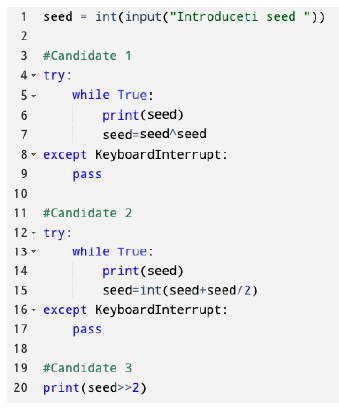
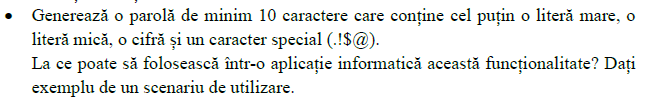
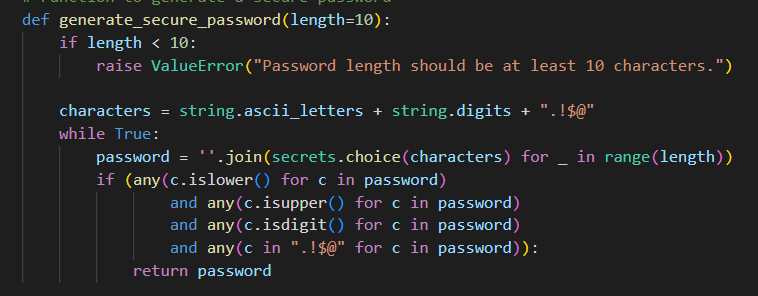
**Laboratorul 5**

1. Notiuni introductive

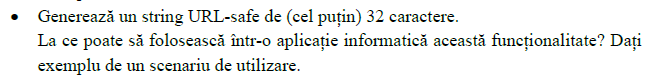
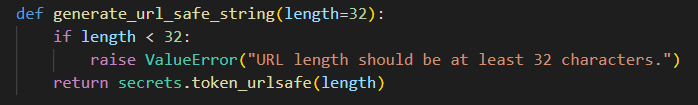


* **Candidatul 1:** Acest fragment de cod efectuează operația XOR pe variabila seed cu ea însăși. Rezultatul operației XOR a oricărui număr cu sine însuși este întotdeauna 0, așa că acest cod va afisa valoarea inițială a seed o singură dată și apoi va afisa 0 în mod repetitiv. Aceasta nu este comportamentul unui PRNG.
* **Candidatul 2**: Acest fragment scrie valoarea variabilei seed și apoi o împarte la 2 (convertind rezultatul la un întreg), continuând cu această nouă valoare a variabilei seed. Acesta este mai apropiat de un PRNG, dar totuși nu este unul convențional deoarece ar reduce previzibil variabila seed la jumătate în fiecare iterație.
* **Candidatul 3**: Acest candidat afiseaza jumătate din valoarea variabilei seed o singură dată și nu face nimic mai mult. Acesta cu siguranță nu este un PRNG.

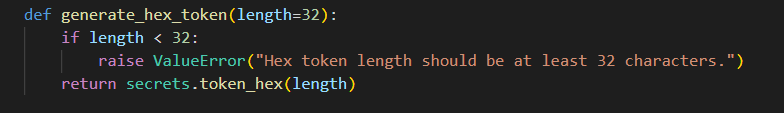
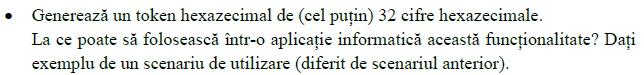
1. Secrets.py



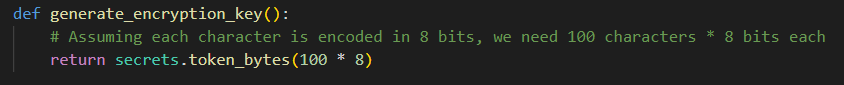
Această funcționalitate este utilă într-o aplicație informatică pentru înregistrarea sau resetarea parolelor utilizatorilor, asigurând că parolele sunt puternice și rezistente la atacuri de forță brută.

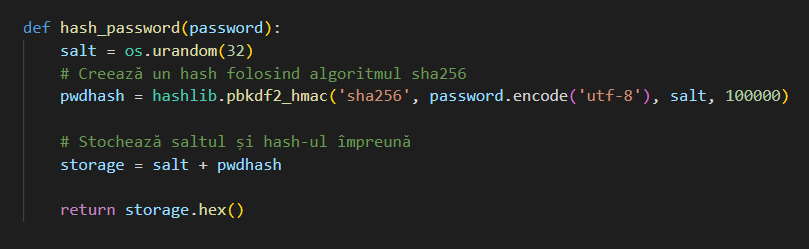


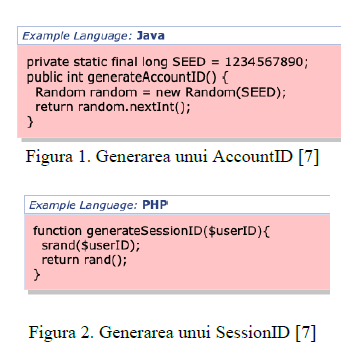
De exemplu, acesta poate fi folosit pentru a genera un token de resetare a parolei care să fie trimis utilizatorului prin email, astfel încât acesta să poată accesa un URL pentru a-și reseta parola.



