# 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

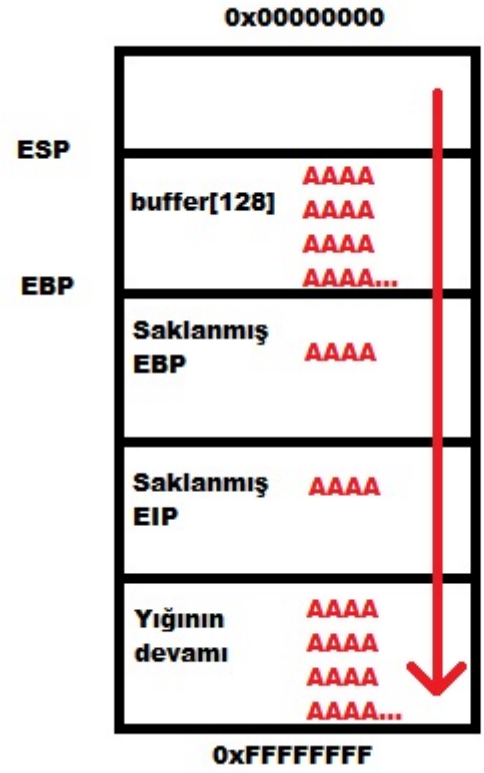
dönüş adresi) tutulabilir, saklanabilir.

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.

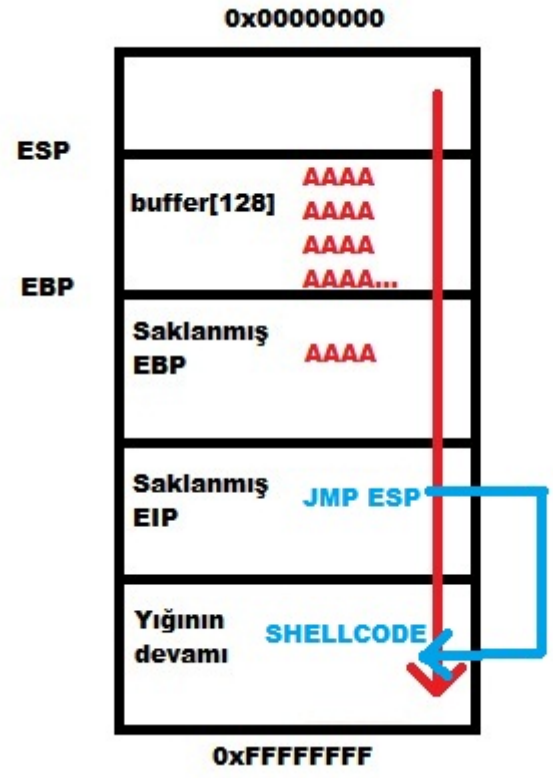
Örnek olarak A ve B fonksiyonundan oluşan bir program düşünelim ve A fonksiyonu içinden B fonksiyonunun çağrıldığını ve 128 byte büyüklüğündeki bir belleğe (array) 136 byte uzunluğunda ve ‘A’ (0x41 hex değeri) karakterinden oluşan bir dizini (string) kopyaladığımızı varsayalım.

```
#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)a arabellek taşması zafiyetini istismar edebiliriz.

