

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



Laboratorio #1:
Utilizando debuggers en bajo nivel

Grupo 1:
Valery Carvajal Oreamuno – Carné: 2022314299
Raquel Gómez Zamora – Carné: 2022099256

Profesor:
Ing. Emmanuel Ramírez Segura

Fecha de entrega
28/03/2023

I Semestre, 2023

ÍNDICE

Objetivo	2
Descripción de la solución	3
Lecciones Aprendidas	5
Logros o fallos	5
Bibliografía	6
Anexo 1	7

Objetivo

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

Descripción de la solución

Para llegar a la solución tuvimos que ejecutar una serie de pasos:

1. Ejecutamos el gdb en modo gráfico con el comando "lay next".

```
sbctiom@sbctiom-VirtualBox: ~/Downloads

0x7ffff7d1496e < _openat64_2+30>    push    %rax
0x7ffff7d1496f < _openat64_2+31>    lea     0xcb2fa(%rip),%rdi        # 0x7ffff7ddfc70
0x7ffff7d14976 < _openat64_2+38>    call   0x7ffff7d36740 < __GI___fortify_fail>
0x7ffff7d1497b                nopl    0x0(%rax,%rax,1)
0x7ffff7d14980 < __GI___libc_read>      endbr64
0x7ffff7d14984 < __GI___libc_read+4>    mov     %fs:0x18,%eax
0x7ffff7d1498c < __GI___libc_read+12>   test    %eax,%eax
0x7ffff7d1498e < __GI___libc_read+14>   jne     0x7ffff7d149a0 < __GI___libc_read+32>
0x7ffff7d14990 < __GI___libc_read+16>   syscall
> 0x7ffff7d14992 < __GI___libc_read+18>  cmp     $0xffffffffffff000,%rax
0x7ffff7d14998 < __GI___libc_read+24>   ja      0x7ffff7d149f0 < __GI___libc_read+112>
0x7ffff7d1499a < __GI___libc_read+26>   ret
0x7ffff7d1499b < __GI___libc_read+27>   nopl    0x0(%rax,%rax,1)
0x7ffff7d149a0 < __GI___libc_read+32>   sub     $0x28,%rsp
0x7ffff7d149a4 < __GI___libc_read+36>   mov     %rdx,0x18(%rsp)
0x7ffff7d149a9 < __GI___libc_read+41>   mov     %rsi,0x10(%rsp)
0x7ffff7d149ae < __GI___libc_read+46>   mov     %edi,0x8(%rsp)
0x7ffff7d149b2 < __GI___libc_read+50>   call    0x7ffff7c90a70 < __GI___pthread_enable_asynccancel>
0x7ffff7d149b7 < __GI___libc_read+55>   mov     0x18(%rsp),%rdx
0x7ffff7d149bc < __GI___libc_read+60>   mov     0x10(%rsp),%rsi
0x7ffff7d149c1 < __GI___libc_read+65>   mov     %eax,%r8d
0x7ffff7d149c4 < __GI___libc_read+68>   mov     0x8(%rsp),%edi
0x7ffff7d149c8 < __GI___libc_read+72>   xor     %eax,%eax
0x7ffff7d149ca < __GI___libc_read+74>   syscall
0x7ffff7d149cc < __GI___libc_read+76>   cmp     $0xffffffffffff000,%rax
0x7ffff7d149d2 < __GI___libc_read+82>   ja      0x7ffff7d14a08 < __GI___libc_read+136>
0x7ffff7d149d4 < __GI___libc_read+84>   mov     %r8d,%edi
0x7ffff7d149d7 < __GI___libc_read+87>   mov     %rax,0x8(%rsp)
0x7ffff7d149dc < __GI___libc_read+92>   call    0x7ffff7c90ae0 < __GI___pthread_disable_asynccancel>
0x7ffff7d149e1 < __GI___libc_read+97>   mov     0x8(%rsp),%rax
0x7ffff7d149e6 < __GI___libc_read+102>  add     $0x28,%rsp
0x7ffff7d149ea < __GI___libc_read+106>  ret
0x7ffff7d149eb < __GI___libc_read+107>  nopl    0x0(%rax,%rax,1)
0x7ffff7d149f0 < __GI___libc_read+112>  mov     0x104419(%rip),%rdx      # 0x7ffff7e18e10
0x7ffff7d149f7 < __GI___libc_read+119>  neg     %eax
0x7ffff7d149f9 < __GI___libc_read+121>  mov     %eax,%fs:(%rdx)
0x7ffff7d149fc < __GI___libc_read+124>  mov     $0xffffffffffff,%rax
0x7ffff7d14a03 < __GI___libc_read+131>  ret

multi-thre Thread 0x7ffff7fab7 In:  __GI___libc_read
(gdb)
```

