Q1 Teamname

0 Points

Cryvengers

Q2 Commands

5 Points

List the commands used in the game to reach the ciphertext.

go,wave,dive,go,read

Q3 Analysis

50 Points

Give a detailed description of the cryptanalysis used to figure out the password. (Explain in less than 100 lines and use Latex wherever required. If your solution is not readable, you will lose marks. If necessary, the file upload option in this question must be used TO SHARE IMAGES ONLY.)

After we have reached the ciphertext by using above commands, we have found the encrypted password (i.e p="$gsjgktipfolqgfmqiffffiljjgujfklhn$") by passing the text 'password' as input at the screen.
After passing several inputs and observing corresponding outputs, we have observed that the letters at odd positions in the 'output' have

range from $'f'$ to $'m'$(8 letters), so the letter at odd positions requires 3 bits to represent, and letters at even places have range from $'f'$ to $'u'$ (16 letters), so the letter at even positions requires 4 bits to represent. Similarly for possible 'input' the range is from $'f'$ to $'o'$ for letters at odd positions and $'f'$ to $'z'$ for letters at even positions. But then the range is chopped off to that of the output range, as it easier to represent the earlier as a F_{128} element. And this presumption is seemed to be valid at the end.

So i^{th} byte of output (which belongs to F_{128}) corresponds to $a_{2i-1}a_{2i}$ (overall 7 bits) where $a_i(i \geq 0)$ is i^{th} letter in output. So our encrypted password(32 letters) is of 2 block length.

For several inputs we have given we came to know that changing i^{th} input byte changes only the part of output from the i^{th} byte.

Eg:

"$ffghghjkjkllmlm$" gives "$fflugnlihjfshmhq$",
"$ffghghjkjkllmkl$" gives "$fflugnlihjfshmfi$",
"$ffghghjkjkllmukl$" gives "$fflugnlihjfsmnkm$".

First we have considered the first block:

Now we have looped over 128 possibilities ($'\alpha\beta' \mid \alpha \in 'f'$ to $'m', \beta \in 'f'$ to $'u'$) and found the one that gives 1st byte of output same as 1st byte of encrypted password. (say $'\alpha_1\beta_1'$)

Now we know the first byte and now we have looped over other 128 possibilities(i.e $'\alpha_1\beta_1ff'$ to $'\alpha_1\beta_1mu'$) and found the one that gives first 2 bytes of output same as 1st 2 bytes of encrypted password.(say $'\alpha_1\beta_1\alpha_2\beta_2'$)

Now we know the first 2 bytes and now we have checked over other 128 possibilities(i.e $'\alpha_1\beta_1\alpha_2\beta_2ff'$ to $'\alpha_1\beta_1\alpha_2\beta_2mu'$) and found the one that gives first 3 bytes of output same as 1st 3 bytes of encrypted password.

We have continued this process until we have got 8 bytes.

This has given us the first block of the password and similar approach is

used to decrypt the second block.
Finally we have got "$mimnmgmkmpmflulq$" as the decrypted 1st block and "$lglhifififififif$" as the decrypted 2nd block.
So overall is "$mimnmgmkmpmflulqlglhifififififif$" which when converted to ASCII representation gives "$sxquzpokab$"(ignoring 0's)
(eg: 'mi', 'm' $\equiv$ 111 (7, $'m' - 'f'$) , 'i' $\equiv$ 0011 (3, $'i' - 'f'$) stacking both gives 01110011 $\equiv$ 115 $\equiv$'s' in ascii representation)
Thus "$sxquzpokab$" is the encrypted password for this level.

In the below codes $ks1.py$ takes known bytes(say n bytes known) in first block from $ipassword.txt$(none if 0 bytes are known) and loops over 128 possibilities. We stored output of code in out1.txt. Similarly for 2nd block in out2.txt . Now we have filtered the out1.txt and out2.txt to only encrypted outputs for given 128 inputs in out_filter1.txt and out_filter2.txt using script1_1.py and script1_2.py respectively. In these out_filter1.txt and out_filter2.txt, we have found corresponding next byte using script2.py and stored them along with n known bytes in $i_password1.txt$ and $ipassword2.txt$.
We have done it for 8 times(from n=0 to n=7) and finally got 8 bytes in each block and then we have converted them to ASCII.

