Pseudocasualità con OpenSSL

Alfredo De Santis

Dipartimento di Informatica
Università di Salerno

ads@unisa.it

http://www.dia.unisa.it/professori/ads

Aprile 2017

Casualità nei Sistemi Operativi UNIX-Based: Cenni

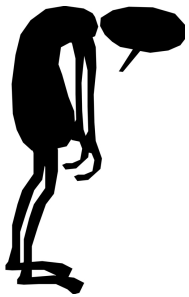
- Idea: in tutti i dispositivi di elaborazione sono presenti "fenomeni" o "eventi" che accadono costantemente e sono conseguenza di una certa interazione del computer verso utenti o altri computer
- Tali fenomeni rappresentano una sorta di *processo casuale*
- Esempio: istanti temporali associati alla digitazione dei tasti
  - Dopo aver digitato qualcosa, un utente potrebbe controllare se ci sono eventuali errori di digitazione, prendere un caffè, leggere un giornale, riflettere sulla migliore soluzione per un problema complesso, etc
- Altri esempi di processi che possono essere considerati casuali
  - Tempistica degli interrupt generati cliccando sui tasti del mouse
  - Movimenti del puntatore del mouse sullo schermo
  - Tempi associati agli eventi di I/O del disco o dei device driver
  - Informazioni inserite nei file di log (ad es., in /var/log/syslog, usato per registrare eventi di rete o di sicurezza)
  - Attività rilevata dai sensori presenti sui dispositivi
  - Etc

Casualità nei Sistemi Operativi UNIX-Based: Cenni

- L'output di tali processi rappresenta una sorgente di casualità e viene tipicamente memorizzato in una struttura chiamata *entropy pool*
- Le sorgenti di casualità possono essere divise in due categorie
  - Appartenenti al *kernel space*: generano random bit a partire da eventi che avvengono nel kernel space
  - Appartenenti allo *user space*: generano random bit a partire da eventi che avvengono nello user space
- I random bit forniti dalle sorgenti di casualità appartenenti al kernel space sono resi disponibili tramite un file speciale
  - /dev/random
- I random bit forniti dalle sorgenti di casualità appartenenti allo user space sono di solito resi disponibili tramite *Entropy Gathering Daemon (EGD)* o *Pseudo Random Number Generator Daemon (PRNGD)*

Casualità nei Sistemi Operativi UNIX-Based: Cenni

- Che succede se nel dispositivo di elaborazione non accadono eventi che coinvolgono l'entropy pool?
  - In /dev/random potrebbero non esserci sufficienti random bit
  - Una funzione che richiede l'accesso al file /dev/random può rimanere bloccata, finché in tale file non è presente il numero di random bit richiesto dalla funzione chiamante
- N.B. Di solito vengono generate solo poche centinaia di random bit al secondo
  - Se è necessario accedere ad una quantità di random bit che eccede questo tasso, non ci si può basare su /dev/random



Casualità nei Sistemi Operativi UNIX-Based: Cenni

- I sistemi operativi UNIX-Based risolvono questo problema mettendo a disposizione una sorgente non bloccante di casualità
  - Accessibile mediante il file /dev/urandom
  - Utilizza bit casuali forniti da /dev/random per inizializzare un *Cryptographically Secure Pseudo-Random Number Generator (CSPRNG)*
  - Produce flussi di bit di buona qualità crittografica

Generazione pseudocasuale in OpenSSL

- OpenSSL permette di generare byte pseudocasuali
  - Possono essere scritti sullo standard output o su un file
- La generazione avviene mediante *Deterministic Random Bit Generator (DRBG)*, definiti in SP 800-90
  - Hash DRBG
  - HMAC DRBG
  - CTR DRBG
- OpenSSL utilizza di default un CTR DRBG, basato su AES a 256 bit
- Il DRBG usa come seme i random byte forniti da /dev/urandom

openssl.com/testing/validation-2.0/docs/UserGuide-2.0-draft.odt
http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140sp/140sp1747.pdf
http://csrc.nist.gov/publications/nistpubs/800-90A/SP800-90A.pdf

Generazione pseudocasuale in OpenSSL

OpenSSL permette di generare byte pseudocasuali mediante il comando **rand**

Opzioni principali del comando rand

```
openssl rand [options] [numbyte]
```

- **options**
  - **-rand file(s)** File usati come seme per il generatore
  - **-out file** File su cui scrivere i dati generati, che altrimenti verrebbero scritti sullo standard output
  - **-base64** I dati generati sono codificati in formato base64
  - **-hex** I dati generati sono codificati in formato esadecimale
- **numbyte** Numero di byte che si intende generare

