

Instituto Tecnológico de Costa Rica  
Centro Académico de Alajuela  
IC3101. Arquitectura de Computadoras



**Laboratorio #1:**  
**Encontrando claves por des-ensamblaje de binarios generados en compiladores de alto nivel**

**Grupo 6:**  
Marco Rodriguez Vargas – Carné: 2022149445  
Ion Angel Dolanescu Bravo – Carné: 2022049034  
Alejandro Campos Paredes – Carné: 2022058238

**Profesor:**  
Ing. Emmanuel Ramírez Segura

**Fecha de entrega:**  
14/09/2021

II Semestre, 2022

## **ÍNDICE**

<b>Objetivo:</b>	<b>2</b>
<b>Descripción de la solución:</b>	<b>2</b>
Solución ejercicio 1:	2
Solución ejercicio 2:	3
<b>LECCIONES APRENDIDAS:</b>	<b>4</b>
<b>LOGROS O FALLOS:</b>	<b>4</b>
<b>BIBLIOGRAFÍA:</b>	<b>5</b>
<b>ANEXO:</b>	<b>5</b>

## Objetivo:

Entender, mediante el uso de herramientas como los des-ensambladores y debuggers, el lenguaje ensamblador generado por compiladores en alto nivel sobre una arquitectura x86.

## Descripción de la solución:

### Solución ejercicio 1:

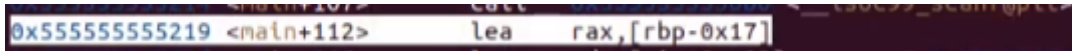
Para la solución del ejercicio 1 hicimos uso del GDB y el compilador GCC

Lo primero que se realizó fue compilar el archivo con el comando de GCC:

```
gcc -m64 -masm=intel -S uno.c -o uno.s && gcc -m64 -masm=intel -c uno.s -o uno.o && gcc -m64 uno.o -o uno
```

Eso nos dio un archivo en binario y lo abrimos con el GDB

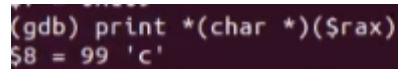
Después de un momento analizando el GDB nos dimos cuenta que muy probablemente la contraseña estuviera guardada en "rax"



```
0x55555555219 <main+112> lea rax, [rbp-0x17]
```

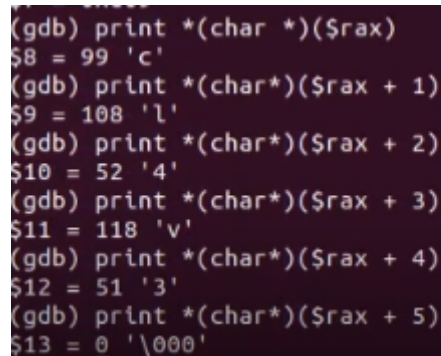
Debido a este comando "lea": ya que este comando hace una copia de la dirección b y lo copia en la dirección a y es muy probable que el motivo de esta copia sea para guardar un string o en este caso la contraseña

Entonces accedimos a la dirección de memoria y lo imprimimos (Como sabemos que la clave es alfanumérica usamos el comando "print \*(char \*)(\$rax)" )



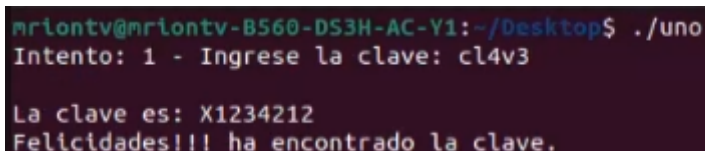
```
(gdb) print *(char *)($rax)
$8 = 99 'c'
```

y nos dio 99 que en ASCII es c, entonces supusimos que esta podría ser la contraseña pero tendríamos que seguir imprimiendo hasta encontrar un fin de "texto" (0)



```
(gdb) print *(char *)($rax)
$8 = 99 'c'
(gdb) print *(char *)($rax + 1)
$9 = 108 'l'
(gdb) print *(char *)($rax + 2)
$10 = 52 '4'
(gdb) print *(char *)($rax + 3)
$11 = 118 'v'
(gdb) print *(char *)($rax + 4)
$12 = 51 '3'
(gdb) print *(char *)($rax + 5)
$13 = 0 '\000'
```

