Question 1

|  |
| --- |
| Using command in linux bash: **cat text|grep -rin "And .\*" |grep -rin " it$"**  The answer is: **And give't Iago: what he will do with it** in line **2541**  And the passphrase is: **csf2021\_{anyone-thespian-gripsack}** |

Question 2

|  |
| --- |
| Using command: sort here |uniq -c |grep 14  And the answer is: **csf2021\_{kilometer-skimpily-vertical}** |

Question 3

|  |
| --- |
| Using command:  sha256sum \* | grep "a92536e3c31979736460be6e6729147f974411ef193629999b022b96  f5682450"  And the target file is: **file00066**, the content is:  **zgHN0Dz5ujYlzkIyyU7mOT3RA7fpR5LRxUR7KOQ85fW5rw1HYtKQZUU20003tuX0** |

Question 4

|  |
| --- |
| The SHA256 value is:  **1c3a1c74baafc1d76b9ec68045ded66f281553c215b4c9716988a77bc66de9a3**  Using command: **cat word.txt | tr [aeio] [4310] > password.txt** |

Question 5

|  |
| --- |
| Using script:  cat password.txt | while read pass;  do result=$(gpg --output 1.txt -d --batch --passphrase $pass secret.txt.gpg 2>/dev/null);  if [ $? -eq 0 ];  then echo $pass;  fi;  done  The password is **pr0b4bl3**  And the content is: **csf2021\_{punch-embassy-unknowing}** |

Question 6

|  |
| --- |
| We could design a program to decode the cipher with cyber.py    The secret is **csf2021\_{clip-material-passenger}** |

Question 7

|  |
| --- |
| Using command: find Q07/ -size 47c  The file is **Q07/folder4/folder2/folder3/file1** |

Question 8

|  |
| --- |
| bash > file secret  secret: bzip2 compressed data, block size = 900k  So, this is a compressed file, unzip it, the secret is **csf2021\_{turbine-ecology-hunger}** |

Question 9

|  |
| --- |
| Using command: **strings a.out**  We could find these information:  \_\_gmon\_start\_\_  GLIBC\_2.2.5  UH-H  UH-H  []A\A]A^A\_  What is the secret?  **csf2021\_{refurbish-nativity-recycling}**  congrats!!  sorry, try again :(  ;\*3$"  GCC: (GNU) 4.8.5 20150623 (Red Hat 4.8.5-39)  GCC: (GNU) 4.8.5 20150623 (Red Hat 4.8.5-44)  After testing, the secret is **csf2021\_{refurbish-nativity-recycling}** |

Question 10

|  |
| --- |
| The content is Y3NmMjAyMV97bWF0dXJlbHktc3BlY2llcy1iYXJsZXktZGVwbGV0aW9ufQo=  Pay attention to the ‘=’at the end, the cipher may be generated by Base64.  Using command: **base64 -d secret.txt**  **csf2021\_{maturely-species-barley-depletion}** |

Part 2

Question 11

|  |
| --- |
| The encrypt algorithm generates random number using seed provided by user (in this problem it is 2021), and then doing XOR operator with character in text.  Fortunately, if we know the seed for random number generator and the XOR operator is reversible, so the encrypt process could be decrypt process.  bash> python3 enc.py secret2021.enc 2021  bash> cat secret2021.enc.enc    Obviously, it is a bad encryption. We could crack it using brute force to test possible seeds. And it could be much easier using plain-text attack. For example, so specific format file such as .java or .png, the bytes at head we have already known, and we know the result after XOR operation, it reveals the pattern of the encryption key.  However, the key space is large enough, because Python has no limitation on the range of integer, it means you can choose any number without limitation as key. |

Question 12

|  |
| --- |
| According to the mechanism of RSA:  We could calculate that the plaintext is:  10473389353927511439721808476233847116240945871752629928205285685373  In text:  **csf2021\_{cautious-unscrew-x}** |

Question 13

|  |
| --- |
| Bash >file secret\_16bit\_1024x768\_ecb.bmp.enc  secret\_16bit\_1024x768\_ecb.bmp.enc: openssl enc'd data with salted password  It provides the information of this file.  Create a blank bmp image, and compared with file provided,      The difference is head bytes. So, we can mix these two files and generated a new bmp file:  bash> dd if=test.bmp count=54 ibs=1 >> 1.bmp  54+0 records in  0+1 records out  54 bytes transferred in 0.000131 secs (411804 bytes/sec)  bash> dd if=secret\_16bit\_1024x768\_ecb.bmp.enc skip=54 ibs=1 >> 2.bmp  393306+0 records in  768+1 records out  393306 bytes transferred in 0.348187 secs (1129583 bytes/sec)  bash> cat 2.bmp >> 1.bmp  背景图案  描述已自动生成  The message hidden in it is **CYBER** |

Question 14

|  |
| --- |
| This cipher has structure for normal text, we could hypothesis that it is simple substitution cipher such as Caesar cipher. However, after trying all 25 keys for Caesar cipher, there is no result look like a normal text.  We could use online tools to analysis frequency.  The plaintext:  IT WAS THE BEST OF TIMES, IT WAS THE WORST OF TIMES, IT WAS THE AGE OF WISDOM, IT WAS THE AGE OF FOOLISHNESS, IT WAS THE EPOCH OF BELIEF, IT WAS THE EPOCH OF INCREDULITY, IT WAS THE SEASON OF LIGHT, IT WAS THE SEASON OF DARKNESS, IT WAS THE SPRING OF HOPE, IT WAS THE WINTER OF DESPAIR, WE HAD EVERYTHING BEFORE US, WE HAD NOTHING BEFORE US, WE WERE ALL GOING DIRECT TO HEAVEN, WE WERE ALL GOING DIRECT THE OTHER WAY - IN SHORT, THE PERIOD WAS SO FAR LIKE THE PRESENT PERIOD, THAT SOME OF ITS NOISIEST AUTHORITIES INSISTED ON ITS BEING RECEIVED, FOR GOOD OR FOR EVIL, IN THE SUPERLATIVE DEGREE OF COMPARISON ONLY.  The encryption key is:  **A B C D E F G H I J K L M N O P Q R S T U V W X Y Z**  **A Z E R T Y U I O N Q S D F G H J K L M W X C P B V**  The dencryption key is:  **A B C D E F G H I J K L M N O P Q R S T U V W X Y Z**  **A Y W M C N O P H Q R S T J I X K D L E G Z U V F B** |

Question 15

|  |
| --- |
| In shadow file, the information of user called “yoda”:  yoda:$1$1V8SfbzZ$No6X4H.b1.lqGRv2yLYNv0:18330:0:99999:7:::  It uses MD5 for encrypting, the salt is: 1V8SfbzZ  Using command:  hashcat -m 500 -a 0 -o found.txt \$1\$1V8SfbzZ\$No6X4H.b1.lqGRv2yLYNv0 rockyou.txt  -m 500: Password mode, UNIX MD5  -a 0: Using dictionary  The password is: **spiderman1** |