Práctica: Explotación mediante buffer overflows

El objetivo de esta práctica **individual** es aplicar técnicas de explotación de vulnerabilidades, centrándose en el aprovechamiento de un desbordamiento de búfer local para lograr la ejecución de código arbitrario en un programa vulnerable (ver final del documento). Se aprenderá a diseñar e implementar un exploit funcional, así como a analizar aspectos relacionados con los permisos efectivos (SUID) de los binarios explotados. La práctica se divide en dos partes:

- **Parte 1 (0,75 puntos) – Diseño de un exploit para ejecutar una shell local:** Se proporciona un programa vulnerable con una vulnerabilidad de tipo *stack buffer overflow*. El objetivo es diseñar e implementar un exploit que permita ejecutar una shell local mediante la inyección de una *shellcode*. Las tareas a realizar son:
  - Analizar el código vulnerable y su comportamiento.
  - Calcular el desplazamiento correcto para provocar el desbordamiento.
  - Desarrollar una *payload* que permita la ejecución de `/bin/sh`.
  - Validar la explotación sin interferencia de mecanismos como ASLR o NX (deshabilitados para esta parte).
- **Parte 2 (0,75 puntos) – Explotación con SUID y escalada de privilegios:** En esta fase se analizará el funcionamiento de los binarios con el bit SUID activado. El binario vulnerable se ejecutará con permisos elevados (perteneciente a otro usuario o a `root`). El objetivo será modificar la *shellcode* para que, al ser explotado el binario, se obtenga una shell con los permisos efectivos del programa vulnerable. Las tareas son:
  - Entender cómo funciona el bit SUID y de qué manera permite la herencia de privilegios durante la ejecución de procesos.
  - Adaptar la *shellcode* para que herede y mantenga los privilegios elevados (por ejemplo, utilizando la syscall `setuid(geteuid())`).
  - Validar que, tras la explotación, la shell obtenida tiene los permisos del propietario del binario (verificable mediante `id` o `whoami`).

Entregables en Egela:

- Código fuente del exploit o exploits desarrollados.
- Explicación detallada del funcionamiento de la *shellcode* utilizada en cada parte.
- Informe breve con los pasos realizados y las técnicas aplicadas.
- La fecha de entrega es el **31 de marzo a las 00:00**.

Nota: Aunque el escenario planteado en esta práctica no representa un entorno realista —ya que se desactivan mecanismos de protección como ASLR, NX o canarios—, permite comprender en detalle el proceso de creación de un exploit desde cero. En particular, es importante entender como estructurar un *payload*, cómo controlar el flujo de ejecución tras un desbordamiento de búfer, y cómo integrar una *shellcode* funcional.

```
#include <stdio.h>
#include <string.h>

void function(char *input) {
    char buffer[64];
    strcpy(buffer, input);
}

int main(int argc, char *argv[]) {
    if (argc > 1) {
        function(argv[1]);
    }
    else {
        printf("Usage: %s <input>\n", argv[0]);
    }
    return(0);
}
```