2. Corrimos el programa e introducimos un valor al azar (1234) para que fallara.

```
multi-thre Thread 0x7ffff7fab7 In:  __GI___libc_read
(gdb) r
The program being debugged has been started already.
Start it from the beginning? (y or n) yStarting program: /home/sbctiom/Downloads/uno
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Intento: 1 - Ingrese la clave: 1234
Intento: 2 - Ingrese la clave: ^CProgram received signal SIGINT, Interrupt.
0x00007ffff7d14992 in __GI___libc_read (fd=0, buf=0x5555555596b0, nbytes=1024) at ../sysdeps/unix/sysv/linux/read.c:26
(gdb)
```

3. Eso genera que se quede paralizado el gdb marcando la zona donde hubo el error, usamos eso como referencia para buscar el lugar donde se estaba tomando la entrada del teclado.

```

0x55555555162 <_do_global_dtors_aux+34> call 0x55555555060 <__cxa_finalize@plt>
0x55555555167 <_do_global_dtors_aux+39> call 0x555555550d0 <deregister_tm_clones>
0x5555555516c <_do_global_dtors_aux+44> movb $0x1,0x2e9d(%rip) # 0x555555558010 <completed.8061>
0x55555555173 <_do_global_dtors_aux+51> pop %rbp
0x55555555174 <_do_global_dtors_aux+52> ret
0x55555555175 <_do_global_dtors_aux+53> nopl (%rax)
0x55555555178 <_do_global_dtors_aux+56> ret
0x55555555179 <_do_global_dtors_aux+57> nopl 0x0(%rax)
0x55555555180 <frame_dummy> endbr64
0x55555555184 <frame_dummy+4> jmp 0x55555555100 <register_tm_clones>
0x55555555189 <main> endbr64
0x5555555518d <main+4> push %rbp
0x5555555518e <main+5> mov %rsp,%rbp
B+> 0x55555555191 <main+8> sub $0x20,%rsp
0x55555555195 <main+12> mov %fs:0x28,%rax
0x5555555519e <main+21> mov %rax,-0x8(%rbp)
0x555555551a2 <main+25> xor %eax,%eax
0x555555551a4 <main+27> movb $0x0,-0x11(%rbp)
0x555555551a8 <main+31> movl $0x1,-0xc(%rbp)
0x555555551af <main+38> movl $0x0,-0x10(%rbp)
0x555555551b6 <main+45> jmp 0x55555555209 <main+128>
0x555555551b8 <main+47> mov -0xc(%rbp),%eax
0x555555551bb <main+50> mov %eax,%esi
0x555555551bd <main+52> lea 0xe44(%rip),%rdi # 0x555555556008
0x555555551c4 <main+59> mov $0x0,%eax
0x555555551c9 <main+64> call 0x55555555080 <printf@plt>
0x555555551ce <main+69> lea 0xe42(%rip),%rdi # 0x555555556017
0x555555551d5 <main+76> mov $0x0,%eax
0x555555551da <main+81> call 0x55555555080 <printf@plt>
0x555555551df <main+86> lea -0x10(%rbp),%rax
0x555555551e3 <main+90> mov %rax,%rsi
0x555555551e6 <main+93> lea 0xe3d(%rip),%rdi # 0x55555555602a
0x555555551ed <main+100> mov $0x0,%eax
0x555555551f2 <main+105> call 0x55555555090 <__isoc99_scanf@plt>
0x555555551f7 <main+110> mov -0x10(%rbp),%eax
0x555555551fa <main+113> cmp $0x6b4cd3,%eax
0x555555551ff <main+118> jne 0x55555555205 <main+124>
0x55555555201 <main+120> movb $0x1,-0x11(%rbp)

