REV IG
Week 3

Dynamic Analysis

Sept 21

By Dudcom

THE PLAN FOR TODAY

What is Dynamic Analysis and why do we want Dynamic analysis ?

1. Strace & Ltrace
2. Windows
3. Decompiler debuggers
4. Pwndbg
5. Frida
6. LD_PRELOAD

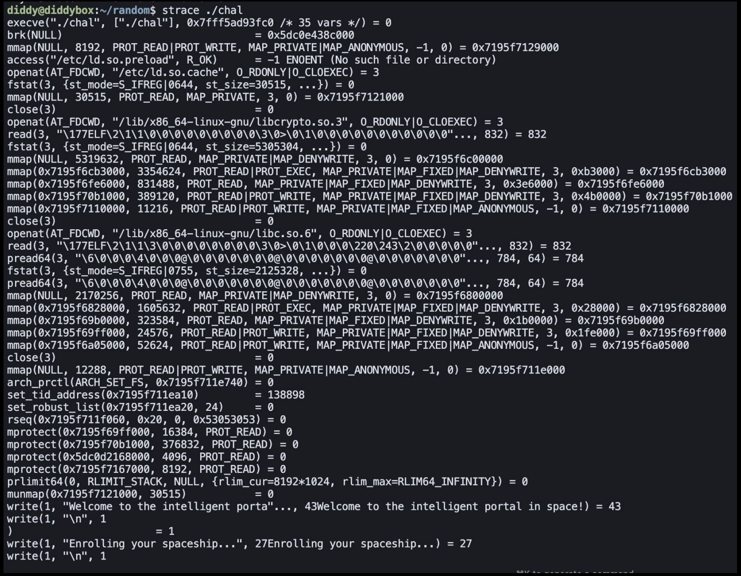
Strace & Ltrace

Strace (System Call Trace):

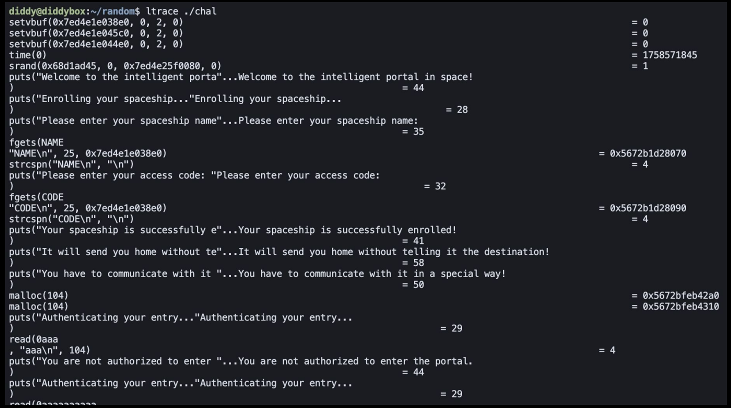
- Interactions between the Kernel and Userland
- Shows every system call a program makes
- Displays arguments passed to system calls
- Shows return values and error codes

Ltrace (Library Call Trace):

- calls to shared libraries (like libc functions: printf, malloc, strlen, etc.).
- Shows calls to dynamic library functions
- Displays function arguments and return values
- Can also trace system calls (with -S option)
- **sudo apt install ltrace**



```
diddy@diddybox:~/random$ strace ./chal
execve("./chal", ["./chal"], 0x7fff5ad93fc0 /* 35 vars */) = 0
brk(NULL)                               = 0x5dc0e438c000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7195f7129000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=30515, ...}) = 0
mmap(NULL, 30515, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7195f7121000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libcrypto.so.3", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=5305304, ...}) = 0
mmap(NULL, 5319632, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7195f6c00000
mmap(0x7195f6cb3000, 3354624, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xb3000) = 0x7195f6cb3000
mmap(0x7195f6fe6000, 831488, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3e6000) = 0x7195f6fe6000
mmap(0x7195f70b1000, 389120, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x4b0000) = 0x7195f70b1000
mmap(0x7195f7110000, 11216, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7195f7110000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\243\2\0\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7195f6800000
mmap(0x7195f6828000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7195f6828000
mmap(0x7195f69b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000) = 0x7195f69b0000
mmap(0x7195f69ff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x7195f69ff000
mmap(0x7195f6a05000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7195f6a05000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7195f711e000
arch_prctl(ARCH_SET_FS, 0x7195f711e740) = 0
set_tid_address(0x7195f711ea10)         = 138898
set_robust_list(0x7195f711ea20, 24)     = 0
rseq(0x7195f711f060, 0x20, 0, 0x53053053) = 0
mprotect(0x7195f69ff000, 16384, PROT_READ) = 0
mprotect(0x7195f70b1000, 376832, PROT_READ) = 0
mprotect(0x5dc0d2168000, 4096, PROT_READ) = 0
mprotect(0x7195f7167000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7195f7121000, 30515)           = 0
write(1, "Welcome to the intelligent porta"..., 43Welcome to the intelligent portal in space!) = 43
write(1, "\n", 1
)                       = 1
write(1, "Enrolling your spaceship...", 27Enrolling your spaceship...) = 27
write(1, "\n", 1
```



```
diddy@diddybox:~/random$ ltrace ./chal
setvbuf(0x7ed4e1e038e0, 0, 2, 0)                                            = 0
setvbuf(0x7ed4e1e045c0, 0, 2, 0)                                            = 0
setvbuf(0x7ed4e1e044e0, 0, 2, 0)                                            = 0
time(0)                                                                     = 1758571845
srand(0x68d1ad45, 0, 0x7ed4e25f0080, 0)                                     = 1
puts("Welcome to the intelligent porta"...Welcome to the intelligent portal in space!
)                                                = 44
puts("Enrolling your spaceship..."Enrolling your spaceship...
)                                                     = 28
puts("Please enter your spaceship name"...Please enter your spaceship name:
)                                                = 35
fgets(NAME
"NAME\n", 25, 0x7ed4e1e038e0)                                           = 0x5672b1d28070
strcspn("NAME\n", "\n")                                                     = 4
puts("Please enter your access code: "Please enter your access code:
)                                                  = 32
fgets(CODE
"CODE\n", 25, 0x7ed4e1e038e0)                                           = 0x5672b1d28090
strcspn("CODE\n", "\n")                                                     = 4
puts("Your spaceship is successfully e"...Your spaceship is successfully enrolled!
)                                                = 41
puts("It will send you home without te"...It will send you home without telling it the destination!
)                                                = 58
puts("You have to communicate with it "...You have to communicate with it in a special way!
)                                                = 50
malloc(104)                                                                 = 0x5672bfeb42a0
malloc(104)                                                                 = 0x5672bfeb4310
puts("Authenticating your entry..."Authenticating your entry...
)                                                    = 29
read(0aaa
, "aaa\n", 104)                                                         = 4
puts("You are not authorized to enter "...You are not authorized to enter the portal.
)                                                = 44
puts("Authenticating your entry..."Authenticating your entry...
)                                                    = 29
```

