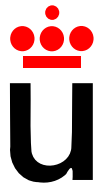
Desarrollo de programas seguros

Seguridad en Redes de Ordenadores

Enrique Soriano

LS, GSYC

6 de marzo de 2018



Universidad
Rey Juan Carlos

Máximas

- **All input is evil!**
- **Aprende bien el lenguaje y programa con cuidado**

Ya deberíamos saber (I)

- Manejar los errores de las llamadas al sistema y funciones de biblioteca.
- No desbordar buffers.
- No desbordar enteros.
- Eliminar condiciones de carrera.

Ya deberíamos saber (II)

- Eliminar deadlocks.
- No imprimir mensajes de error que destapen vulnerabilidades.
- Evitar leaks.
- Probar el software (tests unitarios, tests de regresión, etc.).
- ...

Ejemplos reales: Apple TLS/SSL

```
hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
hashOut.length = SSL_SHA1_DIGEST_LEN;
if ((err = SSLFreeBuffer(&hashCtx)) != 0)
    goto fail;
if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
    goto fail;  // bazinga!
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;

err = sslRawVerify(...);
```

Efecto: nunca se comprueba la firma.

Ejemplos reales: Nintendo Wii

```
int verify_tmd (struct signed_tmd stmd) {
char decrypted_sig[256] =
    RSA_DecryptSig(CA_public_key, stmd.cert.rsa_signature);
char sig_hash = decrypted_sig[256-SHA1LEN:256];
char payload_hash[SHA1LEN] = SHA1(stmd.tmd);

if (strncmp(payload_hash, sig_hash, SHA1LEN) == 0) // bazinga!
{
    return SIG_OK;
} else {
    return SIG_BAD;
}
```

Efecto: Las hashes que comienza por 0x00 (o que tienen el mismo prefijo hasta el primer 0x00) se consideran iguales.

Ejemplos reales: OpenSSL TLS/DTLS Heartbleed

- El heartbeat de SSL permite mantener la conexión viva.
- Se envía una string, y el mensaje de respuesta tiene que incluir la misma string. La string se envía junto con su tamaño.
- Una petición mal formada con un tamaño muy superior, contesta con toda la memoria desde donde el servidor ha almacenado la string hasta el tamaño.
- Permite extraer hasta 64Kb de la memoria del servidor que está usando SSL: claves privadas, passwords, etc.

Credenciales

Un proceso puede pasar a ejecutar a nombre de otro usuario:

- EUID: UID efectivo. Puede que $UID \neq EUID$
- EGID: GID efectivo. Puede que $GID \neq EGID$
- Es útil si el programa necesita ciertos privilegios de forma transitoria (e.g., cambiar la contraseña).

Credenciales

- *Setuid bit*: el fichero se ejecuta con EUID puesto al dueño del fichero y EGID puesto al grupo del fichero.
- Una vulnerabilidad en un programa con el *Setuid bit* es muy peligrosa.

```
-rwsr-xr-x 1 root shadow 27922 jun 12 12:43 /usr/bin/passwd
```

Credenciales

Mínimo privilegio: "Hay que ser siempre lo más restrictivo que se pueda a la hora de escoger el UID y GID"

Credenciales

"Hay que restablecer el UID y GID efectivo antes de realizar una llamada al sistema exec'

Credenciales

"Debemos cerrar los descriptores abiertos que no sean necesarios antes de llamar a exec"

Truco: usar el bit de apertura `CLOSEONEXEC` si está disponible en el sistema.

Credenciales

> *"Se deben hacer únicamente suposiciones seguras para el manejo y recuperación de errores"*

P. ej. una versión antigua de `su`, si no podía abrir `/etc/passwd`, escalaba a root sin contraseña porque suponía que el sistema estaba en una situación crítica.

Buffers

Siempre hay que usar la versión segura de las funciones:

- `strcpy` → `strncpy`*
- `strcat` → `strncat`
- `sprint` → `snprint`
- `gets` → `fgets`
- ...

* Ojo! si se sobrepasa el tamaño con `strncpy`, NO se pone el terminador de string.

Buffers

"Se debe tener mucho cuidado de limitar la copia de un buffer con el tamaño del destino y no con el tamaño del origen."

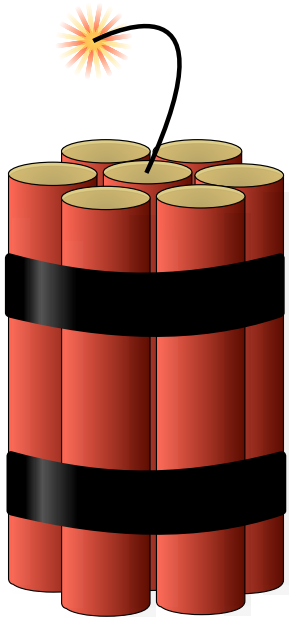
Strings de formato

Usar mal las funciones para escribir la salida es más peligroso de lo que pensamos. Ejemplo: `printf` y cia.

