Curso: (62612) Diseño de aplicaciones seguras

Fernando Tricas García

Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza
http://webdiis.unizar.es/~ftricas/
http://moodle.unizar.es/
ftricas@unizar.es

Tema III ¿Qué hacemos con el código fuente?

Fernando Tricas García

Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza
http://webdiis.unizar.es/~ftricas/
http://moodle.unizar.es/
ftricas@unizar.es

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

¿Código fuente sí o no?

Seguridad por la oscuridad

- No es necesario tener el código (fuente ni ejecutable) para encontrar vulnerabilidades.
  - Secretos almacenados en cliente pueden ser vistos: utilizar criptografía fuerte
  - II Guerra Mundial: cifrado de *Enigma* fue roto, observando los mensajes codificados.
- Fallo = indicio de alguna vulnerabilidad
- Atacante no pasivo (p. ej., entradas largas)

Ingeniería inversa

- El código binario es suficiente en muchos casos
- Se puede leer el binario directamente, pero hay herramientas...
- Los desensambladores son muy potentes (herramientas de depuración)
- También existen 'descompiladores' (*decompilers*)
- El código de máquinas de más alto nivel es mas fácil de manejar (JVM, .Net)

Ofuscación del código (*code obfuscation*)

Almacenar secretos en el código no es una buena idea

- A veces no hay más remedio
- Transformar el código
- Por ejemplo, cambiar el nombre de las variables (en Java)
- No existen muy buenas técnicas de transformación

La oscuridad no es la panacea

- Mostrar el código puede ayudar en la seguridad (siempre que alguien se moleste en mirarlo)
  - Es muy fácil que nadie llegue a mirarlo, sobre todo si es comercial
- La decisión de proporcionar el código o no debería basarse en otras consideraciones, no en la seguridad

Mitos del código de fuente abierta

- Muchos ojos mirando: poder cambiar cosas no significa que se vayan a cambiar (o mirar) las adecuadas. La mayoría de desarrolladores ni siquiera saben/se preocupan de estas cosas.
- Detectar vulnerabilidades es difícil: no sólo hay que saber qué buscar, sino también cuándo eso es un problema
- Hay herramientas (pero nuevamente, hace falta entender su salida)

Elevación de privilegios en el kernel Linux desde el año 2001

`http://www.hispasec.com/unaaldia/3949` "16/08/2009"

Algoritmos criptográficos

- Es esencial la publicación de una algoritmo para que sea revisado y validado
  - Fortaleza: propiedades matemáticas, no secreto
  - Revisión externa: complicado y fácil equivocarse

  Pero ...
- Programas seguros $\neq$ criptografía
- Siempre: más importante el conocimiento del tema y la diversidad de experiencias

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

¿Entonces?

- Notificar a los usuarios tan pronto como se detecten problemas y se ofrezca la solución.
- Es una buena idea tener un mecanismo de notificación exclusivamente dedicado.
- Que sea fácil saber que existe y de utilizar
- Ofrecer el código del programa puede ser una buena opción, si realmente lo van a mirar (pero mejor primero 'en casa' y, si puede ser, expertos)