Filip Radu-Cristian

342 C4

Mersenne Twister

Proiectul este o implementare a algoritmului non-criptografic PRNG Mersenne Twister. Acesta genereaza numere aleatoare si este folosit in PHP, Python, Ruby, etc.

Parametrii acestui algoritm sunt:

* *w*: word size (in number of bits)
* *n*: degree of recurrence
* *m*: middle word, an offset used in the recurrence relation defining the series ***x***, 1 ≤ *m* < *n*
* *r*: separation point of one word, or the number of bits of the lower bitmask, 0 ≤ *r* ≤ *w* - 1
* *a*: coefficients of the rational normal form twist matrix
* *b*, *c*: TGFSR(R) tempering bitmasks
* *s*, *t*: TGFSR(R) tempering bit shifts
* *u*, *d*, *l*: additional Mersenne Twister tempering bit shifts/masks

with the restriction that 2*nw*−*r* − 1

Sursa: <https://en.wikipedia.org/wiki/Mersenne_Twister>

Implementarea standard a MT este MT19937, cu o perioada foarte mare de 219937-1

Formula pe care se bazeaza MT este:



Coeficientii pentru MT-19937 sunt:

* (*w*, *n*, *m*, *r*) = (32, 624, 397, 31)
* *a* = 9908B0DF16
* (*u*, *d*) = (11, FFFFFFFF16)
* (*s*, *b*) = (7, 9D2C568016)
* (*t*, *c*) = (15, EFC6000016)
* *l* = 18

Pasi de implementare:

1. Initializare array MT folosind formula: *xi* = *f* × (*xi*−1 ⊕ (*xi*−1 >> (*w*−2))) + *i*
2. Functia twist: (MT[i] and upper\_mask) + (MT[(i+1) mod n] and lower\_mask). Tot in functia twist se calculeaza si starea urmatoare in functie de valoare LSB a lui x
3. ExtractNumber(): se face “tempering” pentru a compensa pentru dimensiunea redusa a echidistributiei si se returneaza ultimii 32 de biti.
4. Seed-ul folosit este 5489 ( <https://en.wikipedia.org/wiki/Mersenne_Twister>)

Vulnerabilitati:

Mersenne Twister nu este un algoritm sigur deoarece daca sunt aflate suficiente state-uri interne, un atacator poate genera toate starile viitoare ale algoritmului.

Atac pe Mersenne Twister:

Pasi Atac:

1.Inversare functiei de tempering:

Functia de tempering:

y ^= (y >> 11) # u = 11  
y ^= ((y << 7) & 2636928640) # b in baza 16  
y ^= ((y << 15) & 4022730752) # c in baza 16  
y ^= y >> 18 # l

2.Obtinerea a 624 de valori, inainte de reapelarea functiei twist. Aplicand untempering pe fiecare din aceste valori se va obtiunea state-ul intern al Mersenne Twister.

3. Stiind starea interna putem sa prezicem toate valorile pe care acesta le va genera in viitor, folosindu-ne de formula de generare a state-urilor viitoare

def predict(values):  
 for i in range(624):  
 x = (values[i] & upper\_mask) + (values[(i + 1) % 624] & lower\_mask)  
 xA = x >> 1  
 if x % 2 != 0:  
 xA = xA ^ 2567483615 # a in baza 16  
 values[i] = values[(i + 397) % 624] ^ xA  
 return values

Rulare:

Am obtinut urmatorii timpi de rulare:

Pentru:

* 100.000 de valori: 0.18s
* 1.000.000 de valori:1.85s
* 1.000.000 de valori:17.92s