LACUNA
SOFTWARE_

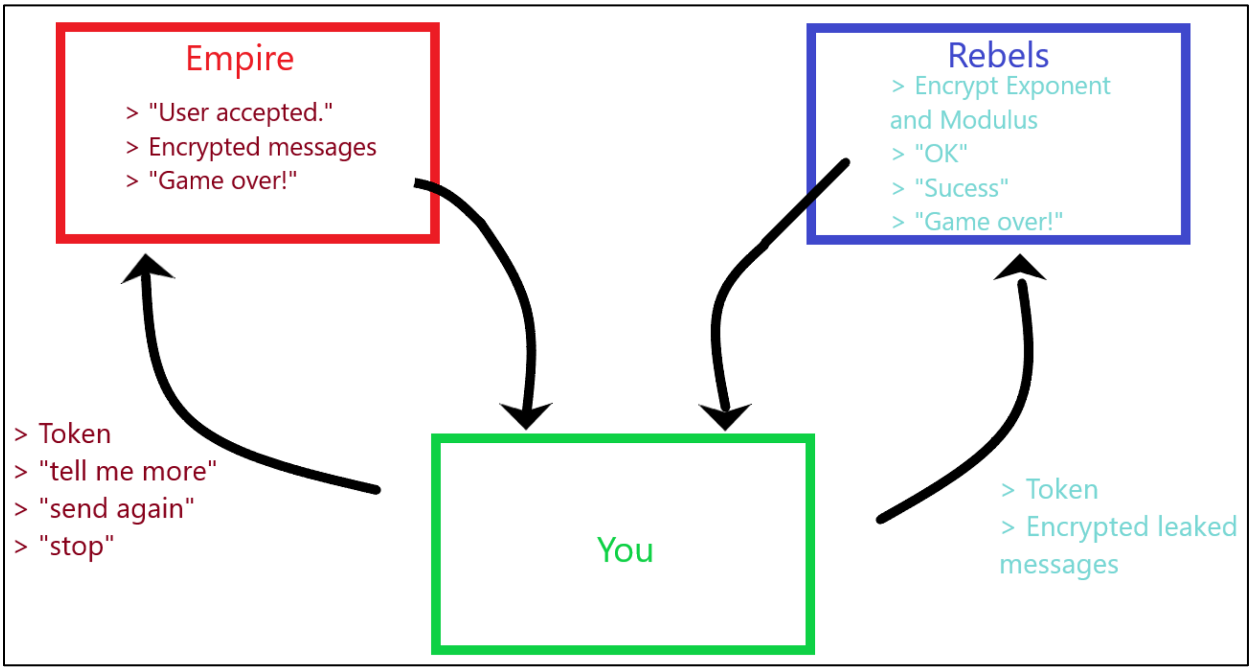
Admission Test – Rogue One

0.3.2|31/07/17

WELCOME ADVENTURER!

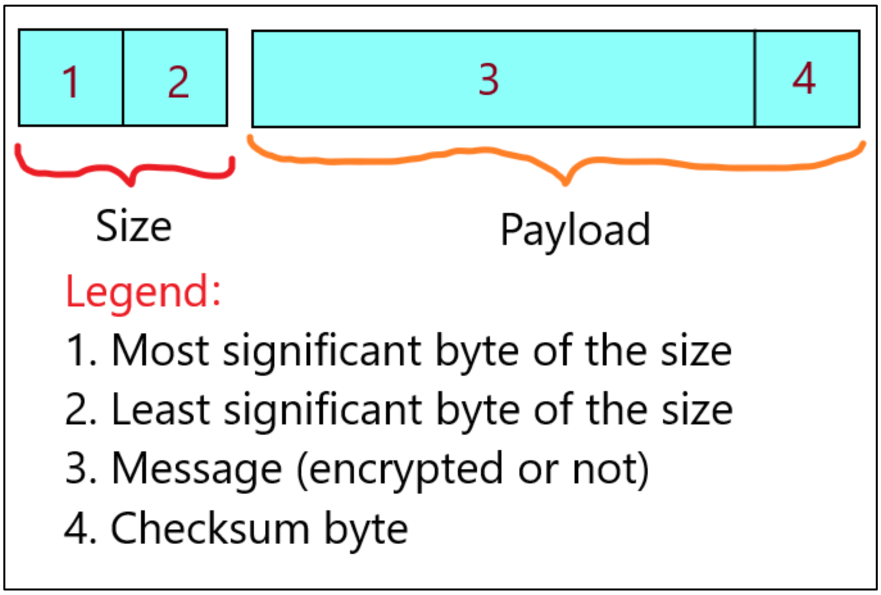
If you are reading this message, it means that the Rogue Force has reached you successfully and it is in need of your help. You, as a great hacker, must help them to acquire the Imperial resource coordinates, decoding and processing the Imperial leaked messages. The Empire is about to reach the rebel base, so the clock is ticking and **you have exactly one week to face this challenge**, in order to turn the tides in favor of the Rebels.