Y nos quedaron los siguientes caracteres 99,108,52,118,51 o en ASCII: cl4v3 supusimos que esta era la clave pero para verificarlo teníamos que probarlo



```
mriontv@mriontv-B560-DS3H-AC-Y1:~/Desktop$ ./uno
Intento: 1 - Ingrese la clave: cl4v3

La clave es: X1234212
Felicidades!!! ha encontrado la clave.
```

Así que lo ingresamos y efectivamente esta era la contraseña

## Solución ejercicio 2:

Para la solución del ejercicio 1 hicimos uso del GDB y el compilador GCC

Lo primero que se realizó fue compilar el archivo con el comando de GCC:

"gcc -m64 -masm=intel -S uno.c -o uno.s && gcc -m64 -masm=intel -c uno.s -o uno.o &&

gcc -m64 uno.o -o uno". Eso nos dio un archivo en binario y lo abrimos con el GDB

Para encontrar la segunda clave del código dos de C, haremos uso del GDB. Usaremos el comando de "set disassembly-flavor intel" para ponerlo en NASM, "layout regs" y "run"

```
multi-thre Thread 0x7ffff7d807 In: main L?? PC: 0x55555555191
(gdb) set disassembly-flavor intel
(gdb) layout regs
(gdb) run
Starting program: /home/mriontv/Desktop/dos
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, 0x000055555555191 in main ()
(gdb)
```

Posteriormente usamos el comando run para recorrer las líneas

```
multi-thre Thread 0x7ffff7d807 In: main L?? PC: 0x55555555191
(gdb) set disassembly-flavor intel
(gdb) layout regs
(gdb) run
Starting program: /home/mriontv/Desktop/dos
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, 0x000055555555191 in main ()
(gdb) nextl
```

Una vez llegamos a donde creemos está el scanf de C, nos encontramos con un compare que creemos es donde se encuentra almacenada la contraseña. Para esto probamos con lo que está almacenado en \$3 y efectivamente era la contraseña.

```
0x55555555101 <main+72> lea rax,[rip+0xe3f] # 0x55555555017
0x55555555108 <main+79> mov rdi,rax
0x5555555510b <main+82> mov eax,0x0
0x5555555510e <main+87> call 0x55555555000 <printf@plt>
0x55555555109 <main+92> lea rax,[rbp-0x10]
0x5555555510c <main+96> mov rsi,rax
0x5555555510e <main+99> lea rax,[rip+0xe37] # 0x5555555502a
0x555555551f3 <main+106> mov rdi,rax
0x555555551f6 <main+109> mov eax,0x0
> 0x555555551fb <main+114> call 0x55555555090 <_isoc99_scanf@plt>
0x555555551fb <main+114> call 0x55555555090 <_isoc99_scanf@plt>
0x55555555200 <main+119> mov eax,DWORD PTR [rbp-0x10]
> 0x55555555203 <main+122> cmp eax,0x163c9 <main+133>
0x55555555204 <main+129> mov BYTE PTR [rbp-0x11],0x1
0x5555555520e <main+133> add DWORD PTR [rbp-0xc],0x1
0x55555555212 <main+137> movzx eax,BYTE PTR [rbp-0x11]
0x55555555215 <main+141> xor eax,0x1

multi-thre Thread 0x7ffff7d807 In: main L?? PC: 0x555555551fb
0x000055555555116 in main ()
0x00005555555511e in main ()
0x0000555555551e5 in main ()
0x0000555555551e9 in main ()
0x0000555555551ec in main ()
0x0000555555551f3 in main ()
0x0000555555551f6 in main ()
0x0000555555551fb in main ()
Intento: 1 - Ingrese la clave: 21
0x000055555555200 in main ()
(gdb) print $eax
$1 = 1
(gdb) nextl
0x000055555555203 in main ()
(gdb) print $eax
$2 = 21
(gdb) print 0x163c9
$3 = 91081
(gdb)
```

Para comprobar nuestra suposición procedemos a ingresar la contraseña y efectivamente la clave era correcta. Por lo que encontramos la nueva clase que es “1234212”

```
mrlontv@mrlontv-B560-DS3H-AC-Y1: ~/Desktop
mrlontv@mrlontv-B560-DS3H-AC-Y1: ~/Desktop
mrlontv@mrlontv-B560-DS3H-AC-Y1: ~/Desktop$ ./dos
Intento: 1 - Ingrese la clave: 91081
La clave es: 1234212
Felicidades!!! ha encontrado la clave.
mrlontv@mrlontv-B560-DS3H-AC-Y1: ~/Desktop$
```

