Austin Marino

Week 6 – Web Recon with Python

CYB 339 – Cyber Operation Tools

Professor Morales

Contents

[Introduction 3](#_Toc1925517)

[Results and Analysis 3](#_Toc1925518)

[Conclusion 7](#_Toc1925519)

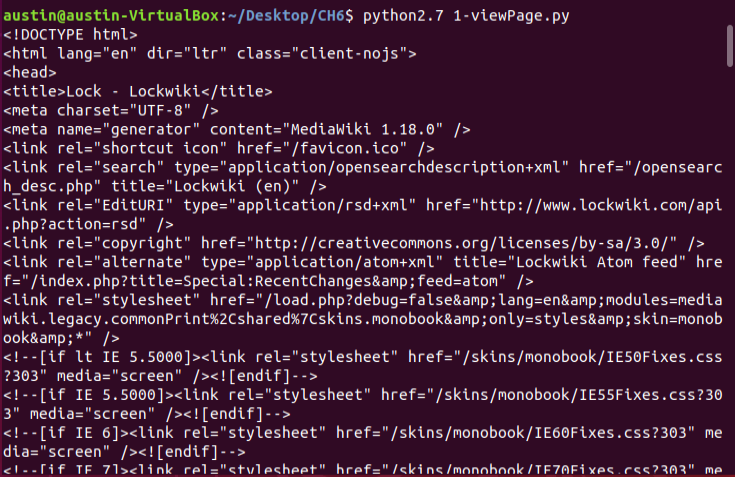
# Introduction

In this lab, the student will be conducting web recon. Web recon is a form of penetration testing that can help maintain confidential information from being publicly accessible through websites. Python scripts allow the user to visually see how unsecure websites are constructed through the HTML code. Another technique is image parsing and link parsing. This technique helps with finding source URL’s embedded into a webpage and also separates images from the URL’s. These images can be stored and further analyzed for more metadata such as geolocation or device information. Locating browser cookies on a user’s system is another great technique for social engineering. Browser cookies are stored on the user’s computer and referenced when revisiting a website. This could potentially be usernames, passwords, links, or other small data remembered by the browser itself. A small python script can pull this type of data directly from a user’s browser. A large portion of web recon is the social engineering aspect. With the right wording and proper execution, employees and users may fall prone to giving up confidential or personal information. Techniques such as these are sent through phishing emails and spam links on social media.

# Results and Analysis

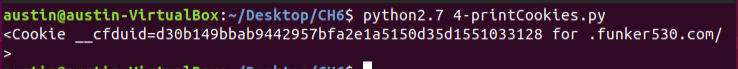
Below (Image 6.1) shows the source code for the website [*www.wikilock.com*](http://www.wikilock.com). This site is unsecure and allows users to pull the HTML source code directly from the web browser. This script did not require any maintenance other than altering the target within the python script to the unsecure website of choice. To execute the script use, *python2.7 1-viewPage.py* and the website HTML source code will be printed to the terminal screen.

(Image 6.1)



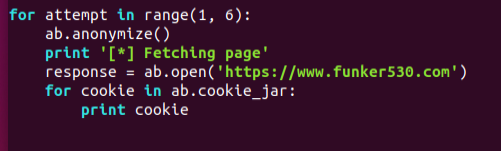
Changing to a more secure website, [*www.funker530.com*](http://www.funker530.com), the user’s computer will now be scanned for browser cookies. This script requires a target ([www.funker530.com](http://www.funker530.com)) for the variable “url”. Once the target has been set, the script can now scan using the “mechanize” library and will locate unique session cookies. (Image 6.2)

(Image 6.2)

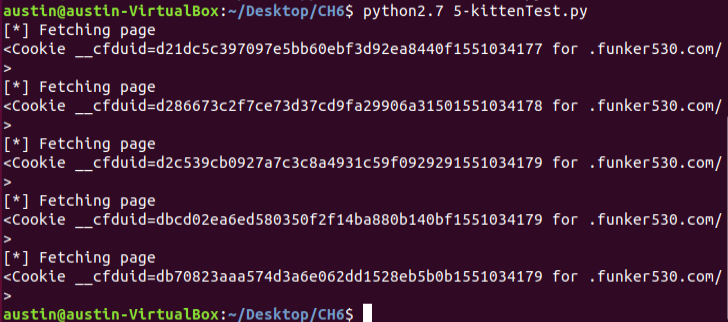


To pull more unique cookies from the same target, a *for* loop is required to locate different cookie ID’s (Image 6.3). From the original source code the range needed to be altered from 1, 5 to 1, 6. In order for 5 unique cookie ID’s, the loop needs to run 5 times. The previous code only allowed 4 iterations. While using the *range* function, iterations end once the number is reached and the loop terminates. This is known in programming as an “off by one error”. Further below (Image 6.4) shows the correct output of the 5 unique cookies that were stored from the target website [*www.funker530.com*](http://www.funker530.com). To run this script use the command *python2.7 5-kittenTest.py* (Image 6.4).