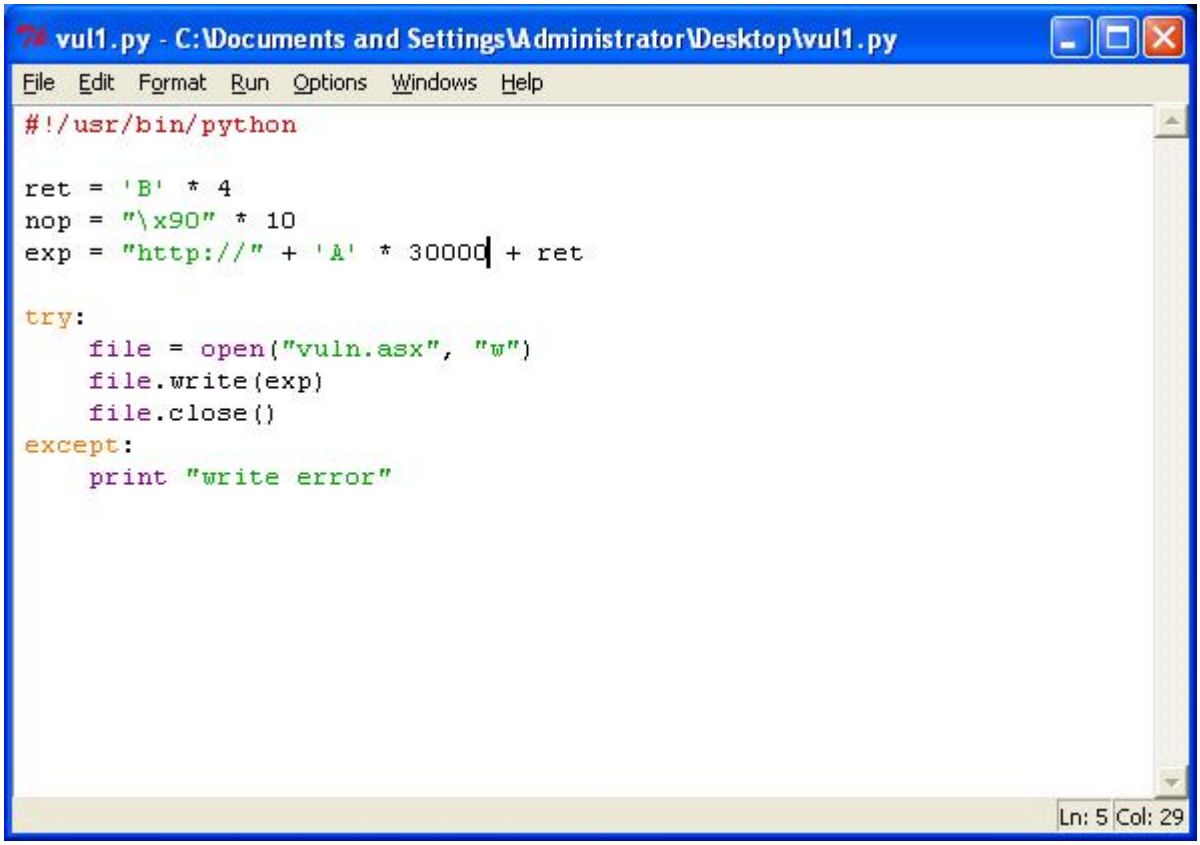


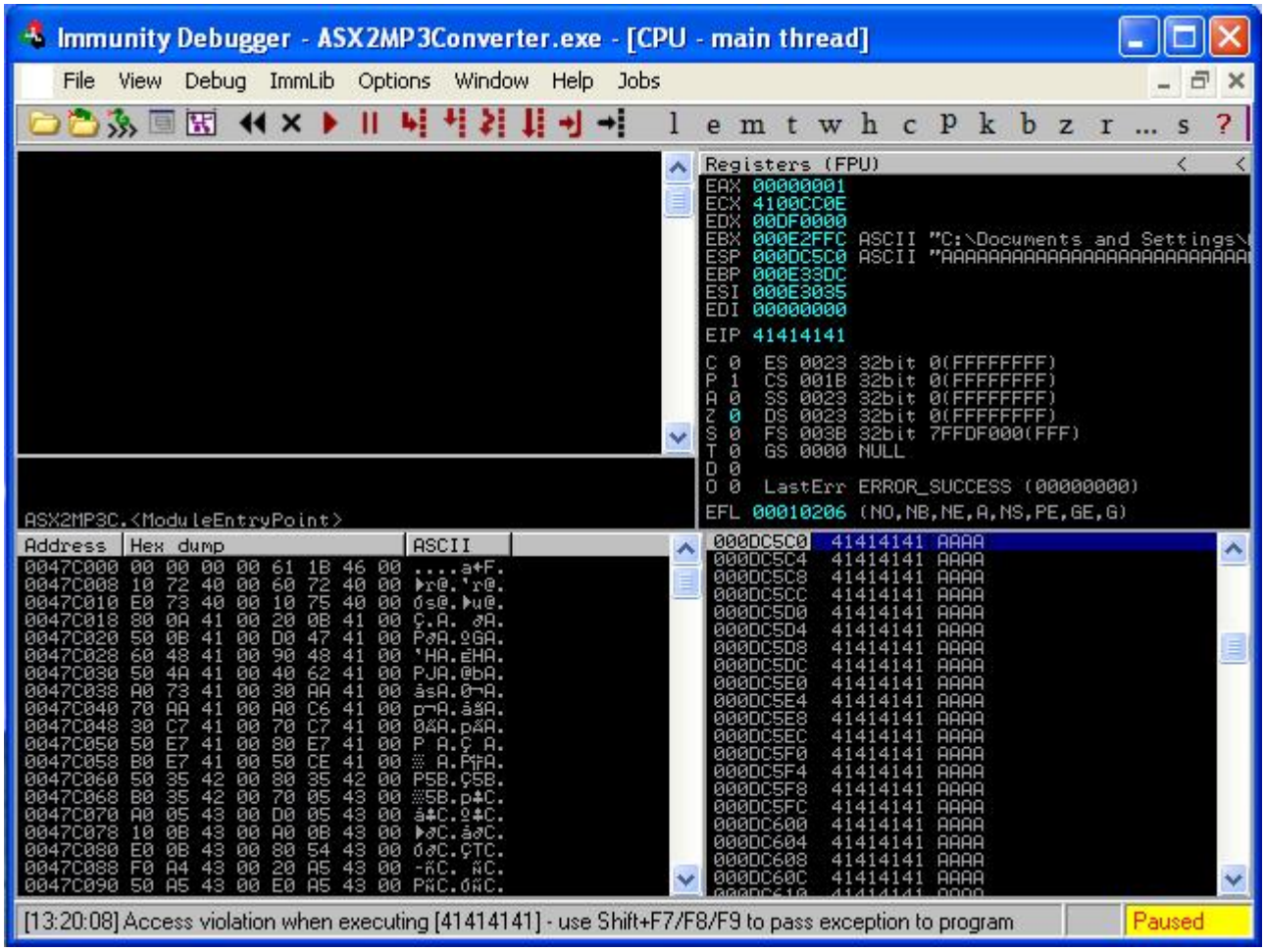

Yığın tabanlı arabellek taşması zafiyeti ve istismar edilmesi kısaca ve

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

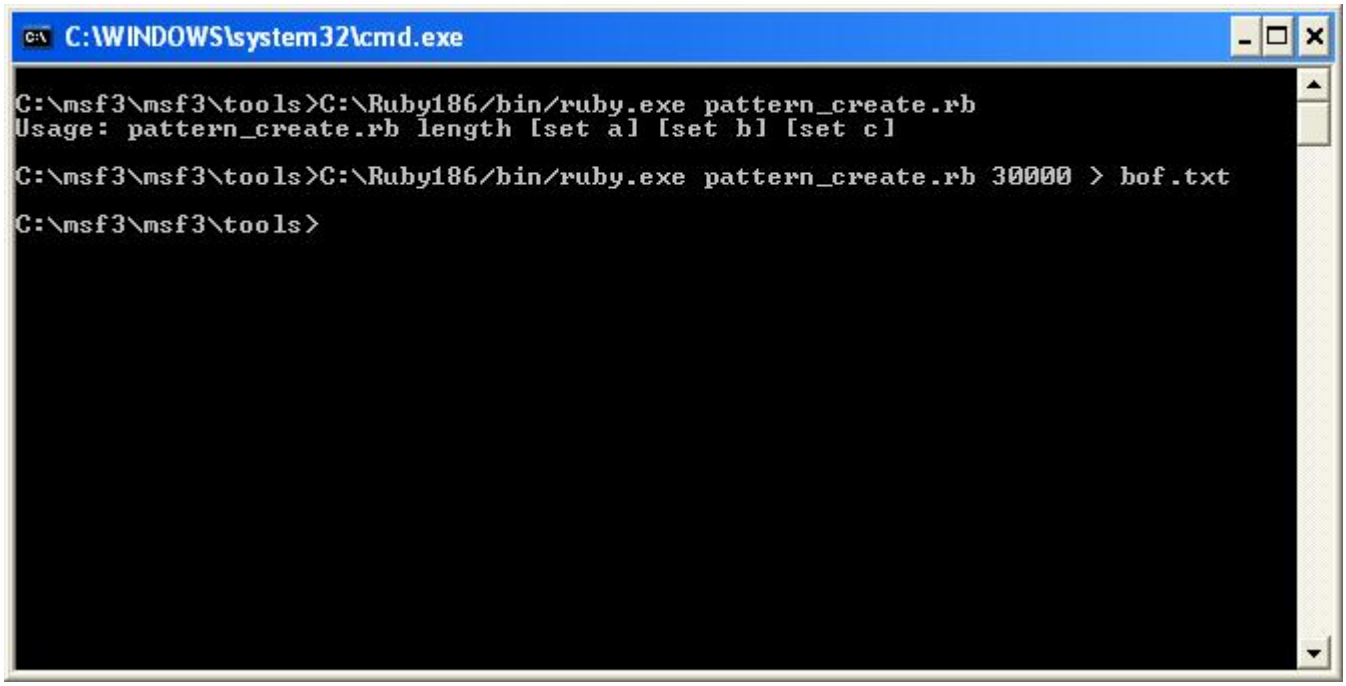
Örnek olarak istismar edeceğimiz programın adı ASX to MP3 Converter. CVE-2009-1642 numaralı CVE ID'sine göre bu programın 3.0.0.7 sürümünde asx uzantılı dosyalarda kullanılan HREF nitelemesinde (attribute) yığın tabanlı arabellek taşması zafiyeti olduğu belirtiliyor.