### LECCIONES APRENDIDAS:

Por medio de la realización del proyecto conseguimos aprender distintos conocimientos como el manejo de “debuggers” (en específico el GDB), lo más destacado del aprendizaje de los debuggers fue a entender cómo se almacena la memoria, como ver los registros de las instrucciones y cómo analizar archivos en binario además el laboratorio nos ayudó a entender la importancia de la ingeniería inversa en una sociedad como la nuestra

**LOGROS O FALLOS:**

Con respecto a logros o fallos, nuestro equipo logró exitosamente en la búsqueda de contraseñas en registros de ensamblador conseguir ambas contraseñas en ambos ejercicios. Algo que nos facilitó la búsqueda de esto fue el “cheat sheet” de gdb, con el cual pudimos usar varios comandos de su lista para encontrar las contraseñas del laboratorio.

[illegible]

## BIBLIOGRAFÍA:

**Rapidtables. (2018). *Attention required!***

<https://www.rapidtables.com/convert/number/hex-to-decimal.html>

**Marc Haisenko. (2007).**

<https://darkdust.net/files/GDB%20Cheat%20Sheet.pdf>

**GDB: The GNU project debugger. (n.d.).** [sourceware.org.](https://sourceware.org/gdb/)

<https://www.sourceware.org/gdb/>

## ANEXO:

IC3101. Arquitectura de Computadoras - II Semestre 2021

Laboratorio: #

Rúbrica para evaluar a los compañeros de mi grupo de trabajo (Grupo de 3 integrantes)

Estudiante evaluado:	Nombre Apellido1 Apellido2									
Trabajo Personal	Rúbrica									
	Siempre (1 punto)			A veces (0,5 puntos)			Nunca (0 puntos)			Puntos Obtenidos
	E1	E2	Autoevaluación	E1	E2	Autoevaluación	E1	E2	Autoevaluación	
Es responsable con la parte del trabajo asignada.										
Participa de las reuniones virtuales coordinadas por el grupo.										
Es respetuoso(a) con los miembros del grupo.										
Contribuye con la solución de las claves de los programas binarios.										
Contribuye en la elaboración del documento del proyecto.										
TOTAL:										

Cálculo del % (De un máximo de 10% por estudiante):

Fórmula: Puntos Obtenidos / 0,15 x 10% =

IC3101. Arquitectura de Computadoras - II Semestre 2021

Laboratorio: #

Rúbrica para evaluar a los compañeros de mi grupo de trabajo (Grupo de 3 integrantes)

Estudiante evaluado:	Nombre Apellido1 Apellido2									
Trabajo Personal	Rúbrica									
	Siempre (1 punto)			A veces (0,5 puntos)			Nunca (0 puntos)			Puntos Obtenidos
	E1	E2	Autoevaluación	E1	E2	Autoevaluación	E1	E2	Autoevaluación	
Es responsable con la parte del trabajo asignada.										
Participa de las reuniones virtuales coordinadas por el grupo.										
Es respetuoso(a) con los miembros del grupo.										
Contribuye con la solución de las claves de los programas binarios.										
Contribuye en la elaboración del documento del proyecto.										
TOTAL:										

Cálculo del % (De un máximo de 10% por estudiante):

Fórmula: Puntos Obtenidos / 0,15 x 10% =

## Rúbrica para evaluar a los compañeros de mi grupo de trabajo (Grupo de 3 integrantes)

Estudiante evaluado:	Nombre Apellido1 Apellido2									
Trabajo Personal	Rúbrica									
	Siempre (1 punto)			A veces (0,5 puntos)			Nunca (0 puntos)			Puntos Obtenidos
	E1	E2	Autoevaluación	E1	E2	Autoevaluación	E1	E2	Autoevaluación	
Es responsable con la parte del trabajo asignada.										
Participa de las reuniones virtuales coordinadas por el grupo.										
Es respetuoso(a) con los miembros del grupo.										
Contribuye con la solución de las claves de los programas binarios.										
Contribuye en la elaboración del documento del proyecto.										
TOTAL:										

Cálculo del % (De un máximo de 10% por estudiante):

Fórmula: Puntos Obtenidos / 0,15 x 10% =