

Radare2 یک فریم‌ورک متن‌باز^۱ است که برای مهندسی معکوس^۲ مورد استفاده قرار می‌گیرد.

ابتدا یک برنامه‌ی hello world به زبان C می‌نویسیم:

A screenshot of a text editor window titled "Text Editor" with a timestamp "Tue 15:15". The window shows a file named "hello-world.c" located on the desktop. The code inside is a simple C program that includes <stdio.h>, defines a main function, and uses printf to print "Hello World\n" before returning 0.

```
#include <stdio.h>
int main()
{
    printf("Hello World\n");
    return 0;
}
```

این برنامه را کامپایل کرده و اجرا می‌کنیم تا خروجی Hello World به ما نمایش داده شود:

A screenshot of a terminal window showing the steps to compile and run the C program. The user is at the root prompt in a Kali Linux environment, in the directory ~/Desktop. They use gedit to open the file, gcc to compile it into an executable named hello-world, and then run the executable, which outputs "Hello World".

```
root@kali:~/Desktop# gedit hello-world.c
root@kali:~/Desktop# gcc hello-world.c -o hello-world
root@kali:~/Desktop# ./hello-world
Hello World
root@kali:~/Desktop#
```

Rabin2 یکی از ابزارهایی است که برای به دست آوردن اطلاعات در مورد یک فایل باینری مانند string ها، زمان کامپایل و اطلاعات مفید دیگر می‌تواند مورد استفاده قرار گیرد. ما نیز از این ابزار استفاده می‌کنیم تا اطلاعاتی در مورد این برنامه به دست بیاوریم. آپشن -I اطلاعات مهم در مورد این فایل باینری را در اختیار قرار می‌دهد:

¹ Open source

² Reverse engineering

```
root@kali: ~/Desktop
File Edit View Search Terminal Help
root@kali:~/Desktop# rabin2 -I hello-world
pic      false
canary   false
nx        true
crypto    false
va        true
bintype   elf
class     ELF32
lang      c
arch      x86
bits      32
machine   Intel 80386
os        linux
subsys    linux
endian    little
stripped  false
static    false
linenum   true
lsyms     true
relocs    true
rpath     NONE
binsz     3693
root@kali:~/Desktop#
```

خروجی این دستور نشان می‌دهد که این برنامه روی لینوکس اجرا می‌شود و با زبان C نوشته شده است.

دستور `rabin2 -z` همه‌ی string های data section این فایل باینری را نشان می‌دهد:

```
root@kali:~/Desktop# rabin2 -z hello-world
vaddr=0x080484a0 paddr=0x000004a0 ordinal=000 sz=12 len=11 section=.rodata type=
a string=Hello World
root@kali:~/Desktop#
```

همان‌گونه که مشاهده می‌کنید این برنامه یک string دارد: "Hello World".

حال به سراغ radare2 می‌رویم تا کد اسمبلی برنامه‌ی خود را مشاهده کنیم. با دستور زیر برنامه‌ی خود را در radare2 ، load کرده و وارد محیط radare2 می‌شویم:

```
root@kali:~/Desktop# r2 hello-world
[0x080482f0]>
```

گام بعدی این است که به radare2 اجازه دهیم تا این فایل باینری را آنالیز کند و اطلاعاتی مانند string ها، توابع و اطلاعات مهم دیگر را از آن استخراج کند. بدین منظور دستور `aa` را اجرا می‌کنیم:

```
root@kali:~/Desktop# r2 hello-world
[0x080482f0]> aa
[0x080482f0]>
```

دستور s در radare2 برای جستجوی نقطه‌ی مشخصی از حافظه استفاده می‌شود که می‌توانیم یک آدرس هگزادسیمال یا اسم یک تابع را به آن بدهیم. از آن‌جا که اکثر برنامه‌های لینوکس با تابع main آغاز می‌شوند ما به دنبال تابع main می‌گردیم:

```
[0x080482f0]> s main
[0x080483eb]>
```

همان‌گونه که مشاهده می‌کنید آدرس از 0x080482f0 به 0x080483eb تغییر کرد. این به این معنا است که ما اکنون در تابع main هستیم و آدرس کنونی ما 0x080483eb است.

با دستور pdf (print disassembly function) می‌توانیم کد این تابع (تابع main) را مشاهده کنیم:

```

Applications ▾ Places ▾ Terminal ▾ Tue 17:39
root@kali: ~/Desktop
File Edit View Search Terminal Help
[0x080482f0]> aa
[0x080482f0]> s main
[0x080483eb]> pdf
/ (fcn) sym.main 46
; var int local_1 @ ebp-0x4
; DATA XREF from 0x08048307 (entry0)
main:
sym.main:
0x080483eb 8d4c2404 lea ecx, dword [esp + 4] ; 0x4
0x080483ef 83e4f0 and esp, 0xffffffff
0x080483f2 ff71fc push dword [ecx - 4]
0x080483f5 55 push ebp
0x080483f6 89e5 mov ebp, esp
0x080483f8 51 push ecx
0x080483f9 83ec04 sub esp, 4
0x080483fc 83ec0c sub esp, 0xc
0x080483ff 68a0840408 push str.Hello_World ; "Hello World"
@ 0x80484a0 0x08048404 e8b7feffff call sym.imp.puts ; fcn.080482bc+
0x4 ^- fcn.080482bc(unk) ; sym.imp.puts
0x08048409 83c410 add esp, 0x10
0x0804840c b800000000 mov eax, 0
0x08048411 8b4dfc mov ecx, dword [ebp-local_1]
0x08048414 c9 leave
0x08048415 8d61fc lea esp, dword [ecx - 4]
0x08048418 c3 ret
[0x080483eb]>
```

همان گونه که مشاهده می کنید در این تابع رشته‌ی “Hello World” در آدرسی از حافظه قرار می گیرد و سپس تابع sym.imp.pus فراخوانی می شود که این تابع رشته‌ی موردنظر (“Hello World”) را چاپ می کند و نهایتاً با ret تابع به پایان می رسد.

بنابراین ما بدون اینکه این برنامه را اجرا کنیم متوجه شدیم که این برنامه رشته‌ی “Hello World” را چاپ می کند.