Aceasta funcționalitate este folositoare în aplicații pentru a genera un identificator sau token sigur pentru sesiuni sau pentru validarea unor acțiuni (cum ar fi confirmarea unei înregistrări prin email).



Pentru stocarea în siguranță a parolelor, putem folosi modulul **hashlib** împreună cu **os** pentru a crea un hash al parolei înainte de a o stoca. Hashingul este un mod unidirecțional de a cripta informația, ceea ce înseamnă că nu putem recupera parola inițială din hash. De asemenea, putem utiliza un "salt" pentru a face hash-ul unic, chiar și pentru parole identice, pentru a îmbunătăți securitatea împotriva atacurilor de tip rainbow table.



1. CVE, CWE, CAPEC

CVE = Common Vulnerabilities and Exposures

CWE = Common Weakness Enumeration

CAPEC = Common Attack Pattern Enumeration and Classification



1. Java: Folosirea unui SEED static înseamnă că secvența de numere generate va fi aceeași de fiecare dată când codul este executat, făcând rezultatele predictibile și nesigure. Acest lucru se potrivește cu CWE-337, care descrie o vulnerabilitate în care un PRNG este inițializat dintr-un seed previzibil, cum ar fi ID-ul procesului sau timpul sistemului​​. De asemenea, CWE-336 menționează aceeași problemă sub numele de "Same Seed in Pseudo-Random Number Generator (PRNG)"​​.
2. PHP: Inițializarea rand() cu srand() folosind userID poate fi de asemenea previzibilă dacă userID este cunoscut sau ușor de ghicit. Acest lucru poate duce la generarea unor SessionID-uri care pot fi anticipate de un atacant. Această problemă este similară cu cea descrisă în CWE-335, care discută despre utilizarea greșită a seed-urilor într-un PRNG​

Dacă nu se folosește același seed de fiecare dată, dar spațiul seed-urilor posibile este mic, acest lucru poate reduce semnificativ numărul de posibile seed-uri pe care un atacant ar trebui să le testeze pentru a prezice ce numere aleatoare vor fi generate de PRNG. Acest scenariu este de asemenea descris în CWE-337



Da, în CAPEC 240, care corespunde CVE-2023-22047, există o mențiune la seed.

În descrierea atacului, se spune că acesta profită de faptul că seedul pentru generarea de numere aleatoare este reutilizat. Aceast seed este utilizat pentru a genera numere aleatoare care sunt utilizate pentru a genera token-uri de sesiune. Un atacator poate genera un token de sesiune valid prin trimiterea unei solicitări HTTP cu un parametru "seed" care este setat la aceleași valor ca și seed-ul utilizat de server.

Astfel, atacul poate fi descris și ca o reutilizare a seed-ului pentru generarea de numere aleatoare.

Iată citatul exact din CAPEC 240:

The vulnerability exists because the seed for the random number generator is reused. This seed is used to generate random numbers that are used to generate session tokens. An attacker can generate a valid session token by sending HTTP requests with a "seed" parameter set to the same value as the seed used by the server.

The attacker can use this session token to gain unauthorized access to the system.

1. **CWE-338**: Utilizarea unui PRNG criptografic slab într-un context de securitate. Când un PRNG necriptografic este folosit într-un context criptografic, poate expune criptografia la anumite tipuri de atacuri​​.
2. **CWE-337:** Inițializarea unui PRNG cu un seed previzibil, cum ar fi ID-ul procesului sau timpul sistemului. Utilizarea de seed-uri previzibile reduce semnificativ numărul de seed-uri posibile pe care un atacator ar trebui să le testeze pentru a prezice numerele aleatoare generate de PRNG​​.
3. **CWE-335**: Utilizarea incorectă a seed-urilor într-un PRNG. PRNG-urile sunt deterministe și, deși rezultatul lor pare aleatoriu, ele nu pot crea entropie​​.
4. **CWE-339**: Un PRNG folosește un spațiu de seed-uri relativ mic, ceea ce îl face mai susceptibil la atacuri de forță brută​​.

Am identificat 107 definite in acest an. Cateva exemple sunt:

1. CVE-2023-22047: Această vulnerabilitate permite unui atacator să genereze un token de sesiune valid pentru un sistem vulnerabil. Acest lucru ar putea permite atacatorului să obțină acces neautorizat la sistemul respectiv.
2. CVE-2023-22048: Această vulnerabilitate permite unui atacator să preia controlul unui sistem vulnerabil. Acest lucru ar putea permite atacatorului să efectueze acțiuni dăunătoare pe sistem, cum ar fi ștergerea datelor sau deschiderea sistemului la alte atacuri.
3. CVE-2023-22049: Această vulnerabilitate permite unui atacator să obțină informații confidențiale de la un sistem vulnerabil. Aceste informații ar putea fi utilizate pentru a comite alte atacuri sau pentru a fura identitatea victimelor.