(Image 6.3)

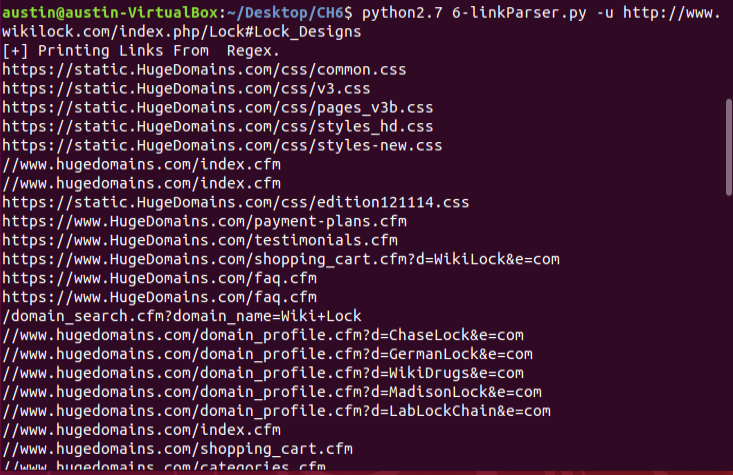


(Image 6.4)



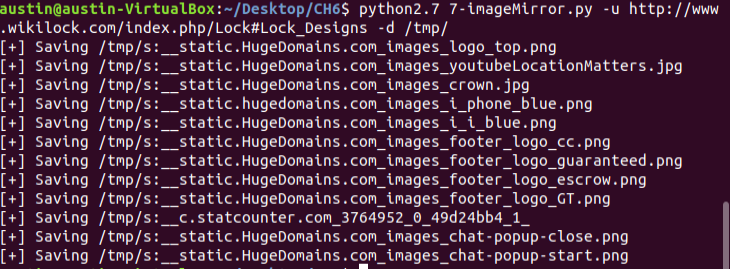
Going back to the unsecure website [*www.lockwiki.com*](http://www.lockwiki.com)*/index.php/Lock#Lock\_Designs,* the user will now be parsing URL’s from the webpage into a tabular format (Image 6.5). Parsing is a great technique to find hidden links within a webpage that may lead to browser vulnerabilities. It is quite possible that an admin may forget about these links and can allows attackers to have backdoor entrance through an un-managed link in the site. To run this script use *python2.7 6-linkParser.py -u* [*http://www.lockwiki.com/index.php/Lock#Lock\_Designs*](http://www.lockwiki.com/index.php/Lock#Lock_Designs)

(Image 6.5)



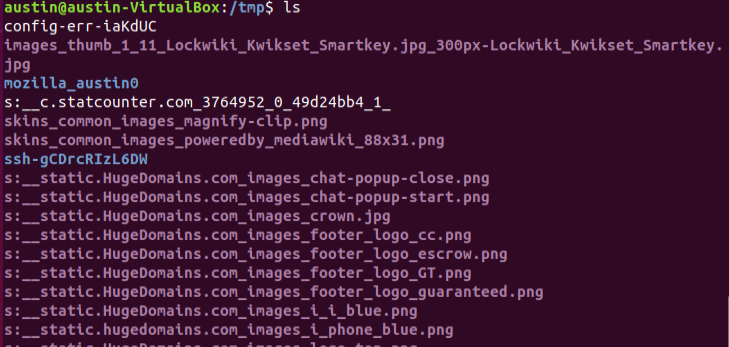
Link parsing is not the only method to seek hidden information, but image parsing is another technique that can be helpful in web recon. Saving parsed images to a user’s /tmp directory (Image 6.7) will allow them to further investigate possible data hidden within images. Earlier in the semester, geolocation from images and IP addresses was a major topic discussed. Images taken directly from a site can possibly be scrubbed for metadata allowing an attacker to discover sensitive physical locations. To parse images from the target site (Image 6.6) use the command *python2.7 7-imageMirror.py -u* [*http://www.lockwiki.com/index.php/Lock#Lock\_Designs*](http://www.lockwiki.com/index.php/Lock#Lock_Designs)

