Q1 Teamname
0 Points

Cipherberg

Q2 Commands
15 Points

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

exit1, exit2, exit4, exit3, exit1, exit4, exit4, exit2, exit2, exit1, read

Q3 Analysis
60 Points

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.)

On the first screen it was written that exit 1 and exit 2 are open. Since we entered from exit 1, so we entered command exit 2 to proceed. But this didn't helped us to proceed to new screens so we went back to the start of the level and this time we proceeded with exit 1. Then, the hexadecimal text 59 6f 75 20 73 65 65 20 appeared on the screen. From each screen, only one command/numbered exit took us to a new screen having different combination of hexadecimal characters, while other exits took us to some previously encountered screen.

After using a unique combination of exits such that no screen (or combination of hexadecimal characters) is encountered twice, we obtained the following list of hexadecimal numbers:

"59 6f 75 20 73 65 65 20 61 20 47 6f 6c 64 2d 42 75 67 20 69 6e 20 6f 6e 65 20 63 6f 72 6e 65 72 2e 20 49 74 20 69 73 20 74 68 65 20 6b 65 79 20 74 6f 20 61 20 74 72 65 61 73 75 72 65 20 66 6f 75 6e 64 20 62 79"
We used $cipher.ipynb$ to first convert these hexadecimal numbers to decimal and then found ASCII value for the decimal values obtained. This gave us the following message:
``You see a Gold-Bug in one corner. It is the key to a treasure found by "

On the last screen, after entering read, we got the following message,
"n =
8436444373572503486440255453382627917470389343976334334386326034275667860921689509377926302880924650595564757217668266944527000881648177170141755476887128502044240300164925440505830343990622920190959934866956569753433165201951640951480026588738853928338105393743349699444214641968202764907970498260085751709
3
Cipherberg: This door has RSA encryption with exponent 5 and the password is
2370178774682911039678909490731983030553818037642728322629590658530188954399653341053938177968436688097089627901880710053017665162508698865521085855413334590627256102779817144092314796016509489198045275785268570702028938469832266534760990574458224815724693200797833912963006702298796670695548259886980015169 3
"

Since RSA is used for encryption, so we could easily decrypt it using $dec(C, d, n) \equiv C^d \bmod \mathrm{n}$, here d is unknown, C is ciphertext and n is modulus. To decrypt the password either we need to find factors of n or find the value of d. Since n is a large number so finding it's factors is not possible. Also, since n is not factorisable, therefore we cannot compute $\phi(n)$ and hence cannot compute d.
Since, the public exponent is 5, so a low-exponent RSA attack is possible. We used Coppersmith's algorithm and LLL Lattice Reduction Technique to attack this RSA problem.

To apply the algorithm, first we check if padding has been used. To check this, we determine $C^{1/e}$ (e is public exponent key), which turns out to be a non-integer. So padding P has been used. The new equation becomes,

$(P+M)^e \equiv C \bmod \mathrm{n}$

Here, e, C and n are known. P is the sentence found above from hexadecimal characters on screen i.e. ``You see a Gold-Bug in one corner. It is the key to a treasure found by ''. M is the original message.

Coppersmith's Algorithm:

Let n be a positive integer, $f(x) = \mathcal{Z}(x)$ be a polynomial of degree d. Given n and f, all integers x_0 such that $f(x_0) \equiv 0 \bmod \mathrm{n}$ and $x_0 < n^{(1/d)-\epsilon}$ can be recovered in polynomial time, where $(1/d) > \epsilon > 0$. [2]

Using this, we can state our problem as $f(x) = (P + x)^e \bmod \mathrm{n}$. Here, x is polynomial ring of integers over modulo n, P is the padding and e = 5. Roots of f(x) is the required password. To solve for x we used $code.sage$ taken and modified from [1], to perform the following steps:

1. Translate padding P to binary.
2. Length of password x is unknown, but due to the assumption $x < n^{1/e}$, length of password x cannot be longer than 200 bits.
3. Therefore, the final polynomial becomes: $((binary\_P << password\_length) + x)^e - C$, where password_length is varied from 1 to 200 in multiples of 4.

Using Coppersmith's algorithm and LLL, the following root was obtained when password_length=64:

100001001000000011010000111010101100010010000010110110000100001

For 64-bit root, we added one 0 and the root is: 01000010 01000000 01101000 01110101 01100010 01000001 01101100 00100001

Taking 8 bits at a time, we found the corresponding ASCII value and obtained the password as : **B@hubAl!**

Ref:

[1] https://github.com/mimoo/RSA-and-LLL-attacks/

[2] https://en.wikipedia.org/wiki/Coppersmith_method

code.sage was run on https://sagecell.sagemath.org

No files uploaded

Q4 Password

25 Points

What was the final command used to clear this level?

B@hubAl!

Q5 Codes

0 Points

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.

▾ Assignment-6-Cipherberg.zip Download

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

Assignment 6

GRADED

GROUP

SHRUTI SHARMA

SAMBHRANT MAURYA

DEEKSHA ARORA

View or edit group

TOTAL POINTS

100 / 100 pts

QUESTION 1

Teamname **0** / 0 pts

QUESTION 2

Commands **15** / 15 pts

QUESTION 3

Analysis **60** / 60 pts

QUESTION 4

Password **25** / 25 pts

QUESTION 5

Codes **0** / 0 pts