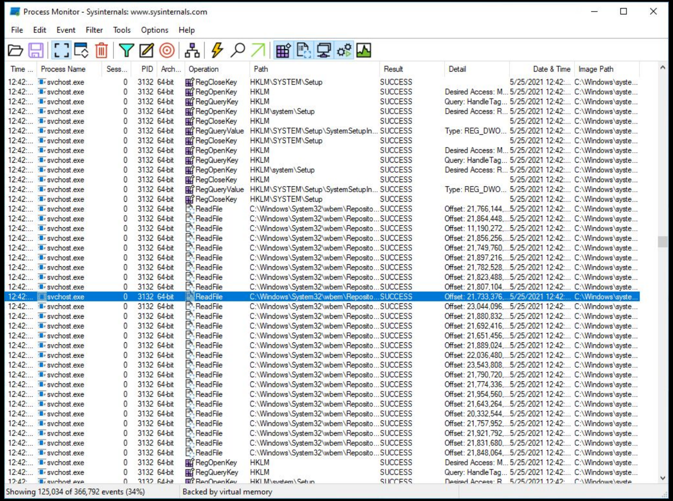
Program Analysis On Windows (Demo)

[Process Monitor - Sysinternals | Microsoft Learn]

Useful for seeing what program(s) are doing, where they are writing to and what they are calling.

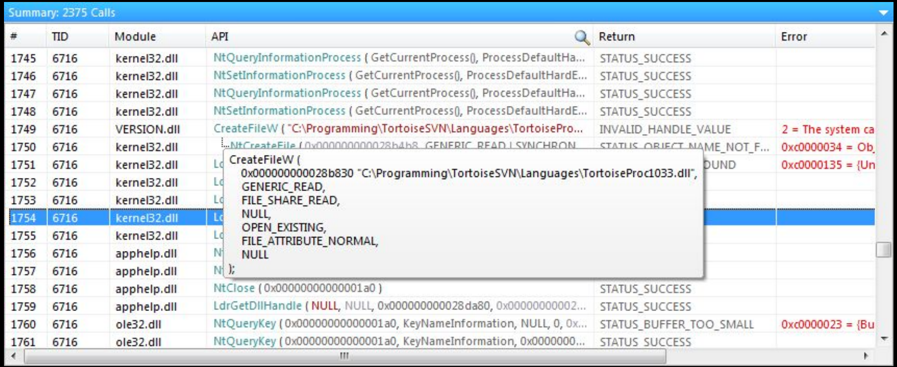
[API Monitor: Spy on API Calls and COM Interfaces (Freeware 32-bit and 64-bit Versions!) | rohitab.com]

Enables you to see DLL calls as well as what they return and send. Great for complex .exes



Time ...	Process Name	Sess...	PID	Arch...	Operation	Path	Result	Detail	Date & Time	Image Path
12:42:...	svchost.exe	0	3132	64-bit	RegCloseKey	HKLM\SYSTEM\Setup	SUCCESS		5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	RegOpenKey	HKLM	SUCCESS	Desired Access: M...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	RegQueryKey	HKLM	SUCCESS	Query: HandleTag...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	RegOpenKey	HKLM\system\Setup	SUCCESS	Desired Access: R...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	RegCloseKey	HKLM	SUCCESS		5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	RegQueryValue	HKLM\SYSTEM\Setup\SystemSetupIn...	SUCCESS	Type: REG_DWO...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	RegCloseKey	HKLM\SYSTEM\Setup	SUCCESS		5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	RegOpenKey	HKLM	SUCCESS	Desired Access: M...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	RegQueryKey	HKLM	SUCCESS	Query: HandleTag...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	RegOpenKey	HKLM\system\Setup	SUCCESS	Desired Access: R...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	RegCloseKey	HKLM	SUCCESS		5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	RegQueryValue	HKLM\SYSTEM\Setup\SystemSetupIn...	SUCCESS	Type: REG_DWO...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	RegCloseKey	HKLM\SYSTEM\Setup	SUCCESS		5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Reposito...	SUCCESS	Offset: 21,766,144,...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Reposito...	SUCCESS	Offset: 21,864,448,...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Reposito...	SUCCESS	Offset: 11,190,272,...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Reposito...	SUCCESS	Offset: 21,856,256,...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Reposito...	SUCCESS	Offset: 21,749,760,...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Reposito...	SUCCESS	Offset: 21,897,216,...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Reposito...	SUCCESS	Offset: 21,782,528,...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Reposito...	SUCCESS	Offset: 21,823,488,...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Reposito...	SUCCESS	Offset: 21,807,104,...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Reposito...	SUCCESS	Offset: 21,733,376,...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Reposito...	SUCCESS	Offset: 23,044,096,...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Reposito...	SUCCESS	Offset: 21,880,832,...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Reposito...	SUCCESS	Offset: 21,692,416,...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Reposito...	SUCCESS	Offset: 21,651,456,...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Reposito...	SUCCESS	Offset: 21,889,024,...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Reposito...	SUCCESS	Offset: 22,036,480,...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Reposito...	SUCCESS	Offset: 23,543,808,...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Reposito...	SUCCESS	Offset: 21,790,720,...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Reposito...	SUCCESS	Offset: 21,774,336,...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Reposito...	SUCCESS	Offset: 21,954,560,...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Reposito...	SUCCESS	Offset: 21,643,264,...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Reposito...	SUCCESS	Offset: 20,332,544,...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Reposito...	SUCCESS	Offset: 21,757,952,...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Reposito...	SUCCESS	Offset: 21,921,792,...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Reposito...	SUCCESS	Offset: 21,831,680,...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	ReadFile	C:\Windows\System32\wbem\Reposito...	SUCCESS	Offset: 21,848,064,...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	RegOpenKey	HKLM	SUCCESS	Desired Access: M...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	RegQueryKey	HKLM	SUCCESS	Query: HandleTag...	5/25/2021 12:42:...	C:\Windows\syste...
12:42:...	svchost.exe	0	3132	64-bit	RegOpenKey	HKLM\system\Setup	SUCCESS	Desired Access: R...	5/25/2021 12:42:...	C:\Windows\syste...

