INTRODUCTION

ΔΙΑΔΙΚΑΣΙΑ ΕΥΡΕΣΗΣ FLAG

Finding Flags: Έβαλα το flag που μου δίνει η εκφώνηση

Great Snakes: Έτρεξα το greate_snakes.py

Network Attacks: Άλλαξα το value του key “buy” απο “clothes” σε “flag”

GENERAL

ΔΙΑΔΙΚΑΣΙΑ ΕΥΡΕΣΗΣ FLAG

ASCII: Χρησιμοποίησα την συνάρτηση chr() σε ένα for ώστε να μετατρέψω κάθε αριθμό της λίστας σε ένα χαρακτήρα.

HEX: Μετέτρεψα με την βοήθεια της συνάρτησης bytes.fromhex() το δεκαεξαδικό string σε bytes.

Base64: Μετέτρεψα με την βοήθεια της συνάρτησης bytes.fromhex() το δεκαεξαδικό string σε bytes και μετά χρησιμοποίησα την συνάρτηση base64.b64encode() ωστε να κάνω encode τα bytes. (I imported base64 so I can use the function base64.b64encode())

Bytes and Big Integers : Χρησιμοποίησα το int(string,0) για να κάνω convert το hexstring σε integer ,μετά χρησιμοποίησα την long_to_bytes() για να κάνω αυτόν τον integer σε μύνημα.

(I used from Crypto.Util.number import * so I can use the function long_to_bytes())

Encoding Challenge : Ουσιαστικά έπρεπε να περάσω και τα 100 levels ώστε να πάρω το flag, έχουμε αρχικά το socket για να μπορώ να κάνω request and receive.Σε ένα for loop που τρέχει 101 φορές, αποκοδικοποιώ σε κάθε loopa το μήνυμα και κάνω request με ένα

json,μετά κάνω receive το κωδικοποιημένο μήνυμα και το αποκοδικοποιώ στην επόμενη loopa.Αυτό επναλαμβάνεται 100 φορές και στην 101 πρώτη φορά κάνω receive το flag.

XOR Starter : Έκανα XOR κάθε χαρακτήρα του string με τον αριθμό 13, μετά κάθε νέος αριθμός που δημιουργήθηκε μπήκε σε μια λίστα και απλώς κάθε αριθμό τον μετατρέψαμε σε χαρακτήρα και δημιούργησα το νεο string με κάθε ένα απο αυτούς τους χαρακτήρες.

XOR Properties : Χρησιμοποιήσα τις ιδιότητες του XOR ώστε να δημιουργήσω το KEY2 αλλα και το KEY3 μόνο του.Στη συνέχεια απλώς έκανα και τα τρία κλειδιά μαζί XOR με το string `FLAG ^ KEY1 ^ KEY3 ^ KEY2 = “04ee9855208a2cd59091d04767ae47963170d1660df7f56f5faf”.`

Favourite byte : Έκανα bruteforce για να βρώ το σωστό byte το οποίο όταν το κανω xor με το string μου δίνει το flag.

You either know, XOR you don't : Το μόνο στοιχείο που είχαμε ήταν η μορφή του flag που ήταν (crypto{flag}),άρα ξέρουμε μόνο τους εφτά πρώτους χαραχτήρες “crypto{“ και τον τελευταίο “}”.Εμείς ψάχνουμε το key που όταν το κάνουμε xor με το ciphertext θα μας δώσει το flag.Κάνουμε xor τους πρώτους 7 χαρακτήρες του ciphertext και τον τελευταίο χαρακτήρα με το “crypto{“ και “}” αντίστοιχα,αυτό μας δίνει το key που απλά το κάνουμε xor με ciphertex και μας δίνει το flag.

Greatest Common Divisor : Εφαρμόσαμε την συνάρτηση math.gcd(66528, 52920) (I imported math) και μας έδωσε τον αριθμό που θέλαμε.

