As a starting point for the CSC 333 project, I decided to utilize the Hydra tool from the available resources on the Kali website. Hydra is a “login cracker” that may take in multiple arguments before processing data to find a successful username and password combination that will allow access to a specific account for a particular service (*Hydra*, 2022). Essentially, Hydra needs a login username or a list of usernames followed by a password or a list of passwords, the service to attack, like “ssh, http, or rpcbind,” and the IP address with an optional port number (Tenbergen, 2022). Hydra then tries out every given password for each specified user until it reaches the end of both lists while keeping track of all valid username and password pairs that correlate with the indicated service, IP address, and port number. Hydra displays any successful login credentials at the end of the test.

As my first experience with Hydra, I entered a command to ask Hydra to try and find the login credentials to my Kali account when given my username, a list of passwords, the service to attack, and the IP address of eth1. The command to perform this task is hydra -l jcountry -P passes.txt ssh://192.168.56.113, where -l represents a single incoming username, -P indicates an incoming file of passwords, and passes.txt is a user-created list that contains three incorrect passwords and the actual password. Hydra successfully discovered my login and password credentials and displayed a summary of the host connection along with my username and valid password. When no successful login and password pairs are detected, Hydra states that none of the known passwords grant access to the service with any of the known usernames.

Hydra will be convenient during the exploitation phase of penetration testing because it must be known that a service has any login vulnerabilities beforehand, or Hydra will have virtually no value. It is best to implement Hydra on services that enforce few or no requirements when allowing users to create their passwords, including minimum password lengths, special

characters, printable characters, numbers, and capital letters. Additionally, Hydra can filter out passwords with all these requirements when given a list of passwords through the pw-inspector command, which removes passwords from a given list that do not meet the required criteria and returns the result to a new list. I tried out pw-inspector by writing a command to filter out a list of passwords that were not between six and ten characters long, which is done by writing pw-inspector -i passwords.txt -o newpasswords.txt -m 6 -M 10, where passwords.txt is my input list that had six passwords where only four met the criteria and newpasswords.txt is my output list, which successfully included the four passwords that met the length requirement.

Hydra is also worthwhile when some services set passwords by default or if the user enters easily guessable passwords. In this case, it may be advantageous to use dpl4hydra, which uses a "local default password list" of over eleven thousand simple or default passwords to find a potential match for each given user (*Hydra*, 2022). However, my computer would take nearly two hours to search through the entire list with the indicated speed of ninety-five attempts per minute, so I did not fully utilize this feature. As I continue to grasp more knowledge in ethical hacking techniques, I hope Hydra will help me understand the simplicity of how attackers may take advantage of password vulnerabilities due to password criteria that websites are not enforcing or that users may not be utilizing that would provide more security to their accounts.

References

OffSec Services Limited. (2022, November 16). *hydra | Kali Linux Tools*. Kali.org. https://www.kali.org/tools/hydra/

Tenbergen, B. (2022, June 4). *Getting Started with Penetration Testing* [Tutorial]. Brightspace. https://mylearning.suny.edu/d2l/le/content/616330/viewContent/17823004/View