



# Introducción al cracking en GNU/Linux

**Pablo Garaizar Sagarminaga**  
**[garaizar@eside.deusto.es](mailto:garaizar@eside.deusto.es)**

---



# Índice

- **¿Por qué crackear en GNU/Linux?**
  - **Herramientas típicas.**
  - **¿A qué nos enfrentamos?**
    - **Serials / passwords.**
    - **Discos llave.**
    - **Binarios empaquetados, ofuscados, cifrados.**
-



## ¿Por qué crackear en GNU/Linux?

- También hay software privativo para GNU/Linux (juegos, programas, drivers).
  - Exploits y herramientas de ataque cifradas: auditoria.
  - Just for fun!
-



## Herramientas típicas

- **Volcadores: od, hexdump, objdump, ELFdump...**
  - **Visualizadores: elfs, KBiew, itsELF...**
  - **Editores hexadecimales: hte, BIEW, ELFe...**
  - **Depuradores: ltrace, strace, gdb, fenris, linICE...**
-



## ¿A qué nos enfrentamos?

- **Comprobaciones de contraseña/nº de serie.**
  - **Discos/ficheros llave.**
  - **Binarios:**
    - **empaquetados (UPX, etc.).**
    - **ofuscados (objobjf, etc.).**
    - **cifrados (burneye, shiva, cryptexec...).**
-



# Números de serie

- **Comprobación de un número serie o palabra durante el proceso de instalación o ejecución.**
  - **Será más difícil crackearla:**
    - **Cuanto más separados en el espacio y el tiempo [sheroc, MenorHack2005] estén:**
      - La lectura del número de serie.
      - Su comprobación.
      - La adquisición de los privilegios otorgados por un número de serie válido.
    - **Cuanto más ofuscado sea el código de comprobación (lenguajes de muy alto nivel generan código difícil de crackear y viceversa, normalmente).**
-



## Números de serie, demos

- **Crack-it en ensamblador con una comprobación.**
  - **Crack-it en C con una comprobación.**
  - **Crack-it en C++ con una comprobación.**
-



## Discos llave

- Normalmente se comprueba un fichero binario con un determinado formato, tanto en la carga como durante la ejecución del programa.
  - Se aplican los mismos criterios de dificultad que en el caso anterior.
  - Puede complicarse si el disco llave no tiene un formato estándar.
-





## **Discos llave, demos**

- **Crack-it en C con fichero llave.**
  - **Crack-it en C con disco llave RAW.**
-



## Binarios complejos

- **En ocasiones el binario está empaquetado, ofuscado o cifrado.**
  - **El paso clave es detectar qué herramienta o técnica se ha empleado:**
    - **Herramientas estándar: file, strings, etc.**
    - **Conocer los "stubs" de los principales cifradores.**
    - **Una vez identificado, tratar de reducir la complejidad al máximo.**
-



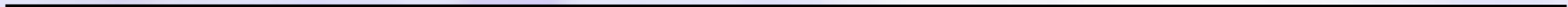
# Binarios complejos, demos

- **Binarios comprimidos con UPX.**
  - **Binarios ofuscados con objobf.**
  - **Binarios cifrados con burneye.**
-



## Demo final

- **Binario protegido con contraseña y múltiples chequeos.**
  - **Elaboración de keygen.**





## Referencias

- **Cracking y anticracking, la eterna batalla en el espacio-tiempo, sheroc, MenorHack 2005.**
  - **Técnicas anti-debugging, Iñaki López.**
  - **Burneye y objobjf, TESO Team.**
  - **Homepages de GDB, BIEW, binutils, NASM, ELFsh, UPX, etc.**
-