FCSC 2022 : **T-Rex** (Write-up)

Jakobus

Mai 2022

- **Catégorie** : Crypto
- **Points** : 500 (score dynamique)
- **Résolutions** : 67
- **Description** : *Vous devez déchiffrer le flag :-)*
- **Fichiers** : `output.txt`,`T-Rex.py`

1 Découverte de la vulnérabilité

Le script `T-Rex.py` consiste en le chiffrement via AES du flag, rien de très original pour le moment. l'IV est cependant généré de manière étrange : il s'agit de la 31337-ième itérée de la clé (convertie préalablement en entier) par l'application polynômiale $x \mapsto (2x+1)x \pmod{2^{128}}$. Rien de très convaincant donc, puisqu'il est en fait assez aisé de déterminer les racines d'un polynôme dans $\mathbb{Z}/p^n\mathbb{Z}$ avec p premier (et donc en fait dans $\mathbb{Z}/N\mathbb{Z}$ pour tout entier N par les restes Chinois), donc de déterminer les préimages d'un entier par un tel polynôme. On va utiliser le fameux **lemme de Hensel**.

2 Inversion de l'algorithme

Ce lemme (dans une version faible) dit la chose suivante : soit f un polynôme à coefficients entiers, p un nombre premier, $k \geq 1$ et $r \in \mathbb{Z}$ tel que $f(r) \equiv 0 \pmod{p^k}$. Alors si $f'(r) \not\equiv 0 \pmod{p}$, on peut trouver $s \in \mathbb{Z}$, unique modulo p^{k+1} tel que $f(s) \equiv 0 \pmod{p^{k+1}}$ et $s \equiv r \pmod{p^k}$.
En fait, $s = tp^k + r \pmod{p^{k+1}}$ où $t = -\dfrac{f(r)}{p^k} f'(r)^{-1} \pmod{p}$, ce qui permet de facilement et récursivement trouver l'intégralité des racines de f modulo p^k puisqu'une telle racine en est aussi une modulo p^ℓ pour tout $\ell \leqslant k$.
On va ainsi procéder de la façon suivante :

- initialiser $R = \{IV\}$, $i = 0$
- trouver l'ensemble R' des racines de $(2X+1)X - r \pmod{2^{128}}$ pour chaque $r \in R$
- incrémenter i et recommencer l'étape 2 tant que $i < 31337$ avec $R = R'$.

3 Exploitation

En fait, on peut remarquer que, pour tout $r \in \mathbb{Z}$, $(2X+1)X - r \equiv X - r \pmod{2}$ qui n'admet qu'une seule racine modulo 2, à savoir $r \pmod{2}$. Ainsi, le lemme de Hensel assure l'existence d'une unique racine de $(2X+1)X - r \pmod{2^{128}}$, ce qui rend les choses bien moins exponentielles en termes de calcul (R n'a qu'un seul élément à chaque étape).
Implémentons tout cela, en Python :

```python
from Crypto.Cipher import AES


iv = b'\x07\nJX\x11\xdd\x01<0\x1f0\x07\t$R\x87'
cipher = bytes.fromhex("96cb8fe8ddd6d8851f90ab1a8977e9c71accbed0a5936414445739ce76763002fd2
9337834c8976fef36decdc522a6b93c967c90d0e69e
46674d634ba5a9badbd834bad8042515029b6fa833c98da0a7")

def find_roots(m, N):
  #trouver les racines de X^2 + X - m modulo 2^n avec el famoso lemme de Hensel.
  r = m % 2 #solution de base
  for k in range(1, N):
    t = (-(2 * r ** 2 + r - m) // 2 ** k) % 2
    r = (r + 2 ** k * t) % (2 ** (k+1))
  return r

N = 128
r = int.from_bytes(iv, "big")

for i in range(31337):
  r = find_roots(r, N)

key = int.to_bytes(r, 16, "big")

E = AES.new(key, AES.MODE_CBC, iv = iv)
print(E.decrypt(cipher))
```

qui nous donne bien le flag : `FCSC{54a680c151c2bff32fd2fdc12b4f8558012dc71e429f075bab6bfc0322354bf4}` :)