Brave challenger, this briefing is essential for any chance of success during the mission, requiring attention to every detail.



INSTRUCTIONS

1. In order to get into the Imperial Network, connect to the following TCP/IP server: IP – lacuna.ddns.net; your personal Empire port is available at the test website. The first message must be simply the token available at the test website. To communicate with the Empire Server, **send your messages as an ASCII stream**, without following the Empire pattern, simply by getting the bytes from the whole string.
2. Imperial messages follow a pattern: as header, 2 bytes for the size of the payload (message + checksum); as payload, the message bytes and 1 byte for checksum, the last byte of the payload **(the checksum is calculated including the 2 bytes of the size).** Pay attention to the bytes read from the server, there may be delay in the network and it may be at any point of the data stream, including inside the payload byte stream.



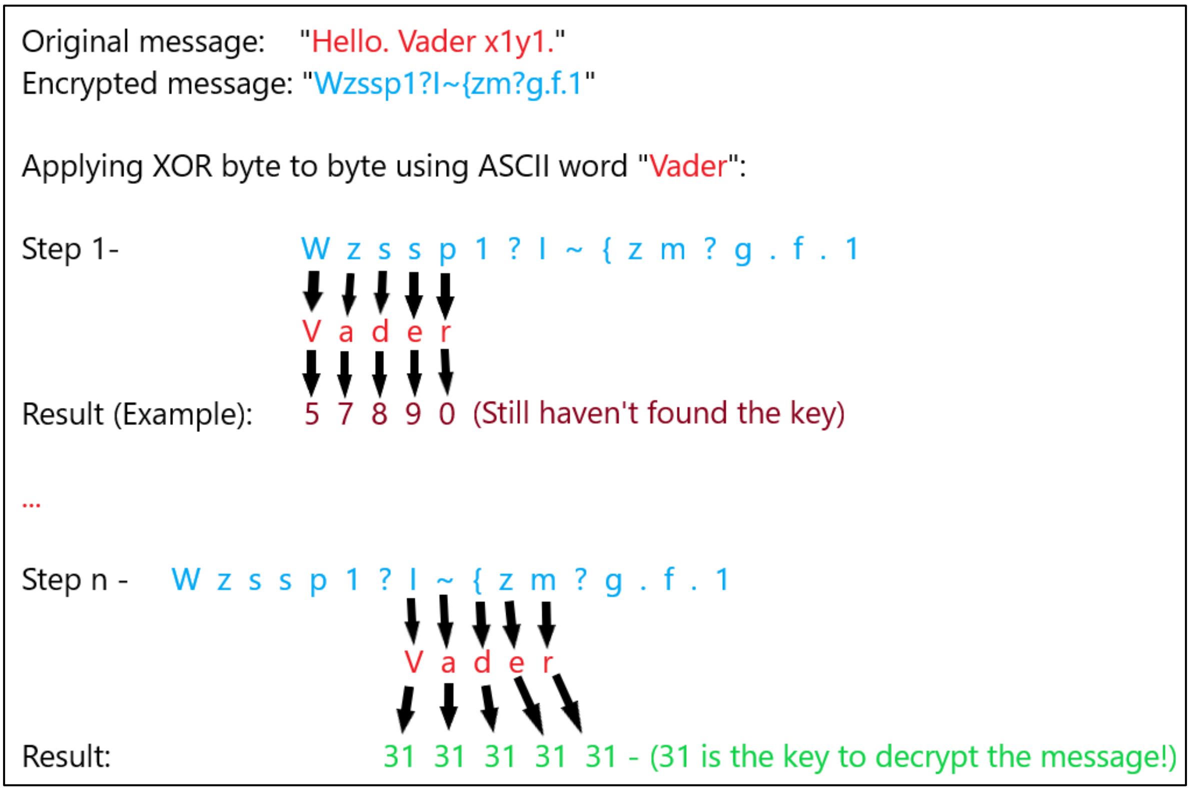
Legend:

1. Most significant byte of the size
2. Least significant byte of the size
3. Message (encrypted or not)
4. Checksum byte