```

multi-thre Thread 0x7ffff7fab7 In: main

4. Una vez localizamos dónde se tomaba la entrada del teclado, nos fijamos en cuál era el registro en el que se estaba guardando y buscamos el próximo compare (cmp) donde se estuviera comparando ese registro con algo más y encontramos que se estaba comparando con un número hexadecimal.

```

0x5555555518e <main+5> mov %rsp,%rbp
B+> 0x55555555191 <main+8> sub $0x20,%rsp
0x55555555195 <main+12> mov %fs:0x28,%rax
0x5555555519e <main+21> mov %rax,-0x8(%rbp)
0x555555551a2 <main+25> xor %eax,%eax
0x555555551a4 <main+27> movb $0x0,-0x11(%rbp)
0x555555551a8 <main+31> movl $0x1,-0xc(%rbp)
0x555555551af <main+38> movl $0x0,-0x10(%rbp)
0x555555551b6 <main+45> jmp 0x55555555209 <main+128>
0x555555551b8 <main+47> mov -0xc(%rbp),%eax
0x555555551bb <main+50> mov %eax,%esi
0x555555551bd <main+52> lea 0xe44(%rip),%rdi # 0x555555556008
0x555555551c4 <main+59> mov $0x0,%eax
0x555555551c9 <main+64> call 0x55555555080 <printf@plt>
0x555555551ce <main+69> lea 0xe42(%rip),%rdi # 0x555555556017
0x555555551d5 <main+76> mov $0x0,%eax
0x555555551da <main+81> call 0x55555555080 <printf@plt>
0x555555551df <main+86> lea -0x10(%rbp),%rax
0x555555551e3 <main+90> mov %rax,%rsi
0x555555551e6 <main+93> lea 0xe3d(%rip),%rdi # 0x55555555602a
0x555555551ed <main+100> mov $0x0,%eax
0x555555551f2 <main+105> call 0x55555555090 <__isoc99_scanf@plt>
0x555555551f7 <main+110> mov -0x10(%rbp),%eax
0x555555551fa <main+113> cmp $0x6b4cd3,%eax
0x555555551ff <main+118> jne 0x55555555205 <main+124>
0x55555555201 <main+120> movb $0x1,-0x11(%rbp)

```

multi-thre Thread 0x7ffff7fab7 In: main

5. Pasamos el número hexadecimal a decimal.

Handwritten calculation on lined paper showing the conversion of the hexadecimal number $GB4CD3_{16}$ to decimal. The digits are aligned with their powers of 16: G (16⁵), B (16⁴), 4 (16³), C (16²), D (16¹), and 3 (16⁰). The calculation is: $6 \times 16^5 + 11 \times 16^4 + 4 \times 16^3 + 12 \times 16^2 + 13 \times 16^1 + 3 \times 16^0 =$. The result, 7032019, is boxed in red.

6. Corrimos de nuevo el programa y pusimos el número en decimal que obtuvimos de la conversión, encontrándonos con que efectivamente era la clave.

```
sbctiom@sbctiom-VirtualBox:~$ cd Downloads/
sbctiom@sbctiom-VirtualBox:~/Downloads$ gdb uno
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from uno...
(gdb) r
Starting program: /home/sbctiom/Downloads/uno
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Intento: 1 - Ingrese la clave: 7032019

La clave es: 1234212
Felicidades!!! ha encontrado la clave.
[Inferior 1 (process 86768) exited normally]
(gdb)
```

Lecciones Aprendidas

Trabajando este laboratorio logramos reforzar y complementar algunos temas explicados en clase con respecto a herramientas de desensamblaje. Investigando, aprendimos cómo funcionan diversas instrucciones de *NASM* que no conocíamos y terminamos de entender algunas mencionadas en clase. Por otro lado, con lo visto en clase en complemento a la investigación grupal del tema, aprendimos a utilizar el *GNU Project Debugger*, también conocido como *gdb*.

