### Question 1

Prompt: Calculate and explain the NIST Entropy score for a password policy that requires a 25 character password consisting entirely of lowercase letters. Assuming that "the entropy calculation is equivalent to a uniformly distributed randomly generated key of length H(x)", what is the estimated max number of guesses required to determine the password (i.e., the probability of attack success is 100%)? What is the average number of guesses required (i.e., probability of attack success is 50%)? Finally, provide an example password following this policy that might be easy for an attacker to guess.

The NIST Entropy score introduced in the original SP800-63 document attempts to define random variables in a given password by specifying how they are created through the use of common password creation policies. Each variable in a password is given a entropy score, and the sum of all the scores are added up to create a final entropy total for the entire system. The entropy score for each variable assigned is as follows:

The first character is 4 bits of entropy

The 2nd to 7th characters are 2 bits per character

The 9th to 20th characters are 1.5 bits per character

Any character from 21 on is 1 bit per character

For our password policy example, we given a length 25 characters consiting of only lowercase letters. Therefore we have 4 bits for our first character, 14 bits for characters 2 through 8, 18 bits for characters 9 through 20, and 5 bits for chacters 21 through 25, giving us a total entropy score of 41

Knowing this score, we can estimate the max number of guesses needed to determine a password with 100% probability.

That’s done with this formula y = x / 2^H(x)

In this case y is 1 for 100% probablity.

That gives us 1 = x/2^41 which simplifies to x = 2^41 giving us approximately 2.2 trillion guesses for 100% probability.

For the case of 50% probabilty,

We use the same formula but use .5 to repsent y instead of 1.

This simplies our equation to x = 2^40 which gives us appxoimately 1.1 trillion guesses for a 50% probability.

An example of a password that matches the given policy that would be easy for an attack to guess would be sequential letters or numbers like abcdefghijklmnopqrstuvwxy. The NIST score does not account for passwords that are easily guessed, which is a dangerous flaw that can lead to a false sense of security, and bad password creation policies that diallows many passwors that would be secure against a deterimned attacker.

### Question 2

Prompt: What is my passw0rd? Explain how you cracked it. Here's a couple of things you need to know. First, I am using Ubuntu 16.04. Second, the command `cat /etc/shadow` prints out `robert:$6$8jI7K9WV$kKKxHttoYC9pd000pmiHykjwZzwmCAbWzkn5Adg3jfDQT14HapiNjot3OOabfVEC/SnQn0UlpuwW6xpdV8Pp8.:18872:0:99999:7:::`. Finally, Kali Linux and Johnny might be useful.

The information provided to help crack the password was printed from the etc/shadow file on Ubuntu Linux. Linux stores user profile passwords as hashes instead of encrypting them, and this file contains all usernames and their associated password hashes. Only users with root access can access and write to this file.

Looking at the provided hash, we can see he username robert, then a $6$ denotes the type of algorithm that handled the password hashing. In this case its SHA-512. The next characters between dollar signs $8jI7K9WV$ denotes the SALT added to the has. Salt is just random string which is generated when you create password which helps protect against rainbow table attacks. The next longstring of characters is the actual password hash, followed by the password change data denoted by the colons.

Using this known information, I can do something called a dictionary attack with the help of some tools mainly virtualbox, kali linux, and an jack the ripper.

Kali linux comes with a helpful dictionary wordlist installed called rockyou which I used in conjunction with the application John The Ripper. John the ripper works by taking in a dictionary wordlist as input, and hashing it in the same format as the password being examined and comparing the output to the hashed string.

After using the running the application I able to sucessfully crack the password for the given username robert which is passw0rd.

To crack the password with the information provided to me in question 2, I sought out the use of 3 tools, VirtualBox, Kali Linux and John the Ripper. I used virtual box to install Kali Linux, which is a security focused distribution of Linux that comes preinstalled with an abundance of computer security applications. The one I used to crack the password, John the Ripper, works by using a given HASH as input, and attempting to match it to a given password in a dictionary word list.

Then, knowing every version of kali Linux comes with a great wordlist called rockyou preinstalled I opened up the terminal, and performed a locate command to find the file. After getting the file location, I did a gunzip command to decompress the file and get access to the rockyou.txt file inside. With access to the wordlist, I was ready to use John the ripper.

FInally using the John command, I pass in the wordlist file location, associate it with the hash.txt file and in 1 iteration the program was able to crack the password which was passw0rd with a 0, the username robert.