Κρυπτογραφία 2020 – Εργασία 5ου εξαμήνου

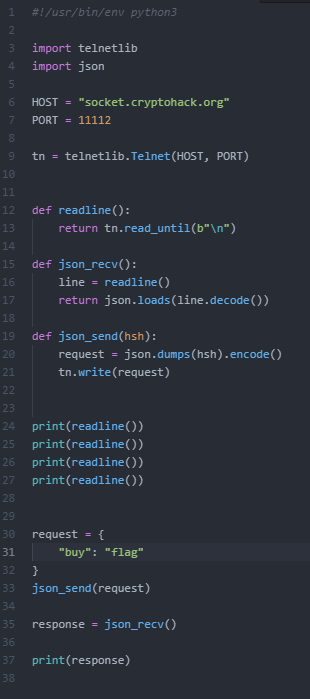
Αντώνιος Ελευθέριος Καρναβάς – Π18063

Introduction (3/3, 10 points)

Finding Flags) The only flag that made sense: crypto{y0ur\_f1rst\_fl4g}

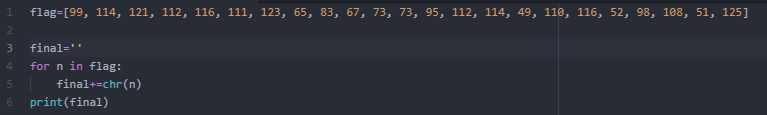
Great Snakes) After I ran the great\_snakes.py script I got this flag: crypto{z3n\_0f\_pyth0n}

Network Attacks) I ran the telnetlib\_example.py after I made a change in the value of the json key ‘buy’ and I got this flag: crypto{sh0pp1ng\_f0r\_fl4g5}



General (15/18, 285 points)

ASCII) I wrote the following script and got as an output this flag: crypto{ASCII\_pr1nt4bl3}



HEX) Wrote this line of code: print(bytes.fromhex("63727970746f7b596f755f77696c6c5f62655f776f726b696e675f776974685f6865785f737472696e67735f615f6c6f747d"))

and got back this flag: crypto{You\_will\_be\_working\_with\_hex\_strings\_a\_lot}

Base64) Wrote this line of code:

print(base64.b64encode(bytes.fromhex('72bca9b68fc16ac7beeb8f849dca1d8a783e8acf9679bf9269f7bf')))

and got back this flag: crypto/Base+64+Encoding+is+Web+Safe/

Bytes and Big Integers) Wrote this line of code:

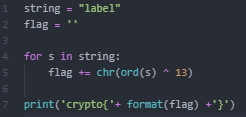
print(long\_to\_bytes('11515195063862318899931685488813747395775516287289682636499965282714637259206269'))

and got back this flag: crypto{3nc0d1n6\_4ll\_7h3\_w4y\_d0wn}

Encoding Challenge) Wrote the following code (screenshot) and got back this flag after the 100th time: crypto{3nc0d3\_d3c0d3\_3nc0d3}. I saw how the server worked (if statements for each encoding type so I made the same statements but for decoding and the 100th time the Json object had 1 attribute named flag and not 2 named type and encoded. So when it was flag, broke the loop)



XOR Starter) XOR every character of the word label (unicoded) with the integer 13 using the following script:



And got back the flag: crypto{aloha}

XOR Properties) To find key2 (and so on) you have to XOR key1 and the first given hex string. I thought of this like an equation that you have to bring the unknown quantities to the one side of = and the known to the other.



And got back the flag: crypto{x0r\_i5\_ass0c1at1v3}

Favorite byte) To find the flag I xored with every byte until I found ‘crypto’ in the decoded(utf-8) message with the following code.

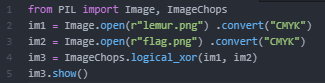
.

And got back the flag: FLAG: crypto{0x10\_15\_my\_f4v0ur173\_by7e}

You either know, XOR you don't) Having a string of data that I know the answer was in the format of “crypto{blah,blah,blah}”, I xored the 7 first characters of the data with the string crypto{ cause I know this is the first part of the answer. I got back "myXORke" and I added a y in the end cause it makes sense. Then I wanted to xor this with the initial data so I repeated the word "myXORkey" like this: myXORkeymyXORkeymyXORkeymyXORkeymyXORkeymy and at the end I had to stop to “my” with a simple modulus. So I xored it and I got back this flag: crypto{1f\_y0u\_Kn0w\_En0uGH\_y0u\_Kn0w\_1t\_4ll}



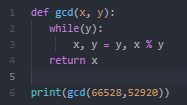
Lemur XOR) The 2 images were xored with the same key so from the xor properties if we xor the 2 we will find the differences. So this is what I did with the following script:



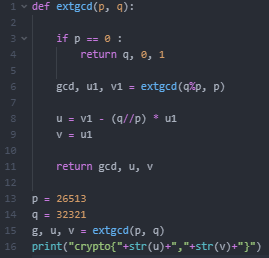
And got back the im3 (lemur.png xor flag.png) with the flag: crypto{X0Rly\_n0t!} as we can see below:



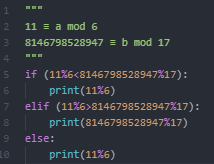
Greatest Common Divisor) Recalled how Euclid’s Algorithm worked, wrote a recursive while loop and got the result: 1512 with the following code.