Extended GCD : Εφαρμόσαμε τον επεκτεταμένο αλγόριθμο του Ευκλειδη ώστε να πάρουμε τον μικρότερο ακέραιο απο τα u και v:

p * u + q * v = gcd(p,q).

Modular Arithmetic 1: Εκτέλεσα το 11 ≡ x mod 6 και το 8146798528947 ≡ y mod 17, το μιρκότερο απο τα δύο αποτελέσματα ήταν η απάντηση δηλαδή το (8146798528947 ≡ y mod 17).

Modular Arithmetic 2: 27324678765465536 mod 65537 βλέπουμε ότι ο εκθέτης είναι (65537-1) βάσει το θέωρημα Fermat αυτή η πράξη κάνει 1 αμέσως.

Modular Inverting: Χρησιμποιώντας ξανά τον αλγόριθμο του Ευκλειδη μπορέσα να πάρω το υπόλοιπο(-4) όταν βρέθηκε ο μέγιστος κοινός διαιρέτης που ήταν 1.Μετά απλά έκανα υπόλοιπο(-4) mod 13 που αυτό μας δίνει το modular inverse.

Privacy-Enhanced Mail?: Δίαβασα το αρχείο , μετά καταχώρησα το key σε μια μεταβλητή με την βοήθεια της RSA.importkey

(from Crypto.PublicKey import RSA) και τότε είχα πρόσβαση στα attributes της κλάσης μέσω της μεταβλητής μου,άρα έκανα print(key.d).

CERTainly not: Δίαβασα το αρχείο(με το ‘rb’) , μετά καταχώρησα το key σε μια μεταβλητή με την βοήθεια της RSA.importkey

(from Crypto.PublicKey import RSA) και τότε είχα πρόσβαση στα attributes της κλάσης μέσω της μεταβλητής μου,άρα έκανα print(key.n).

SSH Keys: Δίαβασα το αρχείο(με το ‘rb’) , μετά καταχώρησα το key σε μια μεταβλητή με την βοήθεια της RSA.importkey

(from Crypto.PublicKey import RSA) και τότε είχα πρόσβαση στα attributes της κλάσης μέσω της μεταβλητής μου,άρα έκανα print(key.n).

Transparency: Διάβασα το αρχείο transparency.pem,μετα το έκανα convert σε der όπου το καταχώρησα σε μία μεταβλητή (der) μετά την έκανα hash με την βοήθεια της import hashlib και μετά hexdigest όπου μου δίνει το sha256 fingerprint που χρησιμοποιήται σαν ID για ένα certificate,συγκεκριμένα το έκανα search απο εδώ censys.io/certificates και πήρα το ‘’link’’ το έβαλα στο url και μου πέταξε το flag.

PUBLIC-KEY CRYPTOGRAPHY

ΔΙΑΔΙΚΑΣΙΑ ΕΥΡΕΣΗΣ FLAG

RSA_STARTER1: Αντικατέστησα τους αριθμούς στις σωστές θέσης της συνάρτησης pow(`101, 17, 22663`) και έκανα print το αποτέλεσμα της.

RSA_STARTER2: Αντικατέστησα τους αριθμούς στις σωστές θέσης της συνάρτησης pow(`12, e, p*q`) και έκανα print το αποτέλεσμα της.

RSA_STARTER3: Υπολόγισα την συνάρτηση

$\varphi(pq)=\varphi(p)\varphi(q)=(p-1)(q-1)$.

RSA_STARTER4: Υπολόγισα την συνάρτηση

$\varphi(pq)=\varphi(p)\varphi(q)=(p-1)(q-1)$ και βρήκα το modular inverse του

e mod $\varphi(pq)$.

RSA_STARTER5: Αντικατέστησα τους αριθμούς στις σωστές θέσης της συνάρτησης pow(`c, d, N`) και έκανα print το αποτέλεσμα της.