(Image 6.6)



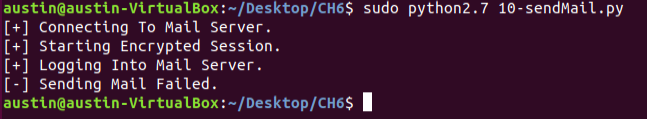
The parsed images are exported to the /tmp directory for further analysis. (Image 6.7)

(Image 6.7)

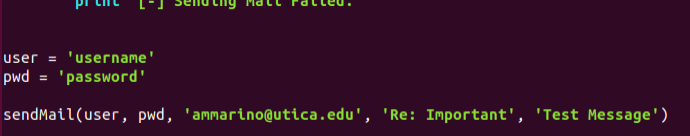


Automating malicious emails and spam is a useful technique for social engineering. While observing phishing emails, some employees may comply with the fake email and release confidential information such as user logins or social security numbers. Information such as this can allow a user to gain easy access to systems and cause an entire network to become compromised. The *sendMail.py* and *sendSpam.py* scripts would not properly execute so further analysis of the code will be discussed. The *sendMail.py* script would not throw any exception errors other than “Sending Mail Failed” (Image 6.8). While analyzing the code and reading the text (Image 6.9), the user altered the target email to a personal email and was still receiving the error. This issue could be because the gmail server does not allow traffic from non-authorized users. As for the *sendSpam.py* script (Image 6.10), the user does not utilize twitter or any social media platforms. While utilizing the text book example, numerous errors were thrown because it was not a “real” account that was accessible. Reviewing the code, there are pros and cons to this attack. The *if* statements that find a user’s relatable interests are a great social engineering tactic that may interest a victim and believe the attack. The script locates interests, locations, and links located on the victim’s page, making the attacker a relatable, trustworthy person. Leading up to the attacking link, the victim will be “fluffed” up to the attacker and may click this link, providing personal information. On most social media platforms people click links without reading the actual URL. A con to this example is the actual target link. An attacker should not make their malicious link obvious. Seeing the link “evil.tgt” is not a very trustworthy name and more than likely will be useless. Using something more discrete would be more effective in an attack like this.

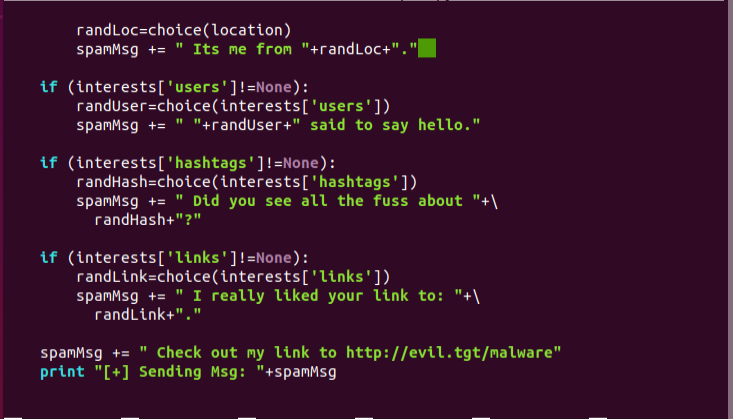
(Image 6.8)



(Image 6.9)



(Image 6.10)



# Conclusion

During this lab procedure, I found it interesting that the HTML source code could be directly taken from a non-secure website. This is an easy way for attackers to mimic websites and steal login credentials or other personal information. Me personally, I do not affiliate myself with unsecure websites unless I have to. Not utilizing HTTPS on a website is just asking to be hit by attackers. When running the *sendMail.py* script, I became very confused as to what was going on. Researching online, I found that the issue may lie on the server’s side of things. Using gmail as an example, may automatically filter out external email addresses for spam and un-verified email addresses. I tried using my own personal login credentials and email address to get the mail to send but could not properly get the mail to go through to the target email. As for the *sendSpam.py* script, I do not have a Twitter account and was unsure of how to conduct this without an account. Attempting to utilize the book example, I was not able to produce results from an inbox because I did not have access to the account. That being said, I explained my opinion on whether I believe the attack would properly work in a live scenario. Overall, this lab was very interesting and informative. My favorite section was the parsing of links and images. Being able to see every link embedded on the webpage could bring hidden links to the surface that may no longer be managed. This being a key asset to gaining backdoor entry or finding yourself somewhere you should not be. Seeing information provided through these scripts shows directly how web recon plays a major roll in penetration testing and can be a key tool to conducting a social engineering attack on employees.