Generazione pseudocasuale in OpenSSL

OpenSSL permette di generare byte pseudocasuali mediante il comando **rand**

Opzioni principali del comando rand

```
openssl rand [options] [numbyte]
```

- **options**
  - **-rand file(s)** Fil come seme per il generatore
  - **-out file** Fi ne altrimenti verrebbero scritti sullo standard
  - **-base64** I da to base64
  - **-hex** I dati ge esadecimale
- **numbyte** Numero di byte che si intende generare

Per ottenere la lista completa delle opzioni del comando **rand** è possibile utilizzare **man rand**

Generazione pseudocasuale in OpenSSL: Esempio

Nel seguente esempio vengono generati 12 byte pseudocasuali e scritti nel file **pseudorandom-data.bin**

```
openssl rand -out pseudorandom-data.bin 12
```

Mediante il seguente comando è possibile visualizzare il contenuto del file **pseudorandom-data.bin**

```
$ xxd pseudorandom-data.bin
0000000: ac36 111b 4b0a c4d2 01b2 4ae0             .6..K.....J.
```

Mediante il seguente comando è possibile visualizzare, in formato binario, il contenuto del file **pseudorandom-data.bin**

```
$ xxd -b -g 8 -c 8 pseudorandom-data.bin | cut -d " " -f 2
10101100000110110000100010001101101001011000010101100010011
01001000000001101100100100101011100000
```

Generazione pseudocasuale in OpenSSL: Esempio

Mediante il seguente comando è possibile generare 6 random byte e codificarli in base 64

- Questo produrrà una stringa di 8 caratteri

```
openssl rand -base64 6
```

```
PHRu0aeI
```

Mediante il seguente comando è possibile generare 12 random byte e codificarli in base 64

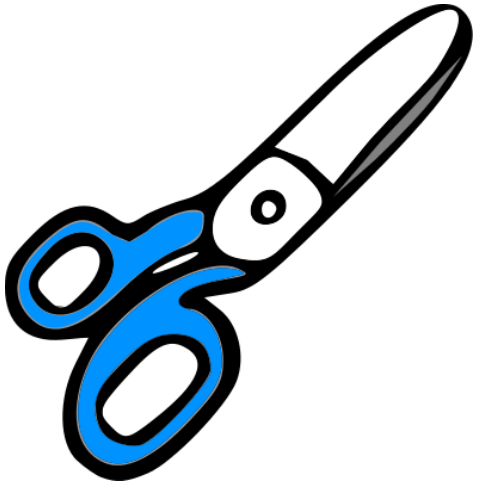
- Questo produrrà una stringa di 16 caratteri

```
openssl rand -base64 12
```

```
ttGNyw0twfYQZtLf
```

Generazione pseudocasuale in OpenSSL: Esempio

- **Osservazioni**
  - Quando i dati sono convertiti in Base64, la stringa di output prodotta da tale codifica avrà sempre una lunghezza multipla di quattro
  - Se si desidera una stringa pseudocasuale la cui lunghezza non sia multipla di quattro, è necessario "accorciare" tale stringa mediante altri strumenti



Generazione pseudocasuale in OpenSSL: Esempio

Mediante il seguente comando è possibile generare una stringa pseudocasuale composta da 39 caratteri stampabili

```
openssl rand -base64 39 | cut -c1-39
```

```
MigL10P23/0OIFsiOhXH07jYYOgiG+Ud6k/kqix
```

Nell'esempio seguente, i dati generati da OpenSSL sono passati in pipeline alla funzione SHA-1

- L'output di tale funzione è "accorciato" per ottenere il numero di caratteri richiesto
- Il risultato è salvato nel file **random.txt**

```
openssl rand -base64 39 |shasum |head -c39 > random.txt
```

Generazione pseudocasuale in OpenSSL: Esempio

Mediante il seguente comando è possibile generare una stringa pseudocasuale composta da 39 caratteri stampabili

```
openssl rand -base64 39 | cut -c1-39
```

```
MigL10P23/0OIFsiOhXH07jYYOgiG+Ud
```

Per ottenere la lista completa delle opzioni del comando **cut** è possibile utilizzare **man cut**

Per ottenere la lista completa delle opzioni del comando **head** è possibile utilizzare **man head**

Nell'esempio [illegible] da OpenSS[illegible] pipeline alla [illegible]

- L'output di [illegible] to" per ottenere il numero di caratteri ric[illegible]
- Il risultato è salvato nel file **ran[illegible]m.txt**

```
openssl rand -base64 39 |shasum |head -c39 > random.txt
```