Showing 125,034 of 366,792 events (34%) Backed by virtual memory


Summary: 2375 Calls

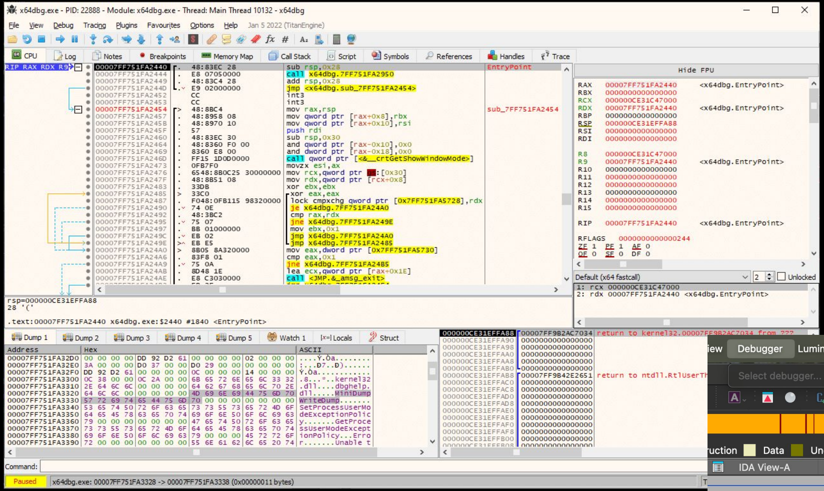
#	TID	Module	API	Return	Error
1745	6716	kernel32.dll	NtQueryInformationProcess (GetCurrentProcess(), ProcessDefaultHa...	STATUS_SUCCESS	
1746	6716	kernel32.dll	NtSetInformationProcess (GetCurrentProcess(), ProcessDefaultHardE...	STATUS_SUCCESS	
1747	6716	kernel32.dll	NtQueryInformationProcess (GetCurrentProcess(), ProcessDefaultHa...	STATUS_SUCCESS	
1748	6716	kernel32.dll	NtSetInformationProcess (GetCurrentProcess(), ProcessDefaultHardE...	STATUS_SUCCESS	
1749	6716	VERSION.dll	CreateFileW ("C:\Programming\TortoiseSVN\Languages\TortoisePro...	INVALID_HANDLE_VALUE	2 = The system ca
1750	6716	kernel32.dll	NtCreateFile (0x000000000028b4b8, GENERIC_READ | SYNCHRON...	STATUS_OBJECT_NAME_NOT_F...	0xc0000034 = Ob
1751	6716	kernel32.dll	Lc	...OUND	0xc0000135 = {Un
1752	6716	kernel32.dll	Lc		
1753	6716	kernel32.dll	Lc		
1754	6716	kernel32.dll	Lc		
1755	6716	kernel32.dll	Lc		
1756	6716	apphelp.dll	N		
1757	6716	apphelp.dll	N		
1758	6716	apphelp.dll	NtClose (0x00000000000001a0)	STATUS_SUCCESS	
1759	6716	apphelp.dll	LdrGetDllHandle (NULL, NULL, 0x000000000028da80, 0x00000000002...	STATUS_SUCCESS	
1760	6716	ole32.dll	NtQueryKey (0x00000000000001a0, KeyNameInformation, NULL, 0, 0x...	STATUS_BUFFER_TOO_SMALL	0xc0000023 = {Bu
1761	6716	ole32.dll	NtQueryKey (0x00000000000001a0, KeyNameInformation, 0x0000000...	STATUS_SUCCESS	

```
CreateFileW (
    0x000000000028b830 "C:\Programming\TortoiseSVN\Languages\TortoiseProc1033.dll",
    GENERIC_READ,
    FILE_SHARE_READ,
    NULL,
    OPEN_EXISTING,
    FILE_ATTRIBUTE_NORMAL,
    NULL
);
```

Debugging on Windows (Demo)

https://x64dbg.com
- GUI based Windows Debugger

Your Very own Decompiler !
- IDA/Binja support native debugging and I personally really like it


x64dbg.exe - PID: 22888 - Module: x64dbg.exe - Thread: Main Thread 10132 - x64dbg