No files uploaded

Q4 Password

5 Points

What was the final command used to clear this level?

sxquzpokab

Q5 Codes

0 Points

ks1.py Download

```
1  import threading, paramiko
2  import sys
3  import os
4
5  class ssh:
6      shell = None
7      client = None
8      transport = None
9
10     def __init__(self, address, username, password):
11         print("Connecting to server on ip", str(address) +
   ".")
12         self.client = paramiko.client.SSHClient()
13
   self.client.set_missing_host_key_policy(paramiko.client.AutoAd
14         self.client.connect(address, username=username,
   password=password, look_for_keys=False)
15         self.transport = paramiko.Transport((address, 22))
16         self.transport.connect(username=username,
   password=password)
17
18         thread = threading.Thread(target=self.process)
19         thread.daemon = True
20         thread.start()
21
22     def closeConnection(self):
23         if(self.client != None):
24             self.client.close()
25             self.transport.close()
26
27     def openShell(self):
28         self.shell = self.client.invoke_shell()
29
```

```
    def sendShell(self, command):
        if(self.shell):
            self.shell.send(command + "\n")
        else:
            print("Shell not opened.")

    def process(self):
        global connection
        while True:
            # Print data when available
            if self.shell != None and
self.shell.recv_ready():
                alldata = self.shell.recv(1024)
                while self.shell.recv_ready():
                    alldata += self.shell.recv(1024)
                strdata = str(alldata, "utf8")
                strdata.replace('\r', '')
                print(strdata, end = "")
                if(strdata.endswith("$ ")):
                    print("\n$ ", end = "")

sshUsername = "student"
sshPassword = "caesar"
sshServer = "65.0.124.36"
k=int(input())

connection = ssh(sshServer, sshUsername, sshPassword)
connection.openShell()
connection.sendShell('Cryvengers')
os.system("sleep 1")
connection.sendShell('Cryvengers@pkm')
os.system("sleep 1")
connection.sendShell('5')
os.system("sleep 1")
connection.sendShell('go')
os.system("sleep 1")
connection.sendShell('wave')
os.system("sleep 1")
```

```
connection.sendShell('dive')
os.system("sleep 1")
connection.sendShell('go')
os.system("sleep 1")
connection.sendShell('read')
os.system("sleep 0.5")
text = "gsjgktipfolqgfmq"
pvals=[]
for i in range(0,8):
    for j in range(0,16):
        pvals.append(chr(i+ord("f"))+chr(j+ord("f")))
if(k!=0):
    fptr1=open("ipassword1.txt")
    p1=fptr1.readline()
else:
    p1=""

for i in pvals:
    inp=p1+i
    connection.sendShell(inp)
    os.system("sleep 0.5")
    connection.sendShell('c')
    os.system("sleep 0.5")
#while True:
 #   command = input('$ ')
 #   if command.startswith(" "):
 #       command = command[1:]
 #   if command == 'exit':
 #       connection.sendShell('quit')
 #       break
 #   connection.sendShell(command)
connection.closeConnection()
```