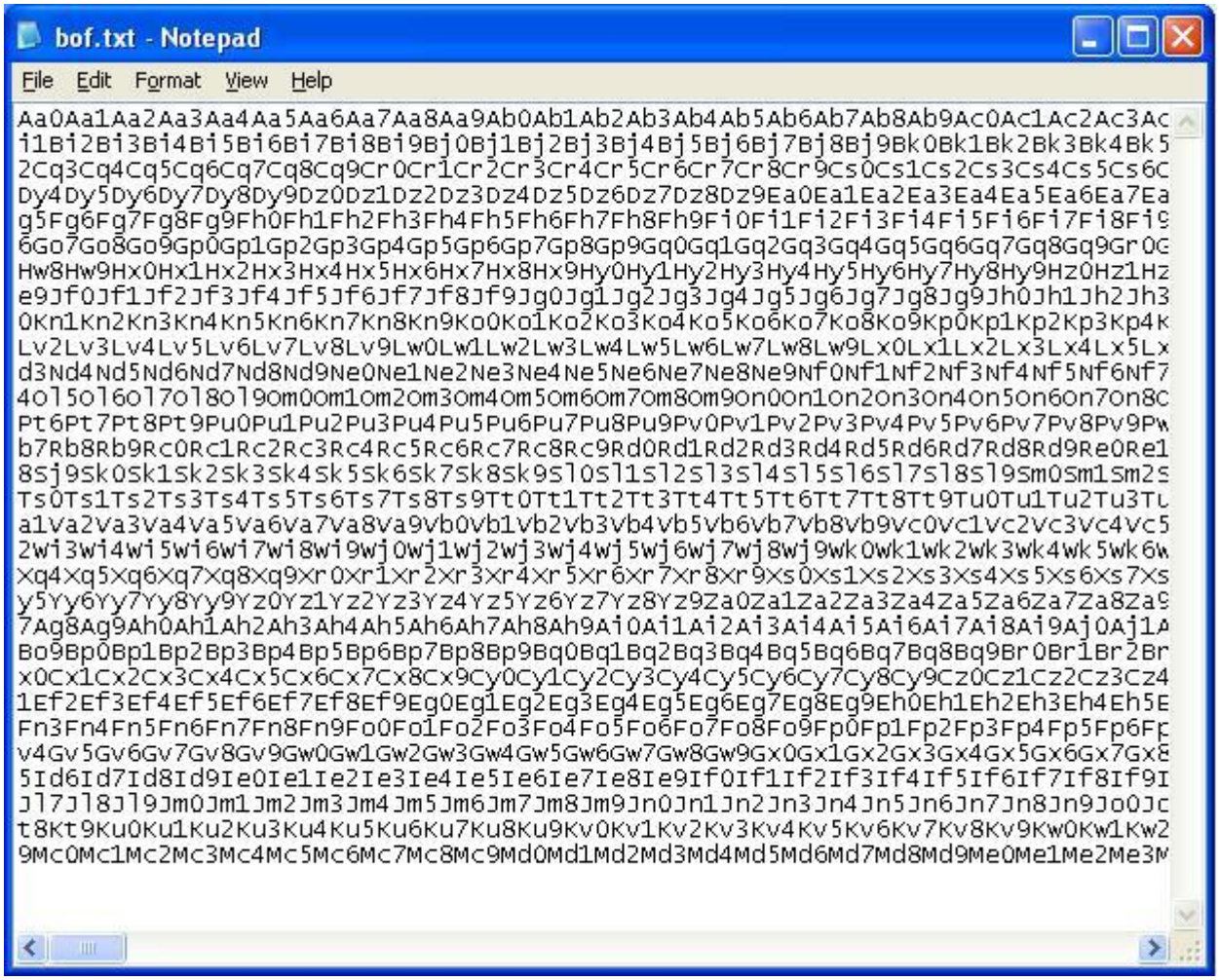
Bu açıklama üzerine asx uzantılı bir dosya yaratan ve içine "http://AAAAAAAAAAA (30000 tane)" dizisi kopyalayan ufak bir program hazırlıyoruz ve daha sonrasında programı çalıştırdığımızda EIP kaydedicisine müdahale ederek zafiyetin varlığını teyit edebiliyoruz.