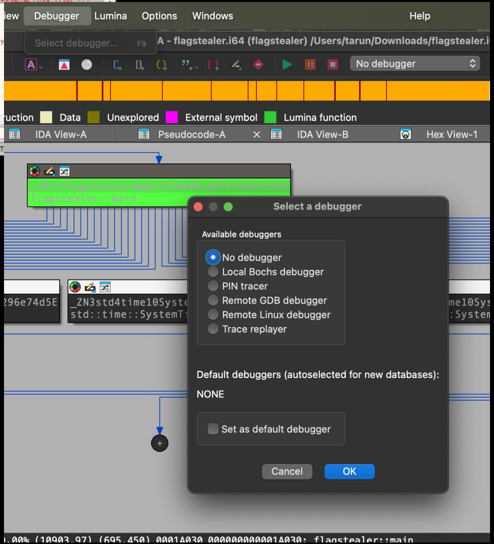
```
00007FF751FA2440   48:83EC 28            sub rsp,0x28                                  EntryPoint
00007FF751FA2444   E8 07050000           call x64dbg.7FF751FA2950
00007FF751FA2449   48:83C4 28            add rsp,0x28
00007FF751FA244D   E9 02000000           jmp <x64dbg.sub_7FF751FA2454>
00007FF751FA2452   CC                    int3
00007FF751FA2453   CC                    int3
00007FF751FA2454   48:8BC4               mov rax,rsp                                   sub_7FF751FA2454
00007FF751FA2457   48:8958 08            mov qword ptr [rax+0x8],rbx
00007FF751FA245B   48:8970 10            mov qword ptr [rax+0x10],rsi
00007FF751FA245F   57                    push rdi
00007FF751FA2460   48:83EC 30            sub rsp,0x30
00007FF751FA2464   48:8360 F0 00         and qword ptr [rax-0x10],0x0
00007FF751FA2469   8360 E8 00            and dword ptr [rax-0x18],0x0
00007FF751FA246D   FF15 1D0D0000         call qword ptr [<&__crtGetShowWindowMode>]
00007FF751FA2473   0FB7F0                movzx esi,ax
00007FF751FA2476   6548:8B0C25 30000000  mov rcx,qword ptr gs:[0x30]
00007FF751FA247F   48:8B51 08            mov rdx,qword ptr [rcx+0x8]
00007FF751FA2483   33DB                  xor ebx,ebx
00007FF751FA2485   33C0                  xor eax,eax
00007FF751FA2487   F048:0FB115 98320000  lock cmpxchg qword ptr [0x7FF751FA5728],rdx
00007FF751FA2490   74 0E                 je x64dbg.7FF751FA24A0
00007FF751FA2492   48:3BC2               cmp rax,rdx
00007FF751FA2495   75 07                 jne x64dbg.7FF751FA249E
00007FF751FA2497   BB 01000000           mov ebx,0x1
00007FF751FA249C   EB 02                 jmp x64dbg.7FF751FA24A0
00007FF751FA249E   EB E5                 jmp x64dbg.7FF751FA2485
00007FF751FA24A0   8B05 8A320000         mov eax,dword ptr [0x7FF751FA5730]
00007FF751FA24A6   83F8 01               cmp eax,0x1
00007FF751FA24A9   75 0A                 jne x64dbg.7FF751FA24B5
00007FF751FA24AB   8D48 1E               lea ecx,qword ptr [rax+0x1E]
00007FF751FA24AE   E8 C3030000           call <JMP.&_amsg_exit>

rsp=000000CE31EFFA88
28 '('

.text:00007FF751FA2440 x64dbg.exe:$2440 #1840 <EntryPoint>
```

```
RAX    00007FF751FA2440    <x64dbg.EntryPoint>
RBX    0000000000000000
RCX    000000CE31C47000
RDX    00007FF751FA2440    <x64dbg.EntryPoint>
RBP    0000000000000000
RSP    000000CE31EFFA88
RSI    0000000000000000
RDI    0000000000000000

R8     000000CE31C47000
R9     00007FF751FA2440    <x64dbg.EntryPoint>
R10    0000000000000000
R11    0000000000000000
R12    0000000000000000
R13    0000000000000000
R14    0000000000000000
R15    0000000000000000

RIP    00007FF751FA2440    <x64dbg.EntryPoint>

RFLAGS   0000000000000244
ZF 1  PF 1  AF 0
OF 0  SF 0  DF 0

Default (x64 fastcall)   2   Unlocked
1: rcx 000000CE31C47000
2: rdx 00007FF751FA2440 <x64dbg.EntryPoint>
```

```
Address           Hex                                              ASCII
00007FF751FA32D0  00 00 00 00 DD 92 D2 61 00 00 00 00 02 00 00 00  ....Ý.Òa........
00007FF751FA32E0  3A 00 00 00 D0 37 00 00 D0 29 00 00 00 00 00 00  :...Ð7..Ð)......
00007FF751FA32F0  DD 92 D2 61 00 00 00 00 0C 00 00 00 14 00 00 00  Ý.Òa............
00007FF751FA3300  0C 38 00 00 0C 2A 00 00 6B 65 72 6E 65 6C 33 32  .8...*..kernel32
00007FF751FA3310  2E 64 6C 6C 00 00 00 00 64 62 67 68 65 6C 70 2E  .dll....dbghelp.
00007FF751FA3320  64 6C 6C 00 00 00 00 00 4D 69 6E 69 44 75 6D 70  dll.....MiniDump
00007FF751FA3330  57 72 69 74 65 44 75 6D 70 00 00 00 00 00 00 00  WriteDump.......
00007FF751FA3340  53 65 74 50 72 6F 63 65 73 73 55 73 65 72 4D 6F  SetProcessUserMo
00007FF751FA3350  64 65 45 78 63 65 70 74 69 6F 6E 50 6F 6C 69 63  deExceptionPolic
00007FF751FA3360  79 00 00 00 00 00 00 00 47 65 74 50 72 6F 63 65  y.......GetProce
00007FF751FA3370  73 73 55 73 65 72 4D 6F 64 65 45 78 63 65 70 74  ssUserModeExcept
00007FF751FA3380  69 6F 6E 50 6F 6C 69 63 79 00 00 00 45 72 72 6F  ionPolicy...Erro
00007FF751FA3390  72 00 00 00 00 00 00 00 55 6E 61 62 6C 65 20 74  r.......Unable t
```

```
000000CE31EFFA88  00007FF9B2AC7034  return to kernel32.00007FF9B2AC7034 from ???
000000CE31EFFA90  0000000000000000
000000CE31EFFA98  0000000000000000
000000CE31EFFAA0  0000000000000000
000000CE31EFFAA8  0000000000000000
000000CE31EFFAB0  0000000000000000
000000CE31EFFAB8  00007FF9B42E2651  return to ntdll.RtlUserTh
000000CE31EFFAC0  0000000000000000
000000CE31EFFAC8  0000000000000000
000000CE31EFFAD0  0000000000000000
000000CE31EFFAD8  0000000000000000
000000CE31EFFAE0  0000000000000000
000000CE31EFFAE8  0000000000000000
000000CE31EFFAF0  0000000000000000
000000CE31EFFAF8  0000000000000000
000000CE31EFFB00  0000000000000000
```

```
Command:
Paused   x64dbg.exe: 00007FF751FA3328 -> 00007FF751FA3338 (0x00000011 bytes)
```


A - flagstealer.i64 (flagstealer) /Users/tarun/Downloads/flagstealer.i

Select a debugger

Available debuggers
- [✓ No debugger]
- Local Bochs debugger
- PIN tracer
- Remote GDB debugger
- Remote Linux debugger
- Trace replayer

Default debuggers (autoselected for new databases):

NONE

Set as default debugger

```
.00% (10903,97) (695,450) 0001A030 000000000001A030: flagstealer::main
```

Debuggers Overview

Standard Debugger - GDB

Talks to the OS using ptrace syscall

- Ptrace allows one process (gdb) control another process (your code)

Ptrace allows gdb to:

- to pause, resume program
- inspect/change register
- read/write process memory
- Insert breakpoints by replacing instructions with int3 0xCC

Very useful for figuring out what a program is doing on the file and validating your understanding of code execution.