- Si el atacante tiene control sobre la cadena de formato que se pasa a `printf`, tenemos una vulnerabilidad **muy peligrosa**.
- Se puede usar para inspeccionar la memoria e incluso para escribir una dirección de memoria.
- `%n` escribe en el entero que se le pasa como argumento el número de caracteres escritos hasta ahora en esta llamada a `printf` $\rightarrow$ se puede usar para secuestrar el flujo de ejecución. P. ej: se puede usar para escribir el PC de retorno saltándonos los canarios de la pila.

Strings de formato

DEMO: examinando la pila con printf



Strings de formato

"Debemos tener cuidado al escribir la salida del programa."

Inyección de comandos

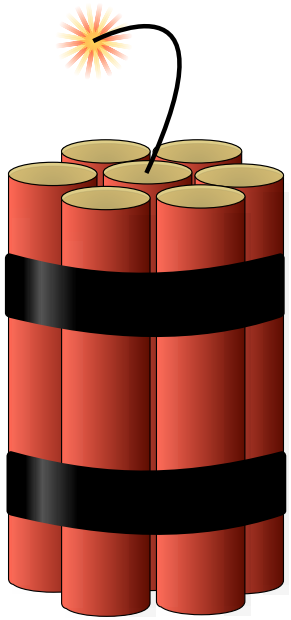
Problema:

- Usamos los datos de la entrada como argumento para un comando.
- Mallory introduce datos de entrada tienen un *significado especial* para el comando.
- Así se puede modificar el comportamiento del comando, y/o ejecutar nuevos comandos.

Inyección de comandos

- Las *inyecciones SQL* son muy *populares*.
- SQL: lenguaje de consultas para bases de datos.
- Si no tienes cuidado...

DEMO: SQL injection



Inyección de comandos

"Debemos escapar los caracteres con significado especial en la entrada de nuestros programas."

Inyección de comandos

- Esto puede pasar con cualquier tipo de comando.
- Hay que tener mucho cuidado con la llamada al sistema `system` de UNIX.

Inyección de comandos

Siempre debemos validar la entrada:

- Comprobar los tipos de datos.
- Comprobar los rangos válidos.
- Comprobar que la entrada sólo contiene caracteres del conjunto permitido (e.g. sólo caracteres alfanuméricos).
- Comprobar que la entrada tiene el formato esperado (e.g. con una expresión regular).

Contaminación

- Una inyección puede *contaminar* las variables de entorno.
- Ejemplo: `$LD_PRELOAD`
- Ejemplo: Cadena de conexión a DB.
- Ejemplo: Contaminamos `$PATH` para explotar un execlp(), execvp(), o execvpe().

Aislamiento

Los servicios peligrosos deberían ejecutar aislados del resto en un *sandbox*.

- *Applets*.
- *Chroot*.
- *FreeBSD Jails*.
- *VM*
- *Containers*

Protegiendo la memoria

Si la memoria de tu proceso tiene información sensible:

- No debe ir a almacenamiento secundario (*swap*). P. ej: `mlock`.
- No debe generar *core dumps*.
- Debemos borrar los datos sensibles cuando ya no sean necesarios.
- Desactivar la depuración para los programas críticos (e.g. acceso a la memoria). Por ejemplo en linux se depura con `ptrace`:
  `cat /proc/sys/kernel/yama/ptrace_scope`
  - 0: todos los procesos pueden ser depurados si corren con el mismo UID.
  - 1: sólo se puede depurar el proceso padre
  - 2: sólo root puede usar ptrace
  - 3: ningún proceso se puede depurar con ptrace.

Análisis de código

- Estático: analiza sin ejecutar el código.
  - Cubre todo el código, no sólo el que se ejecuta en los test.
  - Pueden analizar el código fuente o el binario (lo primero es más habitual).
  - Detectan el uso de funciones inseguras, desbordamiento de arrays, errores de formato, etc.
  - Ejemplos: Adlint, FlawFinder, BOON, ...
- Dinámico: analiza ejecutando el código.
  - Facilita detectar ciertos comportamientos erróneos o maliciosos.
  - Son más precisos que los estáticos (menos falsos positivos).
  - Intentan detectar desbordamientos, leaks, condiciones de carrera, crear grafos de llamadas, *profiling* del heap, verificar el procesado de la entrada (fuzzing), etc.
  - Ejemplos: Valgrind, DynamoRIO, Purify, FIST, STOBO, ...
- Se pueden aplicar las dos aproximaciones (p. ej. IBM Security AppScan, Veracode, etc.).

Discreción

- Al programar funciones criptográficas, o al manejar información confidencial, hay que evitar dar pistas para evitar ataques de *side channel*.
- Ejemplo: cuanto más parecidos son dos buffers, más tardas en compararlos.
- Ejemplo: el algoritmo *square-and-multiply* usado para exponenciación modular depende linealmente del número de bits a 1 en la clave: no se debe usar en RSA.

Discreción

Por ejemplo, el paquete *subtle* de Go: `constant_time.go`:

- `func ConstantTimeByteEq(x, y uint8) int`
- `func ConstantTimeCompare(x, y []byte) int`
- `func ConstantTimeCopy(v int, x, y []byte)`
- `func ConstantTimeEq(x, y int32) int`
- `func ConstantTimeSelect(v, x, y int) int`