Práctica. CWE-767: Access to Critical Private Variable via Public Method

Instrucciones. Elaborar una aplicación de consola en C que permita ejemplificar la vulnerabilidad de Acceso a variable privada crítica a través del método público en equipos Linux.

1. Sigue las instrucciones de la práctica en la siguiente página.
2. Crea un reporte de práctica en un archivo de Word con una portada con tu nombre. Guárdalo con el nombre NombreAlumno\_CWE-767.docx.
3. Coloca las siguientes capturas de pantalla, cada una con una descripción textual arriba de la captura.
   1. Coloca la captura de pantalla de tu código fuente de C en Visual Studio Code.
   2. Coloca la captura de pantalla de tu código fuente de Python en Visual Studio Code.
   3. Coloca la captura de pantalla de la salida de tu programa con el payload correcto para imprimir el resultado de una variable privada:

Password correcto. Bienvenido!

* 1. Publica tu código fuente en un proyecto público de GitHub. Coloca la URL del código fuente publicado en GitHub.

1. Sube tu reporte en un archivo Word a la actividad en Eminus. Prerrequisitos
2. Esta práctica tiene como prerrequisito haber realizado la práctica de instalación de servidor Linux. *Puede seguir las instrucciones de cómo instalar Ubuntu Server desde la práctica* [*Instalación de*](https://www.uv.mx/personal/gvera/files/2024/01/PRAC-Instalacion-de-Ubuntu-server.pdf)[*Ubuntu server.*](https://www.uv.mx/personal/gvera/files/2024/01/PRAC-Instalacion-de-Ubuntu-server.pdf)
3. Esta práctica tiene como prerrequisito tener instalado el software Visual Studio Code con la extensión

para C++.

*Puede seguir las instrucciones de cómo instalar Visual Studio Code desde la práctica* [*Instalación de*](https://www.uv.mx/personal/gvera/files/2024/01/PRAC-Instalacion-de-Visual-Studio-Code.pdf)[*Visual Studio Code.*](https://www.uv.mx/personal/gvera/files/2024/01/PRAC-Instalacion-de-Visual-Studio-Code.pdf)

1. Puede seguir las instrucciones de cómo publicar un proyecto en Visual Studio Code a GitHub desde la

práctica [Como publicar un proyecto a GitHub desde Visual Studio Code](https://www.uv.mx/personal/gvera/files/2024/01/PRAC-Como-publicar-un-proyecto-a-GitHub-desde-VS-Code.pdf).

Instrucciones

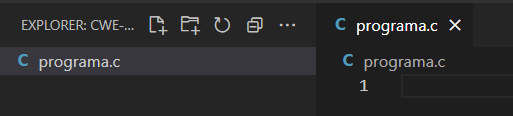
Esta vulnerabilidad consiste en que por medio de una entrada maliciosa, el programa tiene acceso a una variable privada por medio de la manipulación de la memoria. Vamos a ejemplificar el impacto de una vulnerabilidad de este tipo, mediante una práctica que consista en escribir un programa de código en c y un pequeño reto.

Para resolver el reto, se pide que el programa deba imprimir el mensaje con el valor Password correcto. Bienvenido! que indica que se pudo modificar el valor del password de entrada para obtener acceso.

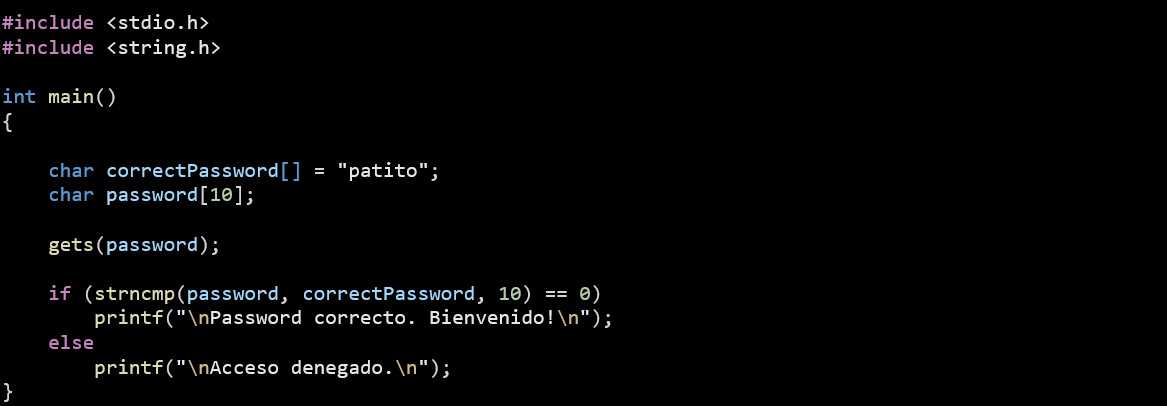
1. Crear el programa
   1. Asegúrese de contar con una carpeta de trabajo en C:\codigo. Si aún no ha creado la carpeta C:\codigo hágalo antes de continuar y compártala con su equipo Linux. Puede seguir las

instrucciones de cómo hacerlo desde [Compartir su carpeta de trabajo de Windows hacia Linux](https://www.uv.mx/personal/gvera/files/2024/01/PRAC-Compartir-su-carpeta-de-trabajo-de-Windows-hacia-Linux.pdf).

* 1. Inicie Visual Studio Code.
  2. Seleccione Archivo > Abrir carpeta en el menú principal.
  3. En el cuadro de diálogo Abrir carpeta, cree una carpeta en la ruta *C:\codigo\ps\cwe-767* y selecciónela. Luego haga clic en Seleccionar carpeta.
  4. El nombre de la carpeta se convierte en el nombre del proyecto y el nombre del espacio de nombres de forma predeterminada. Agregará código más adelante en el tutorial que asume que el espacio de nombres del proyecto es *cwe-767*.
  5. En la barra de herramientas del Explorador de archivos, seleccione el botón Nuevo archivo.
  6. Asigne un nombre al archivo programa.c y VS Code lo abrirá automáticamente en el editor.