NOTE: Watch out for anti debugger checks - raise(SIGTRAP), self debugging, check for 0xCC, time checks, IsDebuggerPresent()

Pwndbg

Install

- curl -qsL 'https://install.pwndbg.re' | sh -s -- -t pwndbg-gdb
- brew install pwndbg/tap/pwndbg-gdb

https://pwndbg.re/pwndbg/latest/commands

What is it?

- GDB and LLDB plug-in, created for exploitation by ctfers
- Created by Zach Riggle GPZ
- Maintained by disconnect3d (ToB)
- Very feature Dense

Pwndbg Basics

file <file>: Load in you are trying to debug

Process <pid>: load in running process

Shell <commands>: run shell commands inside pwndbg

info functions <name>: get list of functions that you can breakpoint on

- The system has auto “grep” where if you search for a name it will filter for it just type in the name right after info functions.

Entry: start the program, stops right at the start

ni <x>: next instruction, moves to the next instruction will jump over the previous

si <x>: step instruction, moves into the next instruction will follow calls

- Note you can add a number to say move this many times

Vmmap: Shows you the memory mappings of your program, useful for getting rebase info

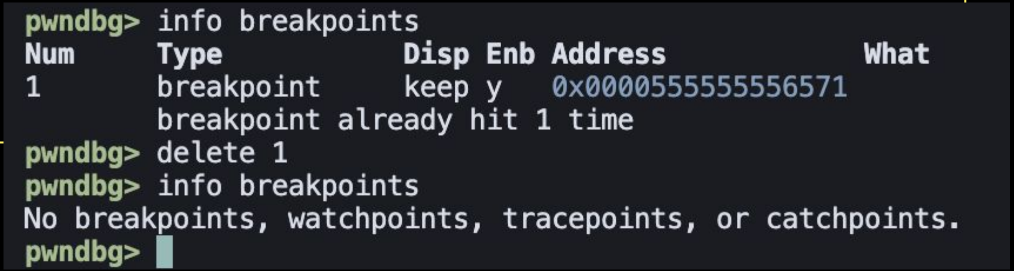
Pwndbg Basics cnt

breakpoint/b <address | name | file:linenumber>: If you have debug symbols you can use function names, or the filename:linenumber (me.c:123) - this breaks at the start of the line's basic block. Function name will break at the start of a function entry point. Address based breakpoints will still exactly where you told to.

info breakpoints : shows you all the breakpoints you currently have set and number of times hit

delete <number> : removes any breakpoint

watch <addres> : If you just want to see how many times an addr is reached



```
pwndbg> info breakpoints
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x0000555555556571
        breakpoint already hit 1 time
pwndbg> delete 1
pwndbg> info breakpoints
No breakpoints, watchpoints, tracepoints, or catchpoints.
pwndbg>
```

Pwndbg Printing Stuff

Info registers : print all registers

p $<reg> : print value of specific register

telescope $rsp/<addr> : inspect stack contents from stack pointer or addr

x/10gx <addr> : examine memory 10 giant words hex

x/s <addr> : print string at address

x/<n>i $rip : show n instructions at instruction ptr

disassemble <func> :

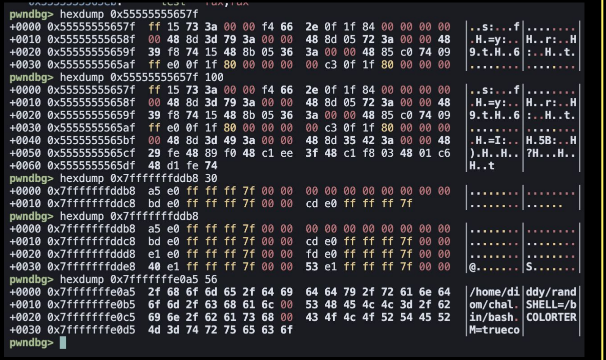
p var :

P *(int *) <adrr>:

p/(d | x | c) $<reg> : print register in decimal/hex/character

bt: backstrace

Hexdump <address> <n>: dump memory at address for n amount using hexdump format



```
pwndbg> hexdump 0x55555555657f
+0000 0x55555555657f  ff 15 73 3a 00 00 f4 66  2e 0f 1f 84 00 00 00 00  │..s:...f│........│
+0010 0x55555555658f  00 48 8d 3d 79 3a 00 00  48 8d 05 72 3a 00 00 48  │.H.=y:..│H..r:..H│
+0020 0x55555555659f  39 f8 74 15 48 8b 05 36  3a 00 00 48 85 c0 74 09  │9.t.H..6│:..H..t.│
+0030 0x5555555565af  ff e0 0f 1f 80 00 00 00  00 c3 0f 1f 80 00 00 00  │........│........│
pwndbg> hexdump 0x55555555657f 100
+0000 0x55555555657f  ff 15 73 3a 00 00 f4 66  2e 0f 1f 84 00 00 00 00  │..s:...f│........│
+0010 0x55555555658f  00 48 8d 3d 79 3a 00 00  48 8d 05 72 3a 00 00 48  │.H.=y:..│H..r:..H│
+0020 0x55555555659f  39 f8 74 15 48 8b 05 36  3a 00 00 48 85 c0 74 09  │9.t.H..6│:..H..t.│
+0030 0x5555555565af  ff e0 0f 1f 80 00 00 00  00 c3 0f 1f 80 00 00 00  │........│........│
+0040 0x5555555565bf  00 48 8d 3d 49 3a 00 00  48 8d 35 42 3a 00 00 48  │.H.=I:..│H.5B:..H│
+0050 0x5555555565cf  29 fe 48 89 f0 48 c1 ee  3f 48 c1 f8 03 48 01 c6  │).H..H..│?H...H..│
+0060 0x5555555565df  48 d1 fe 74                                       │H..t    │        │
pwndbg> hexdump 0x7fffffffddb8 30
+0000 0x7fffffffddb8  a5 e0 ff ff ff 7f 00 00  00 00 00 00 00 00 00 00  │........│........│
+0010 0x7fffffffddc8  bd e0 ff ff ff 7f 00 00  cd e0 ff ff ff 7f        │........│......  │
pwndbg> hexdump 0x7fffffffddb8
+0000 0x7fffffffddb8  a5 e0 ff ff ff 7f 00 00  00 00 00 00 00 00 00 00  │........│........│
+0010 0x7fffffffddc8  bd e0 ff ff ff 7f 00 00  cd e0 ff ff ff 7f 00 00  │........│........│
+0020 0x7fffffffddd8  e1 e0 ff ff ff 7f 00 00  fd e0 ff ff ff 7f 00 00  │........│........│
+0030 0x7fffffffdde8  40 e1 ff ff ff 7f 00 00  53 e1 ff ff ff 7f 00 00  │@.......│S.......│
pwndbg> hexdump 0x7fffffffe0a5 56
+0000 0x7fffffffe0a5  2f 68 6f 6d 65 2f 64 69  64 64 79 2f 72 61 6e 64  │/home/di│ddy/rand│
+0010 0x7fffffffe0b5  6f 6d 2f 63 68 61 6c 00  53 48 45 4c 4c 3d 2f 62  │om/chal.│SHELL=/b│
+0020 0x7fffffffe0c5  69 6e 2f 62 61 73 68 00  43 4f 4c 4f 52 54 45 52  │in/bash.│COLORTER│
+0030 0x7fffffffe0d5  4d 3d 74 72 75 65 63 6f                           │M=trueco│        │
pwndbg> 
```

Frida, how to become a hooker !

Install: pip install frida-tools

Interceptor and Stalker are the two main frida API's for analysis

Stalker:

Used to trace basics blocks and pre instruction level execution or large scale execution.

How it works?

Code tracing based on dynamic recompilation where you are more or less compiling the code at runtime. The running program machine code is copied into a local copy that is modified and as needed and then executed

- This logic allows to maintain the original checksum while still being able to trace the execution

https://github.com/marcosatti/Dynarec_Guide

https://medium.com/@oleavr/anatomy-of-a-code-tracer-b081aadb0df8

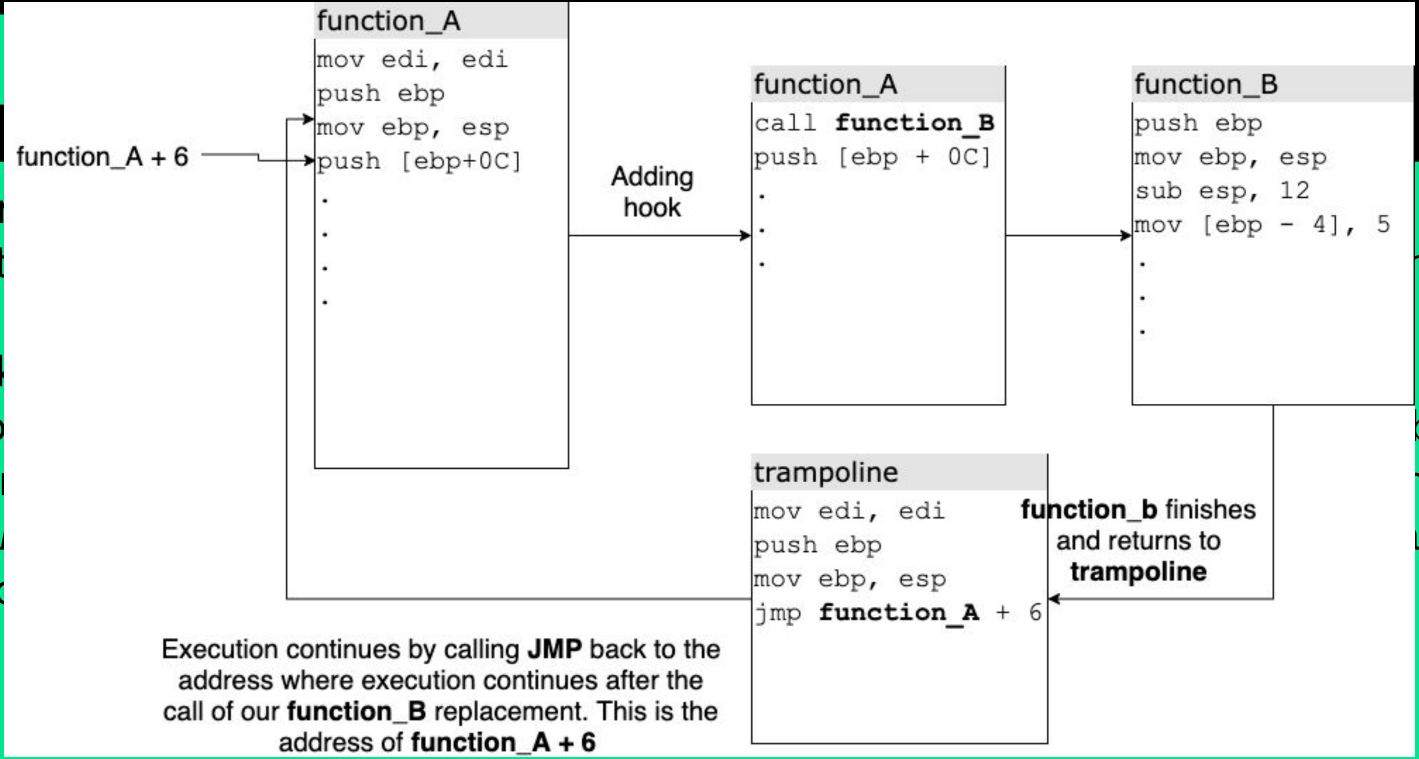
Frida,

Interceptor

The Int nd
either

How it work

The co by
inserti hich
then w ted
functi



```
function_A
mov edi, edi
push ebp
mov ebp, esp
push [ebp+0C]
.
.
.
.
```

```
function_A
call function_B
push [ebp + 0C]
.
.
.
```

```
function_B
push ebp
mov ebp, esp
sub esp, 12
mov [ebp - 4], 5
.
.
.
```

```
trampoline
mov edi, edi
push ebp
mov ebp, esp
jmp function_A + 6
```