RSA_STARTER6: Έκανα hash το message όπου το μετέτρεψα σε decimal και μετά ύψωσα στην d και έκανα mod N ώστε να κρυπτογραφήσω το μύνημα.

Factoring: Χρησημοποίησα την ιστοσελίδα factordb.com ωστε να παραγωντοποιήσω τον αριθμό που μου δίνετε, στους δύο πρώτους που τον αποτελούν.Ο μικρότερος ήταν η απάντηση.

Inferius Prime: Μπόρεσα να κάνω factorise το n μετά υπολόγισα την phi (p - 1) * (q - 1) μετά υπολόγισα το d (pow(e, -1, phi)) και τέλος ύψωσα το ct στην d μετά έκανα mod n και ο αριθμός που βγήκε σαν αποτέλεσμα

τον έκανα σε bytes με την long_to_bytes()(from Crypto.Util.number import *).

Monoprime: Υπολόγισα την phi (n - 1) μετά υπολόγισα το d (pow(e, -1, phi)) και τέλος ύψωσα το ct στην d μετά έκανα mod n και ο αριθμός που βγήκε σαν αποτέλεσμα τον έκανα σε bytes με την long_to_bytes()(from Crypto.Util.number import *).

Square Eyes: Υπολόγισα την phi n*(n - 1) μετά υπολόγισα το d (pow(e, -1, phi)) και τέλος ύψωσα το ct στην d μετά έκανα mod n και ο αριθμός που βγήκε σαν αποτέλεσμα τον έκανα σε bytes με την long_to_bytes()(from Crypto.Util.number import *).

Manyprime: Υπολόγισα την phi

```
mult=1
for i in range(33): mult = mult * (list[i]-1)   #η list περιέχει τους 33 prime
phi = mult
```

μετά υπολόγισα το d (pow(e, -1, phi)) και τέλος ύψωσα το ct στην d μετά έκανα mod n και ο αριθμός που βγήκε σαν αποτέλεσμα τον έκανα σε bytes με την long_to_bytes()(from Crypto.Util.number import *).

Salty: Το e βασικά ήταν πολύ μικρό και μπόρεσα να κάνω bruteforce το k όπου k :(gmpy2.iroot(ct + k * n, e)) (import gmpy2) το σωστό k μου επέστρεψε τον σωστό αριθμό τον οποίο έκανα έκανα σε bytes με την long_to_bytes()(from Crypto.Util.number import *).

Modulus Inutilis: Το e βασικά ήταν πολύ μικρό και μπόρεσα να κάνω bruteforce το k όπου k :(gmpy2.iroot(ct + k * n, e)) (import gmpy2) το σωστό k μου επέστρεψε τον σωστό αριθμό τον οποίο έκανα έκανα σε bytes με την long_to_bytes()(from Crypto.Util.number import *).

Everything is Big: Χρησιμοποίησα το owiener attack(import owiener) το οποίο δουλέυει ουσιαστικά όταν έχουμε πολύ μεγάλους αριθμούς,υπολόγισα το d (owiener.attack(e, n)) μετά ύψωσα το c στην d και μετά mod n ο αριθμός που εμφανίστηκε τον έκανα έκανα σε bytes με την long_to_bytes()(from Crypto.Util.number import *).

Diffie-Hellman

ΔΙΑΔΙΚΑΣΙΑ ΕΥΡΕΣΗΣ FLAG

Diffie-Hellman Starter 1 : Χρησιμποιώντας τον αλγόριθμο του Ευκλειδη μπορέσα να πάρω το υπόλοιπο όταν βρέθηκε ο μέγιστος κοινός διαιρέτης.Μετά απλά έκανα υπόλοιπο mod 991 που αυτό μας δίνει το modular inverse.

ΟΙ ΣΥΝΔΕΣΜΟΙ ΠΟΥ ΔΕΙΧΝΟΥΝ ΟΤΙ ΕΚΑΝΑ ΤΑ CHALLENGES

https://cryptohack.org/user/Rodolfos/