that our spoofed website is now live on our own domain

Here we see that we have Twitter's login page, but you may take notice that the website is now www.cse.unr.edu/~ajaniburgos, meaning that this is definitely our own website, and that it's available on the surface web.

As the attacker, you would ideally change your domain name to more closely match whatever website you're spoofing, such as www.twittr.com or www.twitter.us, but for demonstration purposes we know that we can now trick users with this login page.

Additional Notes & Development Method

As stated earlier, you may need to install certain Python modules in order to run the script. The modules used are:

- sys
- os
- time
- subprocess
- argparse
- scp
- paramiko
- getpass

We developed everything using Python3.7, for the same reasons stated in the Introduction, due to how it allowed for easy manipulation of the terminal and includes a lot of modules that allow us to customize a lot of the script in the first place.

For our method of development, we opted to work on this project using a Paired Programming approach, where we strictly develop the script itself working together at the same time so that both members of the group can understand what's happening in the script. Additionally, this allowed us to identify each others' thought process when going about developing this script and how to best maintain usability of the script overall.

Logging

Ideally, this is meant to pose as a man-in-the-middle attack; however, it's very scary to have users enter in their actual username and password and be taken to the actual twitter site, all while having it be logged. An explanation of what would've happened will be described below, but it was not implemented in order to not steal passwords.

Essentially, what we wanted to have happen is for the user to have entered in their credentials, and the information would be saved to a usernames.txt file that would've been created by the post.php. In the code found in index.html, we simply would replace the highlighted line with "post.php".

```
<form action="https://twitter.com/sessions" class="LoginForm js-front-signin" method="post"
  data-component="login_callout"
  data-element="form"
>
```

Figure 5: By replacing the highlighted line of code in index.html with the php file, we can log information

Then when the user has entered in their credentials, the information would be saved to usernames.txt, all while being taken to the original twitter site and they would carry on unknowingly.

Conclusion

It's genuinely difficult to prepare yourself, and even protect yourself, against phishing attacks. It happens to everyone, and what's even more terrifying is the fact that the replication of these websites and the process of stealing credentials are much easier than expected. Now this poses two challenges for people everywhere: 1. Website owners are unaware that their sites are being used for nefarious acts but also have no way of preventing it from happening, and 2. Website users have to be weary with what they're clicking on, but often times, the fake links are so close to the original that it's difficult to be sure of. Essentially, phishing attacks come in two parts: where users are hooked and then fall for the bait. If a strange email comes in, or a link looks suspicious, then it's best to be cautious of the link you're clicking and what information you enter in.