Execution continues by calling **JMP** back to the address where execution continues after the call of our **function_B** replacement. This is the address of **function_A + 6**

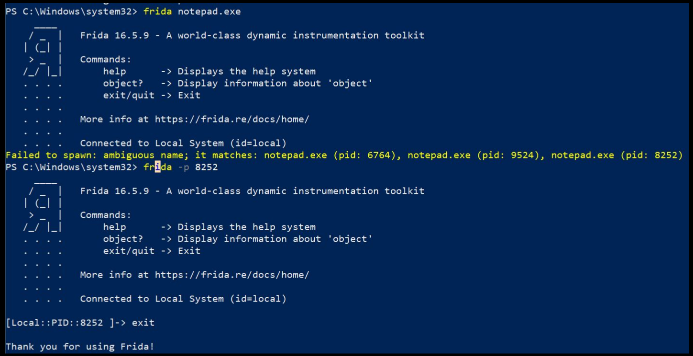
Example:

If function_A is executing wants to call function_B we will hijack function_A prologue and replaced with a JMP instruction to our function_B. Once out function_B is executed, the goes back to the intended function_A execution flow.

Frida CLI Tool

`> frida -p <Process ID / Process Name>`

`> frida <Process ID / Process Name>`



```
PS C:\Windows\system32> frida notepad.exe
     ____
    / _  |   Frida 16.5.9 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://frida.re/docs/home/
   . . . .
   . . . .   Connected to Local System (id=local)
Failed to spawn: ambiguous name; it matches: notepad.exe (pid: 6764), notepad.exe (pid: 9524), notepad.exe (pid: 8252)
PS C:\Windows\system32> frida -p 8252
     ____
    / _  |   Frida 16.5.9 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://frida.re/docs/home/
   . . . .
   . . . .   Connected to Local System (id=local)

[Local::PID::8252 ]-> exit

Thank you for using Frida!
```

If the executable isn't running then use the `-f` flag to start the binary.

If you want to load a script you use the `-l`

Frida-trace : This is a rather simple tool that allows us to see function calls that are being made from a process.

`frida-trace -p <pid> -I <Module Name>`

`frida-trace -p <pid> -i "<name>.dll!*<exported func>*"`

`frida-trace -p <pid> -a "<name>.dll![offset]"`

Frida Scripting

Interceptor: inline function hooks

- Find targets (export, offset, DebugSymbol)
- Log args/retvals, dump memory, change values, replace impls

Interceptor API:

`Interceptor.attach(target, callbacks[, data])`

First step is going to be finding `target`

Export by Name: `const target = Module.getExportByName('libc.so', 'read'),`

Export by Module + Offset:

```
const mod = Process.findModuleByName('Library_Name');
const target = mod.base.add(FUNCTION_OFFSET);
```

Frida Scripting Interceptor target

First step is going to be finding target

Debug Symbols:

```
DebugSymbol.load('some_library');
var Address = DebugSymbol.getFunctionByName('proper_debug_name');
Interceptor.attach(Address, { onEnter(args){}, onLeave(r){} });
```

Frida Scripting Interceptor

```javascript
let interceptor = Interceptor.attach(target, {
  onEnter(args) {
    const connHex = analyzeLDAPConn(args[0]);
    const reqHex  = analyzeLDAPRequest(args[1]);
    if (!args[2].isNull()) {
      console.log("\nInt32 Parameter:");
      dumpMemoryDetailed(args[2], 0x10, "Int32 Value");
    }
    this.savedData = { connHex, reqHex };
  },
  onLeave(retval) {
    console.log(`\n[*] Function returned: ${retval}`);
    // retval.replace(X)  // mutate if needed
  }
});
```

Stalker

```
function main() {
  console.log("Starting stalker with events");
  const threadId = Process.getCurrentThreadId();
  Stalker.follow(threadId, {
    events: { call: true, ret: true, exec: false, block: false, compile: false },
    onCallSummary(summary) {
      console.log('\nCall Summary:');
      for (const [target, count] of Object.entries(summary)) {
        try {
          const symbol = DebugSymbol.fromAddress(ptr(target));
          console.log(`    ${symbol.toString()} - called ${count} times`);
        } catch (e) {
          console.log(`    ${target} - called ${count} times`);
        }
      }
    }
  });
  console.log("Stalker attached");
}
setImmediate(main);
```

Stalker

```
function main() {
  const threadId = Process.getCurrentThreadId();
  Stalker.follow(threadId, {
    events: { call: true, ret: false, exec: false, block: true, compile: false },
    onReceive(events) {
      const parsed = Stalker.parse(events);
      for (const event of parsed) {
        if (Array.isArray(event)) {
          const eventType = event[0];
          if (eventType === 'block') {
            const blockStart = event[1], blockEnd = event[2];
            console.log(`Block @ ${blockStart} size: ${Number(blockEnd)-Number(blockStart)} bytes`);
          } else if (eventType === 'call') {
            const from = event[1], to = event[2];
            console.log(`Call from ${from} to ${to}`);
          }
        }
      }
    }
  });
}
setImmediate(main);
RAW DATA: block,0x...6b7,0x...6bf,call,0x...6ba,0x...69e4,19,block,0x...6bf,0x...6c6,...
```

Stalker

```
transform(iterator) {
  let insn = iterator.next();
  const startAddress = insn.address;
  const isAppCode = startAddress.compare(appStart) >= 0 && startAddress.compare(appEnd) === -1;
  const canEmit = iterator.memoryAccess === 'open';

  do {
    if (isAppCode && canEmit && insn.mnemonic === 'ret') {
      iterator.putCmpRegI32('eax', 60);
      iterator.putJccShortLabel('jb', 'nope', 'no-hint');
      iterator.putCmpRegI32('eax', 90);
      iterator.putJccShortLabel('ja', 'nope', 'no-hint');
      iterator.putCallout(onMatch);
      iterator.putLabel('nope');
    }
    iterator.keep();
  } while ((insn = iterator.next()) !== null);
}
```

LD_PRELOAD

Usd to to intercept and override library function calls at runtime

- This means any dynamically linked lib function call be hijacked this is pretty much adjacent to dll hijacking in windows

How this work ?

