Lost Key

257 points, 29 solves

Statement

```
I lost my public key. Can you find it for me?
nc 18.179.251.168 21700
rsa-b667a9ca0d5c6735e5609565d1fd6ab9.py
```

Solution

We are given a RSA encryption/decryption program and encrypted flag. We can perform up to 150 encrypting/decrypting operations, however decryption returns only the last byte of the decrypted message for us. Additionaly, all constants (including n and e) are unknown to us.

First of all, we would like to retrieve the n value. Let's put some random number (eg. 2) through encryption (so that the program will return $2^e \ mod\ n$. Now run the encryption for the square of our number (4 in this case). The program will return $4^e \ mod\ n = (2^e)^2 \ mod\ n$. If we square the first returned number ($2^e \ mod\ n$) we will get a number with same remainder modulo n as the second number returned by the program. So their difference will be divisible by n. We can repeat this process several times for random values different than 2 and for each repetition we will find a new number divisible by n. Taking GCD of these numbers will result in n with extremely high probability. (I was looking just at the result returned by taking 2, 3, 5 and 7 and it was enough to find n everytime I ran the program).

Now that we know n, what can we do with it? It's 1024 bits large so it's surely too big to factor. We must somehow use the decrypting function to our advantage. If we decrypt the flag, we will get the last byte of it (newline character for this instance). But now that we know N, we can try to alter the encrypted message in such way that it would return the second byte. We know the remainder of the flag modulo 256. We can calculate multiplicative inverse of 256 modulo N, encrypt it (so that we have $inv^e \ mod\ n$) and multiply it with the encrypted flag. Putting it through decryption will return $inv * flag\ mod\ 256$. We also know that $flag = whatwefoundsofar + 256 * restoftheflag$, so $flag * inv\ mod\ n = restoftheflag + whatwefoundsofar * inv$. We know both $whatwefoundsofar$ and inv so we can substract their product from returned value and we will get the last byte of the $restoftheflag$, which is equal to the second byte of the flag! We can perform this process several times, taking multiplicative inverses of higher powers of 256 to get the whole flag!

Flag: `hitcon{1east_4ign1f1cant_BYTE_0racle_is_m0re_pow3rfu1!}`