3. To calculate the checksum, add all the integer values from each character of the message and add the integer values from the two bytes that represent the size. The least significant byte from the result of these operations is the checksum.
4. **The server will always respond following the pattern above**. After sending the token, if the token is correct, the server will send an ASCII non-encrypted message saying "User accepted."; otherwise, you will receive an ASCII non-encrypted message saying "Game over!".
5. After sending your token and receiving the "User accepted." message from the Imperial server, you will be able to send, as a simple ASCII byte stream, 3 kinds of commands on that server:

   - "tell me more" – each time you send this command, the server will send an encrypted message;
   - "send again" – the server will send only the payload again (message + checksum);
   - "stop" – the Empire Server will close the connection with you.

6. Each payload contains an encrypted message with a secret decryption key. The only thing known is that every message contains one string "Vader" and the operation used to encrypt is a bit-wise XOR. Since the key used to encrypt is 1-byte-wide, this ciphertext is created by applying a bit-wise XOR on each of the characters of the payload. Some people say that it is possible to find out what the encryption key is by applying bit-wise XOR of the word "Vader" to any part of the message, until 5 consecutive characters produce the same result, then you'll get your key!



Original message: "Hello. Vader x1y1."
Encrypted message: "Wzssp1?l~{zm?g.f.1"

Applying XOR byte to byte using ASCII word "Vader":

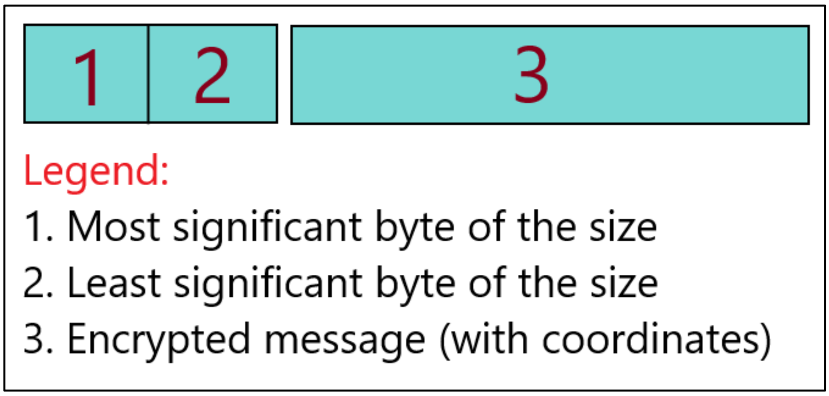
Step 1- W z s s p 1 ? l ~ { z m ? g . f . 1
V a d e r
Result (Example): 5 7 8 9 0 (Still haven't found the key)

...

Step n - W z s s p 1 ? l ~ { z m ? g . f . 1
V a d e r
Result: 31 31 31 31 31 - (31 is the key to decrypt the message!)