Logros o fallos

Para llegar a la respuesta correcta, hicimos dos pruebas. La primera fue la serie de números “1234”, no teníamos certeza de si esta era o no la clave correcta, ingresamos ese intento únicamente para que el programa retornara el fallo y que así marcara la zona donde ocurrió error en el layout gráfico.

```
multi-thre Thread 0x7ffff7fab7 In: _GI__libc_read
(gdb) r
The program being debugged has been started already.
Start it from the beginning? (y or n) yStarting program: /home/sbctiom/Downloads/uno
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Intento: 1 - Ingrese la clave: 1234
Intento: 2 - Ingrese la clave: ^CProgram received signal SIGINT, Interrupt.
0x00007ffff7d14992 in __GI__libc_read (fd=0, buf=0x5555555596b0, nbytes=1024) at ../sysdeps/unix/sysv/linux/read.c:26
(gdb)
```

Con los pasos mencionados en la “Descripción de la solución” encontramos el número en hexadecimal que pensábamos era la respuesta correcta, este número fue “6b4cd3” lo que en decimal equivale a “7032019”, lo ingresamos y vimos que efectivamente, esta era la clave correcta.

```
sbctiom@sbctiom-VirtualBox:~$ cd Downloads/
sbctiom@sbctiom-VirtualBox:~/Downloads$ gdb uno
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from uno...
(No debugging symbols found in uno)
(gdb) r
Starting program: /home/sbctiom/Downloads/uno
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Intento: 1 - Ingrese la clave: 7032019

La clave es: 1234212
Felicidades!!! ha encontrado la clave.
[Inferior 1 (process 86768) exited normally]
(gdb)
```

Bibliografía

Low Level Learning. (2021, April 17). *Getting Started with Debugging using GDB | Find Bugs in Your Code with A Couple Easy Commands* [Video]. YouTube.

https://www.youtube.com/watch?app=desktop&v=Dq8l1_-QgAc

Debugging with GDB. (n.d.).

https://ftp.gnu.org/old-gnu/Manuals/gdb/html_node/gdb_toc.html

Benjamin Xoaquin. (2022, July 1). *NASM Entrada de datos por el usuario | Ensamblador x86_64 linux | Parte 4*. YouTube. <https://www.youtube.com/watch?v=kPZsiEYRh5Y>

Linux x86_64 nasm assembly syscalls. (n.d.). Stack Overflow.

<https://stackoverflow.com/questions/60014068/linux-x86-64-nasm-assembly-syscalls>

Anexo 1

Rúbrica para evaluar a los compañeros de mi grupo de trabajo (Grupo de 2 integrantes)

Estudiante evaluado: Raquel Gómez Zamora							
Trabajo Personal	Rúbrica						Puntos Obtenidos
	Siempre (1 punto)		A veces (0,5 puntos)		Nunca (0 puntos)		
	E1	Autoevaluación	E1	Autoevaluación	E1	Autoevaluación	
Es responsable con la parte del trabajo asignada.	1	1					2
Participa de las reuniones virtuales coordinadas por el grupo.	1	1					2
Es respetuoso(a) con los miembros del grupo.	1	1					2
Contribuye con la solución de las claves de los programas binarios.	1	1					2
Contribuye en la elaboración del documento del proyecto.	1	1					2
						TOTAL:	10

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

Fórmula: Puntos Obtenidos / 0,1 x 10% =

10%

Rúbrica para evaluar a los compañeros de mi grupo de trabajo (Grupo de 2 integrantes)

Estudiante evaluado: Carvajal Oreamuno Valery							
Trabajo Personal	Rúbrica						Puntos Obtenidos
	Siempre (1 punto)		A veces (0,5 puntos)		Nunca (0 puntos)		
	E1	Autoevaluación	E1	Autoevaluación	E1	Autoevaluación	
Es responsable con la parte del trabajo asignada.	1	1					2
Participa de las reuniones virtuales coordinadas por el grupo.	1	1					2
Es respetuoso(a) con los miembros del grupo.	1	1					2
Contribuye con la solución de las claves de los programas binarios.	1	1					2
Contribuye en la elaboración del documento del proyecto.	1	1					2
						TOTAL:	10

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

Fórmula: Puntos Obtenidos / 0,1 x 10% = 10%