II Jornadas sobre Software Libre y Seguridad en GNU/Linux

Programación Segura

Chema Peribáñez <chema@augcyl.org>

Problemas de seguridad

ausencia infraestructura seguridad

errores administrador/usuario

exploits

para evitar exploits, hay que programar bien

- entender cómo funcionan las cosas
- aprender buenas políticas
- aprender trampas habituales

Por qué lo hacemos tan mal

hay fase de prueba para errores

- se ve necesario
- automatizable (dejagnu, jakarta)
- comprobable
- se prueba lo usual

la seguridad se audita

- poca sensibilidad
- sí hay programas para detectar errores
- difícil de controlar (efecto tardío)
- caso rebuscado

"recetario" complicado

- dependiente sistema operativo y lenguaje

Qué hace un programa inseguro

un usuario provoca comportamiento imprevisto

¿qué obtiene?

- denegación de servicio
- intrusión (privilegiada o no)
- información

¿cómo?

- interactua remotamente (http)
  - la seguridad, siempre en el servidor
- ejecuta localmente setuid
  - línea comandos
  - variables de entorno
- ficheros en directorio de escritura
  - comprobar si es enlace
  - ojo condiciones de carrera

Algunas claves

setuid nunca en shell, sólo en C, Perl, etc.

soltar privilegios cuanto antes, si se puede, definitivamente

cuidado al llamar

- extensión RSBAC puntos críticos exec y seteuid

aprovechar separación de procesos (que no hilos)

- postfix
- X-Window
  - "teclado seguro"

entrada, línea de comandos, entorno, no fiable

- Perl, extension de seguridad núcleo
- secuencias de escape

desbordamiento de buffer

No reinventar la rueda

PAM (ahora también en Java)

- módulos autentificación
- módulos de cuenta (decidir si se le deja)
- módulos de cambio de claves
- módulos de sesion

stunnel

- cliente/servidor SSL
- con certificados