Generazione pseudocasuale in OpenSSL: Esempio

Nel seguente esempio, utilizzando il contenuto del file **.bash_history** come seme per il comando **rand**, vengono generati 128 byte random, codificati in Base64

```
openssl rand -rand .bash_history -base64 128
```

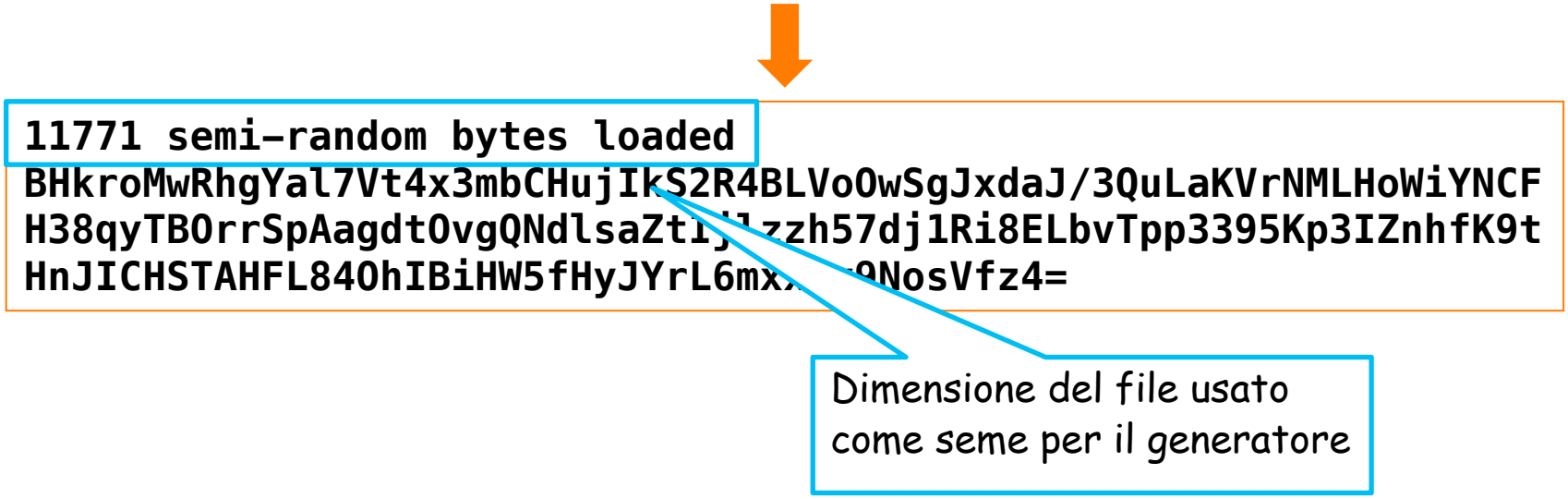
```
11771 semi-random bytes loaded
BHkroMwRhgYal7Vt4x3mbCHujIkS2R4BLVoOwSgJxdaJ/3QuLaKVrNMLHoWiYNCF
H38qyTBOrrSpAagdtOvgQNdlsaZtIjlzzh57dj1Ri8ELbvTpp3395Kp3IZnhfK9t
HnJICHSTAHFL84OhIBiHW5fHyJYrL6mxxmv9NosVfz4=
```

N.B. Di default è utilizzato il file **/dev/urandom** come seme

Generazione pseudocasuale in OpenSSL: Esempio

Nel seguente esempio, utilizzando il contenuto del file **.bash_history** come seme per il comando **rand**, vengono generati 128 byte random, codificati in Base64

```
openssl rand -rand .bash_history -base64 128
```



N.B. Di default è utilizzato il file **/dev/urandom** come seme

Generazione pseudocasuale in OpenSSL: Esempio

Nel seguente esempio, utilizzando il contenuto del file **.bash_history** come seme per il comando **rand**, vengono generati 128 byte random, codificati in Base64

```
openssl rand -rand .bash_history -base64 128
```

```
11771 semi-random bytes loaded
BHkroMwRhgYal7Vt4x3mbCHujIkS2R4BLVoOwSgJxdaJ/3QuLaKVrNMLHoWiYNCF
H38qyTBOrrSpAagdtOvgQNdlsaZtIjlzzh57dj1Ri8ELbvTpp3395Kp3IZnhfK9t
HnJICHSTAHFL84OhIBiHW5fHyJYrL6mxxmv9NosVfz4=
```

171 caratteri + 1 di padding

N.B. Di default è utilizzato il file **/dev/urandom** come seme

Bibliografia

- Documentazione su OpenSSL
  - https://www.openssl.org/docs/

OpenSSL™
Cryptography and SSL/TLS Toolkit

Domande?