Dima Oana-Teodora 341

Laborator 6 – Securitatea Sistemelor Informatice

Excercitiul 1

C1 va avea mereu ca rezultat valoarea 0.

C2 calculeaza starile urmatoare, seed avand mereu valoarea constanta 0.

C3 face shiftare cu 2 biti, iar valoarea returnata va depinde de updatarea starii (0 daca este apdatata si aceeasi valoare in caz contrar).

Concluzie: valoarea va fi mereu constanta, deci nu e generare de numere pseudo-aleatoare, prin urmare secvenetele de cod C1-C3 nu definesc un PRNG.

ex1.py > ...

```python
seed = int(input("introduceti seed "))

# C1
print("C1")
try:
    while True:
        print(seed)
        seed = seed ^ seed
except KeyboardInterrupt:
    pass

# C2
print("C2")
try:
    while True:
        print(seed)
        seed = int(seed + seed / 2 )
except KeyboardInterrupt:
    pass

# C3
print("C3")
print(seed >> 2)
```

Exercitiul 2

ex2.py > verifyPassword

```python
import string
import secrets
import hashlib

# Generează o parolă de minim 10 caractere care conține cel puțin o literă mare, o
# literă mică, o cifră și un caracter special (.!$@)
# Utilizare: generare parola sigura
def verifyPassword(password):
    if (any(c.islower() for c in password)
            and any(c.isupper() for c in password)
            and any(c.isdigit() for c in password)
            and any(c in ",.!$@" for c in password)):
        return True
    return False

while True:
    password = ""
    for i in range(10):
        password += secrets.choice(string.ascii_letters + string.digits + ",.!$@")

    if verifyPassword(password):
        break
print("Parola generata este: ", password)
```

```
24
25  # Generează un string URL-safe de (cel puțin) 32 caractere.
26  # Utilizare: forgot password?, confirm email
27  url_safe = secrets.token_urlsafe(32)
28  print("Stringul Url_safe generat: ", url_safe)
29
30  # token hexazecimal de (cel puțin) 32 cifre hexazecimale
31  # Utilizare: sistem de autentificare
32  token = secrets.token_hex(32)
33  print("Token-ul hexazecimal generat: ", token)
34
35  # verifica daca doua string uri sunt egale si minimizeaza riscul unui atac in timp
36  s1 = secrets.token_hex(32)
37  s2 = secrets.token_hex(32)
38  result = secrets.compare_digest(s1, s2)
39  print("Verificare daca doua secvente sunt identice: ", result)
40
41  # cheie binara fluida(100 de caractere)
42  # Utilizare: cripatera unui mesaj
43  key = secrets.token_bytes(100)
44  print("Cheia generata este:", key)
45
```

```
45
46  # Stochează parole folosind un modul / o librărie care
47  # să ofere un nivel suficient de securitate
48  # Utilizare: prevenirea spargerii conturilor
49  raw_password = "!Admin01"
50  hashed_password = hashlib.sha256(raw_password.encode('utf-8'))
51  print ('Parola criptata este:', hashed_password.hexdigest())
```

```
PS C:\Users\Oana\Desktop\lab 6 SSI> & C:/Users/Oana/AppData/Local/Programs/Python/Python38/python.exe "c:/Users/Oana/Desktop/lab 6 SSI/ex2.py"
Parola generata este:  .EWgTNHEi6
Stringul Url_safe generat:  gCfenTxLXjnYcXlJ5_-HhymFRA9Z8aQOrgAoWbdYVws
Token-ul hexazecimal generat:  77e9c9ce22f2947a08b152ea8dfa0f00446e186670c30366315e97d73be85991
Verificare daca doua secvente sunt identice:  False
Cheia generata este: b'D\xf6Q\x85\xec\xe5\xa4g\xf9g\xac0Y\xcfs#B\x9c\n\xe3\x9d\x8c\x8f\x1b\xf2\xb8J\x10\xbd\xab\xd4\x93;\xd5\t#\xc7Fe\xeei^U\xd9#Gq\xb7\rl_\x01\xa9\x
94\x08\xb7I(>\xe3\xfe\x8bi\xeb\x0e\xa4o\xb0\xda\xb9n\xee<^}q\x18\xf6\xfb\xa3\xa7\xe4XR4>\xd1)\xe7\xba\xbf\x9c\xcd\xc7\xa1E\x8b\x15\xb5\xf0'
Parola criptata este: 0a9c0043058eb7a4c3fd54865948d7c86c5db2772cdf8c16ad0a13da588c7438
PS C:\Users\Oana\Desktop\lab 6 SSI>
```

Exercitiul 3

- Ce problemă identificați în următoarele secvențe de cod?

Figura 1 (Java): este acelasi seed, deci la fiecare generare a numerelor se vor obtine aceleasi numere in ordine, astfel sistemul va deveni vulnerabil.

Figura 2 (PHP): seed-ul PRNG-ului este id-ul userului, deci la fiecare noua rulare a codului id-ul sesiunii va fi acelasi pentru ca seed-ul ramane acelasi, , astfel sistemul va deveni vulnerabil.

- Care este CWE ID asociat scenariilor de mai sus si problemei pe care acestea o ridică?
  Weakness Id= 336

https://cwe.mitre.org/data/definitions/336.html

- Ce se întâmplă dacă nu se folosește același seed de fiecare dată, dar spațiul seed-urilor posibile este mic? Puteți găsi un CWE ID corespunzător acestui caz?

  spatiul seed-urilor care este mai mic : Small Seed Space in PRNG (CWE-339) => un numar mai mic de posibile valori => creste probabilitatea de a afla seed-ul daca atacatorul foloseste *brute force*

  https://cwe.mitre.org/data/definitions/339.html

- Căutați atacul identificat la punctul precedent în [5]. Identificați și aici o mențiune la seed?
  CAPEC-112
  https://capec.mitre.org/data/definitions/112.html - brute force

- Găsiți alte utilizări defectuoase ale PRNG explicate în alte CWE-uri. Există CVE-uri corespunzătoare acestora?
  https://cwe.mitre.org/data/definitions/338.html - se pot folosi generatoare de numere care nu sunt proiectate pt criptografie (putere de procesare mica, surse finite etc).
  https://cwe.mitre.org/data/definitions/335.html - seed utilizat necorespunzator.

- Căutați înregistrări CVE care se referă la vulnerabilități în legătură cu PRNG. Câte ați identificat ca fiind definite în acest an?
  https://www.cve.org/CVERecord?id=CVE-2021-3678