Extended GCD) Completed a simple recursive implementation of the extended gcd with the following code and got back crypto{10245,-8404}.



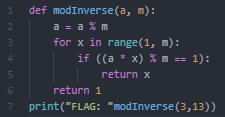
Modular Arithmetic 1) Simple mod calculations. Answer is 4. Code is below:



Modular Arithmetic 2) The answer is 1 based on Fermat’s little theorem.

(a^b)mod(b)=a and (a^(b-1))mod(b)=1

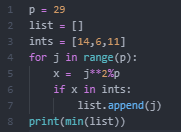
Modular Inverting) Simple modular inverting implementation from previous maths subjects.



and the answer was 9.

Mathematics (1/22, 25 points)

Quadratic Residues) Looped through the prime number that was given (p=29) and calculated the current looped number in the power of 2 modulo the prime number. If that number was in the list ints, I saved the current looped number. I took the min of the saved and was 8.



RSA (10/29, 220 points)

RSA Starter 1) print(pow(101,17,22663)) and the answer was 19906.

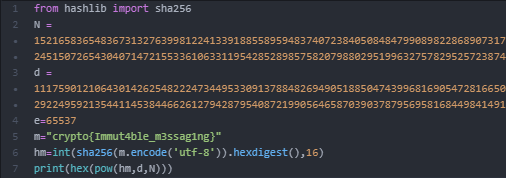
RSA Starter 2) print(pow(12,65537,17\*23)) and the answer was 301.

RSA Starter 3) print((p-1)\*(q-1)), where p,q two large primes and the answer was 882564595536224140639625987657529300394956519977044270821168.

RSA Starter 4) extendedgcd(e,φ(n))= 121832886702415731577073962957377780195510499965398469843281, -9047 where the first number is the private key d.

RSA Starter 5) To decrypt the cyphertext c to the original message m we need to calculate m=c^dmodn where n=p\*q. The answer is 13371337.

RSA Starter 6) With the given N and d I signed the message ”crypto{Immut4ble\_m3ssag1ng}” with the following script:



Factoring) Factored the number 510143758735509025530880200653196460532653147 and the 2 primes where 19704762736204164635843, 25889363174021185185929 with the smallest being the answer.

Inferius Prime) Factorized n with the help of this website, <http://factordb.com/> and then wrote the following code to calculate d, then the original m and then covert the number to bytes using long\_to\_bytes and got back this flag: crypto{N33d\_b1g\_pR1m35}.



Monoprime) Given that n was calculated but the same prime number, we know that p\*p=n. So I calculated the root of n got p, calculated f=(p-1)\*(p-1) and then got the original message by running the code above. Answer: crypto{0n3\_pr1m3\_41n7\_pr1m3\_l0l}.

Square Eyes) Same as before but because the root of a 2048\*2048 bit prime is hard to find I calculated f. I did that using the hint that cryptohack game me which was to use Euler's totient formula. So I calculated f(n) and got back (p-1)\*(p-1). Then I ran the script above to get the private key d , and found the original message: crypto{squar3\_r00t\_i5\_f4st3r\_th4n\_f4ct0r1ng!}.

DIFFIE-HELLMAN (3/11, 60 points)

Diffie-Hellman Starter 1) print(inverse(209,991)) and got back 569.

Diffie-Hellman Starter 3) print(pow(A,b,p)) and got back 1806857697840726523322586721820911358489420128129248078673933653533930681676181753849411715714173604352323556558783759252661061186320274214883104886050164368129191719707402291577330485499513522368289395359523901406138025022522412429238971591272160519144672389532393673832265070057319485399793101182682177465364396277424717543434017666343807276970864475830391776403957550678362368319776566025118492062196941451265638054400177248572271342548616103967411990437357924.

Diffie-Hellman Starter 4) print(pow(A,b,p)) to decrypt our secret using A (alice’s message that I Bob got),b (my secret int) and p the prime and got back this shared secret.

1174130740413820656533832746034841985877302086316388380165984436672307692443711310285014138545204369495478725102882673427892104539120952393788961051992901649694063179853598311473820341215879965343136351436410522850717408445802043003164658348006577408558693502220285700893404674592567626297571222027902631157072143330043118418467094237965591198440803970726604537807146703763571606861448354607502654664700390453794493176794678917352634029713320615865940720837909466