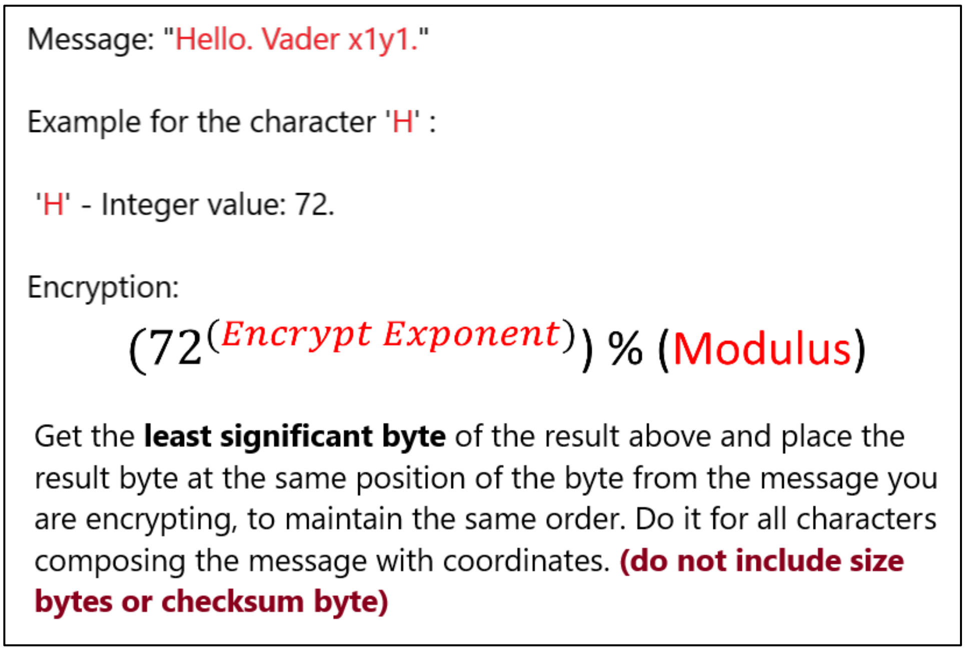
7. If the checksum of a message from the server is not right, send a simple "send again" message with no encryption, in order to receive again only the payload (message + checksum).
8. Once inside the Imperial system, in order to receive leaked messages, it's necessary to send to the server a simple "tell me more" message. You have to send “tell me more” each time you want to receive a message.
9. Some decrypted messages will contain the coordinates of Imperial resources, and we need them!! We have discovered that the Empire always use the exact following pattern to represent coordinates: it starts by the character 'x', followed by a number, then the character 'y', followed by another number. It is important to mention that there are no spaces inside this notation, and both numbers have unknown amounts of digits. Examples: "x90y7", "x12345y76", "x76y92837", "x9y2". They may be in any position of the payload and only this exact pattern is valid;
10. After acquiring all the messages with valid coordinates, half of the work is done, but if these informations don't get to the Rebels, all the efforts will be meaningless, so it's necessary a safe way to send such information. The Rebel Server is: IP - lacuna.ddns.net; your personal Rebel port is available at the test website. To start the communication with the Rebels just send the token available at the test website. If the token is not valid, the rebels will send a byte stream equal to the string “Game over!”, as an ASCII byte stream, and the connection with the Rebels will be lost.
11. To establish the secure channel, you will have to follow the encryption algorithm created by the Rebels and known as BFF. **The Rebels will only accept the stolen messages from the Empire** (after the token) **if you send them in this encryption and following the**

pattern: 2 bytes for the size of the message (do not encrypt the size bytes) and the rest of the bytes are the encrypted message **without checksum.**



Legend:
1. Most significant byte of the size
2. Least significant byte of the size
3. Encrypted message (with coordinates)

12. So, after stablishing the connection with the rebels as explained above, they will send you the BFF public key you need to cypher your message. The public key will be sent in a string with two numbers inside it, separated by a space, in the format "XXXX YYYY", the number of digits is not fixed. In this string, the XXXX is the **Encrypt Exponent**, and the YYYY is the **Modulus**.
13. The BFF algorithm works similarly to a public key pattern encryption, making it very hard do decrypt the messages if you don't have the private key. You will receive your public key from the Rebels, composed by 2 numbers: one for the **Encrypt Exponent** and the other on is the **Modulus**, and will use it to encrypt the stolen messages.
14. The encryption is made **byte by byte**. So, for each byte of the message with coordinates, raise its integer value to the power of **received Encrypt Exponent,** applying the integer modulo operation on that result, using the received **Modulus**. Do not include the bytes of size and the checksum byte in this operation.



Message: "Hello. Vader x1y1."

Example for the character 'H' :

'H' - Integer value: 72.

Encryption:

$$(72^{(Encrypt\ Exponent)})\ \%\ (Modulus)$$

Get the **least significant byte** of the result above and place the result byte at the same position of the byte from the message you are encrypting, to maintain the same order. Do it for all characters composing the message with coordinates. **(do not include size bytes or checksum byte)**

Useful links:

- https://stackoverflow.com/questions/5171002/rsa-calculating-cd-mod-n - Second answer;
- https://en.wikipedia.org/wiki/Modular_exponentiation

15. You have to send to the Rebels only the messages that contains coordinates. At any given time, the rebels can tell that you failed with the simple "Game over!" message, closing the connection with them. In this case, you should send the message "stop" to the Empire server, so it could close the connection without raising any alarms. If you deliver a needed message, the rebels will send you a simple "OK" message, **as an ASCII byte stream**.
16. It's a good choice to receive a message from the Empire, and if it has coordinates, encrypt and send it to the Rebels, and keep listening to the Empire's messages until the Rebels tell you that they already have what they need, with the simple message "**Success**" (this is the only breakpoint), as an ASCII byte stream. After receiving the final needed message from the Empire and after the "Success", as a simple ASCII byte stream message from the Rebels, immediately send a "stop" message to the Empire, as an ASCII byte stream. If you try to read more messages than the amount needed, the Empire will notice your invasion and you will receive a "Game over!" message.

> **The solution will only be accepted if it was written in C/C++/C# or Java**.

May the force be with you, brave hero, and don't forget:

You are our only hope.