ks2.py Download

```
import threading, paramiko
import sys
```

```
import os

class ssh:
    shell = None
    client = None
    transport = None

    def __init__(self, address, username, password):
        print("Connecting to server on ip", str(address) +
"." )
        self.client = paramiko.client.SSHClient()

self.client.set_missing_host_key_policy(paramiko.client.AutoAd
        self.client.connect(address, username=username,
password=password, look_for_keys=False)
        self.transport = paramiko.Transport((address, 22))
        self.transport.connect(username=username,
password=password)

        thread = threading.Thread(target=self.process)
        thread.daemon = True
        thread.start()

    def closeConnection(self):
        if(self.client != None):
            self.client.close()
            self.transport.close()

    def openShell(self):
        self.shell = self.client.invoke_shell()

    def sendShell(self, command):
        if(self.shell):
            self.shell.send(command + "\n")
        else:
            print("Shell not opened.")

    def process(self):
```

```
        global connection
        while True:
            # Print data when available
            if self.shell != None and
self.shell.recv_ready():
                alldata = self.shell.recv(1024)
                while self.shell.recv_ready():
                    alldata += self.shell.recv(1024)
                strdata = str(alldata, "utf8")
                strdata.replace('\r', '')
                print(strdata, end = "")
                if(strdata.endswith("$ ")):
                    print("\n$ ", end = "")

sshUsername = "student"
sshPassword = "caesar"
sshServer = "65.0.124.36"
k=int(input())

connection = ssh(sshServer, sshUsername, sshPassword)
connection.openShell()
connection.sendShell('Cryvengers')
os.system("sleep 1")
connection.sendShell('Cryvengers@pkm')
os.system("sleep 1")
connection.sendShell('5')
os.system("sleep 1")
connection.sendShell('go')
os.system("sleep 1")
connection.sendShell('wave')
os.system("sleep 1")
connection.sendShell('dive')
os.system("sleep 1")
connection.sendShell('go')
os.system("sleep 1")
connection.sendShell('read')
os.system("sleep 0.5")
text = "iffflijquifklhn"
```

```
pvals=[]
for i in range(0,8):
    for j in range(0,16):
        pvals.append(chr(i+ord("f"))+chr(j+ord("f")))
if(k!=0):
    fptr2=open("ipassword2.txt")
    p2=fptr2.readline()
else:
    p2=""

for i in pvals:
    inp=p2+i
    connection.sendShell(inp)
    os.system("sleep 0.5")
    connection.sendShell('c')
    os.system("sleep 0.5")
#while True:
 #   command = input('$ ')
 #   if command.startswith(" "):
 #       command = command[1:]
 #   if command == 'exit':
 #       connection.sendShell('quit')
 #       break
 #   connection.sendShell(command)
connection.closeConnection()
```

script1_1.py Download

```
f1=open("out1.txt","r+")
i=0
while True:
    k=f1.readline()
    k=k.lstrip()
    k=k.rstrip()
    if(len(k)==16 and k[0]!='>'):
        print(k)
        i=i+1
```

```
    if(i==128):
        break
f1.close()
```

script1_2.py Download

```
f2=open("out2.txt","r+")
i=0
while True:
    k=f2.readline()
    k=k.lstrip()
    k=k.rstrip()
    if(len(k)==16 and (k[0])!=">"):
        print(k)
        i=i+1
    if(i==128):
        break
f2.close()
```

script2.py Download

```
text1= "gsjgktipfolqgfmq"
text2= "ifffiljjgujfklhn"
pvals=[]
for i in range(0,8):
    for j in range(0,16):
        pvals.append(chr(i+ord("f"))+chr(j+ord("f")))
f1=open("out_filter1.txt","r+")
f2=open("out_filter2.txt","r+")
fptr1=open("ipassword1.txt","r+")
fptr2=open("ipassword2.txt","r+")
k=int(input("Give number of known bytes:"))#represents number
of bytes decrypted
if(k!=0):
    p1=fptr1.readline()
    p2=fptr2.readline()
else:
    p1=""
```

```
    p2=""
fptr1.close()
fptr2.close()
fptr1=open("ipassword1.txt","w+")
fptr2=open("ipassword2.txt","w+")
i=0
while True:
    q=f1.readline()
    if(q[0:(2*k+2)]==text1[0:(2*k+2)]):
        break
    i=i+1
w1=p1+pvals[i]
fptr1.write(w1)
i=0
while True:
    q=f2.readline()
    if(q[0:(2*k+2)]==text2[0:(2*k+2)]):
        break
    i=i+1
w2=p2+pvals[i]
fptr2.write(w2)
```

Assignment 5

GRADED

GROUP

KARPURAPU MANOJ KUMAR
VANDANAPU PRANAY
VANGALA KRISHNA SAI

View or edit group

TOTAL POINTS

60 / 60 pts

QUESTION 1

Teamname — **0** / 0 pts

QUESTION 2

Commands — **5** / 5 pts

QUESTION 3

Analysis — **50** / 50 pts

QUESTION 4

Password — **5** / 5 pts

QUESTION 5

Codes — **0** / 0 pts