1. When you run a dynamiically linked binary the loader ld-linux.so will resolve function symbols from shared libs.
2. If LD_PRELOAD the loader injects your custom .so before standard libraries like libc
3. Symbol resolution happens in order, which results in our functions getting called instead of a standard lib function
4. You can fully rewrite the code or say add logging + print statements and then send it to the real function via dlsym(RTLD_NEXT, ”func”)

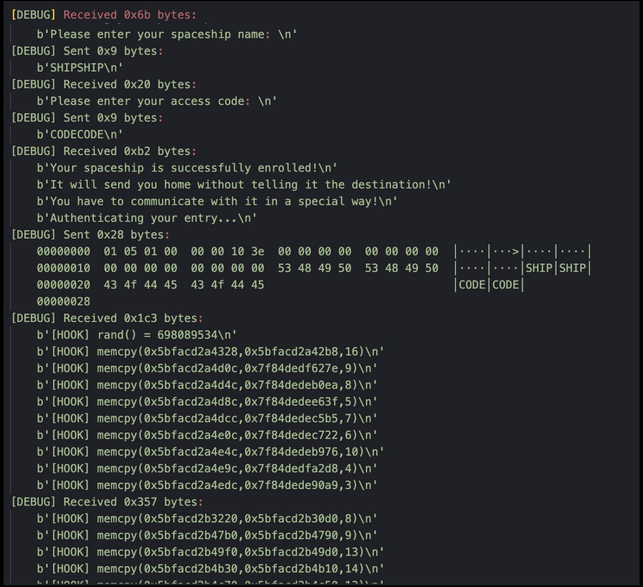
LD_PRELOAD

```c
#define _GNU_SOURCE
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <dlfcn.h>

int memcmp(const void *s1, const void *s2, size_t n) {
    static int (*real_memcmp)(const void *, const void *, size_t) = NULL;
    if (!real_memcmp) real_memcmp = dlsym(RTLD_NEXT, "memcmp");
    int ret = real_memcmp(s1, s2, n);
    printf("[HOOK] memcmp(%p,%p,%zu) = %d\n", s1, s2, n, ret);
    return ret;
}

void *memcpy(void *dest, const void *src, size_t n) {
    static void *(*real_memcpy)(void *, const void *, size_t) = NULL;
    if (!real_memcpy) real_memcpy = dlsym(RTLD_NEXT, "memcpy");
    void *ret = real_memcpy(dest, src, n);
    printf("[HOOK] memcpy(%p,%p,%zu)\n", dest, src, n);
    return ret;
}

int rand(void) {
    static int (*real_rand)(void) = NULL;
    if (!real_rand) real_rand = dlsym(RTLD_NEXT, "rand");
    int ret = real_rand();
    printf("[HOOK] rand() = %d\n", ret);
    return ret;
}
```

Building and using LD_PRELOAD

Complied with `gcc -shared -fPIC -o hook.so hook.c -ldl`

Complied with `LD_PRELOAD=./hook.so ./chal`



```
[DEBUG] Received 0x6b bytes:
    b'Please enter your spaceship name: \n'
[DEBUG] Sent 0x9 bytes:
    b'SHIPSHIP\n'
[DEBUG] Received 0x20 bytes:
    b'Please enter your access code: \n'
[DEBUG] Sent 0x9 bytes:
    b'CODECODE\n'
[DEBUG] Received 0xb2 bytes:
    b'Your spaceship is successfully enrolled!\n'
    b'It will send you home without telling it the destination!\n'
    b'You have to communicate with it in a special way!\n'
    b'Authenticating your entry...\n'
[DEBUG] Sent 0x28 bytes:
    00000000  01 05 01 00  00 00 10 3e  00 00 00 00  00 00 00 00  │····│···>│····│····│
    00000010  00 00 00 00  00 00 00 00  53 48 49 50  53 48 49 50  │····│····│SHIP│SHIP│
    00000020  43 4f 44 45  43 4f 44 45                            │CODE│CODE│
    00000028
[DEBUG] Received 0x1c3 bytes:
    b'[HOOK] rand() = 698089534\n'
    b'[HOOK] memcpy(0x5bfacd2a4328,0x5bfacd2a42b8,16)\n'
    b'[HOOK] memcpy(0x5bfacd2a4d0c,0x7f84dedf627e,9)\n'
    b'[HOOK] memcpy(0x5bfacd2a4d4c,0x7f84dedeb0ea,8)\n'
    b'[HOOK] memcpy(0x5bfacd2a4d8c,0x7f84dedee63f,5)\n'
    b'[HOOK] memcpy(0x5bfacd2a4dcc,0x7f84dedec5b5,7)\n'
    b'[HOOK] memcpy(0x5bfacd2a4e0c,0x7f84dedec722,6)\n'
    b'[HOOK] memcpy(0x5bfacd2a4e4c,0x7f84dedeb976,10)\n'
    b'[HOOK] memcpy(0x5bfacd2a4e9c,0x7f84dedfa2d8,4)\n'
    b'[HOOK] memcpy(0x5bfacd2a4edc,0x7f84dede90a9,3)\n'
[DEBUG] Received 0x357 bytes:
    b'[HOOK] memcpy(0x5bfacd2b3220,0x5bfacd2b30d0,8)\n'
    b'[HOOK] memcpy(0x5bfacd2b47b0,0x5bfacd2b4790,9)\n'
    b'[HOOK] memcpy(0x5bfacd2b49f0,0x5bfacd2b49d0,13)\n'
    b'[HOOK] memcpy(0x5bfacd2b4b30,0x5bfacd2b4b10,14)\n'
```

Definitive Plan

This is going to be our teams definitive schedule for the next few weeks. If you have any Comments or points of concencer let me know !

Sept 22nd
- Today !

Sept 29th
- Ida and Binja Scripting

October 6th
- Magnet Forensics

October 20th
- Mobile Applications

October 27th
- Constraint Solving and Symbex

November
- Tizen Js engine rev (?) or Windows Kernel (?)