Sistemas Operativos

Departamento de Computación – FCEyN – UBA

Segundo cuatrimestre de 2024

Nombre y apellido: _____

Nº orden: _____ L.U.: _____ Cant. hojas: _____

1	2	3	Nota

Segundo parcial – 17/11 - Segundo cuatrimestre de 2022

- Numere las hojas entregadas. Complete en la primera hoja la cantidad total de hojas entregadas.
- Entregue esta hoja junto al examen, la misma **no** se incluye en la cantidad total de hojas entregadas.
- Cada ejercicio debe realizarse en hojas separadas y numeradas. Debe identificarse cada hoja con nombre, apellido, LU y número de orden.
- **Cada código o pseudocódigo debe estar bien explicado y justificado en castellano. ¡Obligatorio!**
- Toda suposición o decisión que tome deberá justificarla adecuadamente. Si la misma no es correcta o no se condice con el enunciado no será tomada como válida y será corregida acorde.
- La devolución de los exámenes corregidos es personal. Los pedidos de revisión se realizarán por escrito, antes de retirar el examen corregido del aula.
- Los parciales tienen tres notas: I (Insuficiente): 0 a 59 pts, A- (Aprobado condicional): 60 a 64 pts y A (Aprobado): 65 a 100 pts. No se puede aprobar con A- ambos parciales. Los recuperatorios tienen dos notas: I: 0 a 64 pts y A: 65 a 100 pts.

Ejercicio 1. Se cuenta con un código el cual verifica si una clave es válida, permitiendo acceder al sistema con privilegios de administrador en caso afirmativo. Para el siguiente código se pide:

a) Explicar qué vulnerabilidad tiene, y cómo serían los valores exactos de cada byte del input de un *exploit* que permita realizar una ejecución de código arbitrario con escalamiento de privilegios. Aclarar las funciones utilizadas y justificar con un diagrama de las posiciones de memoria afectadas que incluya sus significados y sus valores antes y después de aplicar el *exploit*. ¿Cómo generaría dicho input, y cómo ejecutaría el programa en cuestión? (18p)

b) Proponer una corrección para el código en C que solucione la vulnerabilidad, detallando qué líneas modificaría, y por qué. Proponga además dos formas adicionales de mitigar la vulnerabilidad que no requieran modificar el código, explicando detalladamente su funcionamiento, y mostrando cómo dificultarían el ataque. (7p)

```
extern bool clave_es_valida(const char* clave);
extern void acceder_al_sistema();

void convertir_a_minuscula(char* buffer) {
  do {
    *buffer = tolower(*buffer);
  } while (*(buffer++) != '\0');
}

char* copiar_en_minuscula(const char* str) {
  char buffer[16];
  strcpy(buffer, str);
  convertir_a_minuscula(buffer);
  return strdup(buffer);
}

int main(int argc, const char* argv[]) {
  char* clave = copiar_en_minuscula(argv[1]);
  if (clave_es_valida(clave)) {
    acceder_al_sistema();
  }
  return 0;
}
```

Ayuda 1: Asuma que `clave_es_valida()` y `acceder_al_sistema()` son funciones seguras.

Ayuda 2:

```
Dump of assembler code for function main:
   0x0000555555555080: sub    rsp,0x8
   0x0000555555555084: mov    rdi,QWORD PTR [rsi+0x8]
   0x0000555555555088: call   0x5555555551d0 <copiar_en_minuscula>
   0x000055555555508d: mov    rdi,rax
   0x0000555555555090: call   0x555555555210 <clave_es_valida>
   0x0000555555555095: test   al,al
   0x0000555555555097: jne    0x5555555550a0 <main+32>
   0x0000555555555099: xor    eax,eax
   0x000055555555509b: add    rsp,0x8
   0x000055555555509f: ret
   0x00005555555550a0: xor    eax,eax
   0x00005555555550a2: call   0x555555555220 <acceder_al_sistema>
   0x00005555555550a7: jmp    0x555555555099 <main+25>
```

Ejercicio 2. Seguridad: (25 puntos)
Dado el código a continuación:

```
1  void imprimir_habilitado(const char *nombre_usuario, const char* clave, const char * imprimir, int tam_imprimir) {
2      char *cmd = malloc(tam_imprimir+5 * sizeof(char));
3      if (cmd == NULL) exit(1);
4      if (usuario_habilitado("/etc/shadow", nombre_usuario, clave)) {
5          snprintf(cmd, tam_imprimir+4, "echo %s", imprimir);
6          system(cmd);
7      } else {
8          printf("El usuario o clave indicados son incorrectos.");
9          assert(-1);
10 }
```

El objetivo de la función es imprimir por pantalla el texto enviado como parámetro por el usuario, siempre y cuando el nombre de usuario y clave provistos por dicho usuario sean correctos.

Para esto se cuenta con la función `usuario_habilitado` que se sabe funciona correctamente y no cuenta con problemas de seguridad. La misma utiliza strings terminados en caracter nulo (`\0`) y lee el archivo provisto de contraseñas (encriptadas) de todos los usuarios del sistema, que puede ser sólo leído por `root`, devolviendo un booleano indicando si el usuario y la clave ingresados se encuentran en dicho archivo.

a) Indique si es necesario que el programa corra con algún nivel específico de permisos. Justifique en qué líneas y porqué.
b) Indique dos problemas de seguridad que podrían surgir (hint: tenga en cuenta el ítem anterior).
c) Indique alguna manera (valuación de los parámetros) de poder explotar cada una de las vulnerabilidades mencionadas.
d) Indique el impacto de las vulnerabilidades mencionadas, analizándolas según los tres requisitos fundamentales de la seguridad de la información.
e) Para cada vulnerabilidad, proponga una solución, indicando cómo modificaría el código en caso de corresponder.