

# secure programming in C

# fonti di input convenzionali e atipiche

- standard input
- parametri sulla linea di comando
  - nome dell'eseguibile: `argv[0]`
- variabili di ambiente
- form di input nei programmi interattivi
- link dinamico di librerie
- directory corrente del processo
- files
- inter process communication, ecc...
- rete (socket)

# buffer overflow: funzioni standard

- alcune funzioni standard che possono provocare buffer overflow sono
  - strcpy()
  - strcat()
  - sprintf()
  - gets()
  - scanf() (%s)
- vedi sezione 3 dei man unix

# buffer overflow: funzioni standard

- versioni “sicure”, leggermente più complesse da usare
  - `strncpy()`
    - attenzione non sempre inserisce il byte zero alla fine
  - `strncat()`
  - `snprintf()`
    - attenzione su vecchi sistemi può essere identico a `sprintf()`!
    - vedi i vecchi linux con Linux libc4
  - `fgets()`
  - `scanf()` “%<max>s”
    - attenzione al byte zero di fine stringa!

# buffer overflow: funzioni standard

- `realpath()`
- `getopt()`
- `getpass()`
- `getwd()`
  
- di queste non esistono versioni standard sicure

# librerie dinamiche

- il link con librerie dinamiche può essere una vulnerabilità
  - alcuni sistemi permettono all'utente di sostituire le librerie dinamiche
  - comandi privilegiati possono linkare librerie scelte dall'utente
- linux
  - permette di customizzare il path di ricerca
  - LD\_LIBRARY\_PATH
  - ma non permette customizzazione per programmi “privilegiati” (suid, guid)
- windows
  - cerca nella directory corrente le .dll e poi nelle directory di sistema
  - <http://www.securityfocus.com/bid/1699/>

# riferimenti

- David A. Wheeler, “Secure Programming for Linux and Unix HOWTO -- Creating Secure Software”
  - <http://www.dwheeler.com/secure-programs/>