Q1 Teamname
0 Points

Stuxnet

Q2 Commands
10 Points

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

go, dive , dive, back, pull, back, back, go, wave, back, back, thrnxxtzy, read, 3608528850368400786036725, c, read

Q3 Cryptosystem
5 Points

What cryptosystem was used at this level? Please be precise.

6 round DES

Q4 Analysis
80 Points

Knowing which cryptosystem has been used at this level, give a detailed description of the cryptanalysis used to figure out the password. (Explain in less than 150 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.)

At first we tried to use DES 4 rounds, but we got no success with that. So we shifted to 6 round DES as the spirit has hinted us that it may be 4 round or 6 round. The first hint that we saw was "There was something funny about how the text appears, two letters for one byte or something like that". We know that the block size of DES is 8 Bytes, so we will have total 16 letters in a block. By giving random inputs we have observed a very important point that the 16 letters are always within the interval [f,u]. So we assumed the inputs also lie within this interval of [f,u] and moreover 16 letters could be represented by 4 bits, so we numbered [f,u] from 0 to 15. The differential characteristic of 6-round DES ‘405c0000 04000000’ was used. After 4 rounds we got a differential ‘00540000 04000000’ with probability 0.00038. So it is clear that we need approximately 1 lakh plain text cipher text pair. At first we generated one lakh random inputs using 'RandomString.cpp' and then generated 2 lakh input pairs from 'RandomStringXOR.cpp' whose xor is equal to ‘0000902010005000’ which we get from the inverse initial permutation of ‘405c0000 04000000’ and stored in 'random_input.txt'.

We fed this input 'random_input.txt' to the game terminal using 'script.sh' and stored the output in 'output.txt'. Since the outputs also contained other lines apart from the cipher text we used, 'output_cleaner.cpp' to clean the outputs and store them in 'cleaned_output.txt'.

This 'cleaned_output.txt' is fed as an input to 'DES_keygen.py'. In 'DES_keygen.py' we perform the following operations, first, we do the reverse permutation of the block R6 to get the output of the 6 round of DES say Output_1, followed by output differentials that is the xor of alternate pairs of Output_1. We expand Output_1's left half to get the input differential of S-BOXes as the left half of ouput_1 is equal to the right half output of round 5. The input differential are stored in 'inp_diff.txt' We then inverse permute the right half of Output_1 to obtain S-Box's output differential and store it in out_diff.txt. Now from every two entries in Output_1(output1, output2) we extract left half of output1, expand it and store it to out_exp.txt, i.e. it stores the intermediate value in 6th round after the expansion box before performing the XOR operation.

Now we try to extract the 6 bit key for every correct corresponding input-output differential pairs with the help of out_exp.txt. First for every entry in inp_diff.txt, for ech SBOX input pair is generated, so that the XOR of the 2 inputs is same as the iteration inp_diff.txt values(say inp1 and inp2), and if the output pair's XOR is equal to that iteration's out_diff.txt value, we update the counter of out_exp.txt XOR inp1. After that, we find out the 6 bit key sets for each S-Box having high probability. We analyzed the frequency and found out that S-Boxes S3 and S4's maximum frequency is more or less the same as the average frequency. So we are unable to find them this way. This means that out of 56 bits we have knowledge of 36 bits. So what we do now is that we perform bruteforce analysis by trying 2^20 permutations and store these keys in 'Keys.txt'.

After that, we have used DESbruteforce.cpp to find out the correct key from all the possible combination of keys

input: ffffffffffffffff (0 0 0 0 0 0 0 0)
output: mhqqtqqkislqoski (144 187 235 181 61 107 157 83)
key: 01101110010111100111011101001100011101010100001110100111

For mapping letters to numbers, we have used the code 'converter.cpp'

Then we run DecryptDES.cpp to decrypt the password using the above key

Encrypted Password--> luomqnlfnkuuulitiulsqimkfqgsrmfp (111 151 184 96 133 255 246 62 63 109 179 117 11 29 199 10)

Decrypted Password--> 113 117 115 106 104 105 99 115 105 120 48 48 48 48 48 48

One interesting thing to notice here is that the numbers lie between 97 and 122 and there is an unusually long sequence of 48 at the end which can be easily guessed as padding but. So we convert the numbers using the ASCII scheme and get the password. The six 48 at the last are just for padding and it converts to 0. It does not include in password.

The password is--> qusjhicsix

References--> For the DESbruteforce.cpp code we have referred this "https://cryptofever104237425.wordpress.com/2011/10/20/decrypt-des/" website and taken the help from it. Apart from this, we have followed the lecture slides provided by Manindra sir, to solve the problems

No files uploaded

Q5 Password

5 Points

What was the final command used to clear this level?

qusjhicsix

Q6 Codes

0 Points

Unlike previous assignments, this time it is mandatory that you upload the codes used in the cryptanalysis. If you fail to do so, you will be given 0 marks for the entire assignment.

Cryto assignment 4.zip Download

```
Binary file hidden. You can download it using the button above.
```

Assignment 4

GRADED

6 DAYS, 17 HOURS LATE

GROUP

YASH SARASWAT
HIRAK MONDAL
MAYANK BANSAL

View or edit group

TOTAL POINTS

90 / 100 pts

QUESTION 1 Teamname	**0** / 0 pts
QUESTION 2 Commands	**0** / 10 pts
QUESTION 3 Cryptosystem	**5** / 5 pts
QUESTION 4 Analysis	**80** / 80 pts
QUESTION 5 Password	**5** / 5 pts
QUESTION 6 Codes	**0** / 0 pts