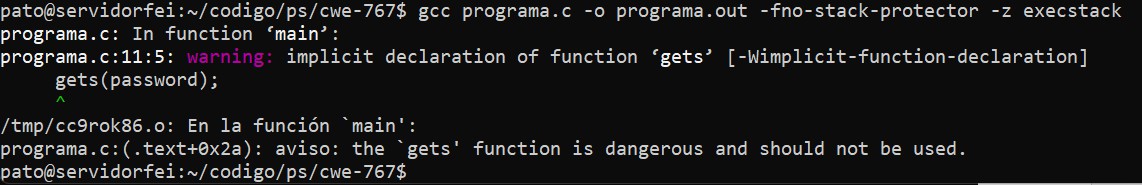
* 1. Escriba el siguiente código.



* 1. Abra una nueva Terminal, colóquese en la ubicación de la carpeta compartida de este código. En este ejemplo es cd codigo/ps/cwe-767/.
  2. Escriba el siguiente comando para compilar su archivo.

$ gcc programa.c -o programa.out -fno-stack-protector -z execstack

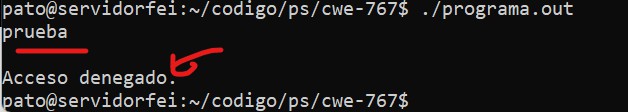
* 1. Observe como aparecen varias advertencias debido a que los nuevos compiladores, ya detectan posibles problemas de vulnerabilidades en sus funciones.



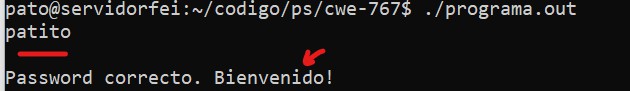
* 1. Este programa le solicita una contraseña y en caso de ser correcta, le imprime un mensaje de bienvenida. La idea es que nosotros escribamos ese mensaje sin utiliza la contraseña correcta. Ejecute su programa.

$ ./programa.out

* 1. El programa esperará que ingrese una cadena. Escriba: prueba. Presione Enter.



* 1. Pruebe con distintas cadenas. Parece imposible de logar puesto que no conocemos la contraseña.
  2. Escriba la contraseña correcta solo para comprobar que el programa si funciona.



* 1. El programa funciona correctamente. Ha creado su programa correctamente.

1. Explotación de la vulnerabilidad

El reto consiste en lograr imprimir el mensaje Password correcto! en pantalla sin utilizar el password correcto.

if (strncmp(password, correctPassword, 10) == 0) printf("\nPassword correcto. Bienvenido!\n");

else

printf("\nAcceso denegado.\n");

Esto parece imposible pues la función strncmp es una comparación de tipo == entre dos variables de longitud 10. Vamos a ejemplificar la vulnerabilidad mediante un payload que manipule la memoria para acceder a una variable privada crítica.

1. Aunque el siguiente payload puede hacerse de manera manual, generalmente estos se escriben en algún lenguaje de programación para practicidad tanto de ejecución y repetición.
2. Verifique que tiene la versión 2 de Python en su equipo Linux.

pato@servidorfei:~/codigo/ps/cwe-126$ python --version Python 2.7.12

1. La versión 3 de Python introduce varios cambios que podrían no ser compatibles con las instrucciones aquí presentadas.
2. En el proyecto de Visual Studio cree un nuevo archivo llamado payload.py e introduzca el siguiente código.

#payload.py

# Sabemos que la cadena password[10] mide 10 bytes # Sabemos que strncmp(password, correctPassword, 10) # compara 10 bytes de ambas variables

# Esto da una longitud de 20 bytes de la pila output = "A" \* 10

print(output)

1. Observe la que hace el payload.
   * Sabemos que la cadena

password[10]

strncmp(password, correctPassword, 10)

20 bytes

* + Sabemos que
  + Esto da una longitud de

mide 10 bytes. de la pila.

compara

de ambas variables.

10 bytes

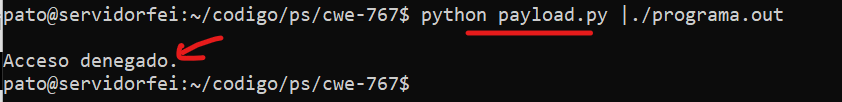
1. Como prueba, vamos a comenzar con un relleno de 10 bytes, es decir, 10 letras A.
2. Ahora ejecute este programa en su equipo Linux.



$ python payload.py

1. Observe como Python imprime 10 letras A. Ingrese el payload del programa de Python como entrada con la siguiente instrucción

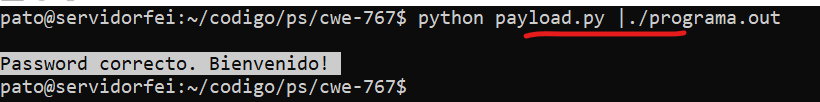
$ python payload.py |./programa.out



1. Observe como aún no es posible ingresar, puesto que solo modificamos los primeros 10 bytes de la variable char password[10]. Aún no resta modificar los 10 bytes de la variable char correctPassword[].
2. Basta con modificar nuestro archivo de Python para que imprima 20 bytes.

# Esto da una longitud de 20 bytes de la pila output = "A" \* 20

1. Ahora ejecute el programa nuevamente. Observe la salida.



1. Con esto ha superado el reto explotando la vulnerabilidad de escribir en una variable privada critica manipulando la memoria.
2. Felicidades, ha creado su aplicación de manera correcta.