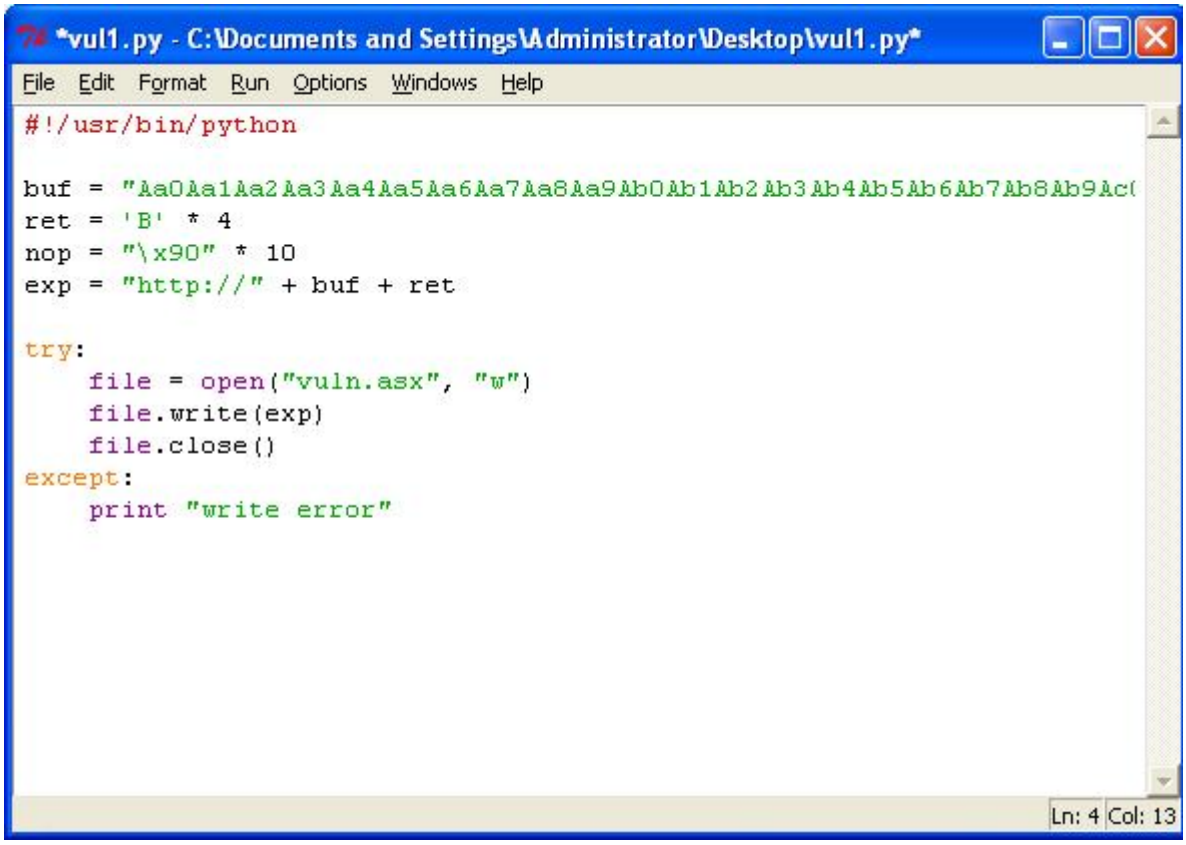


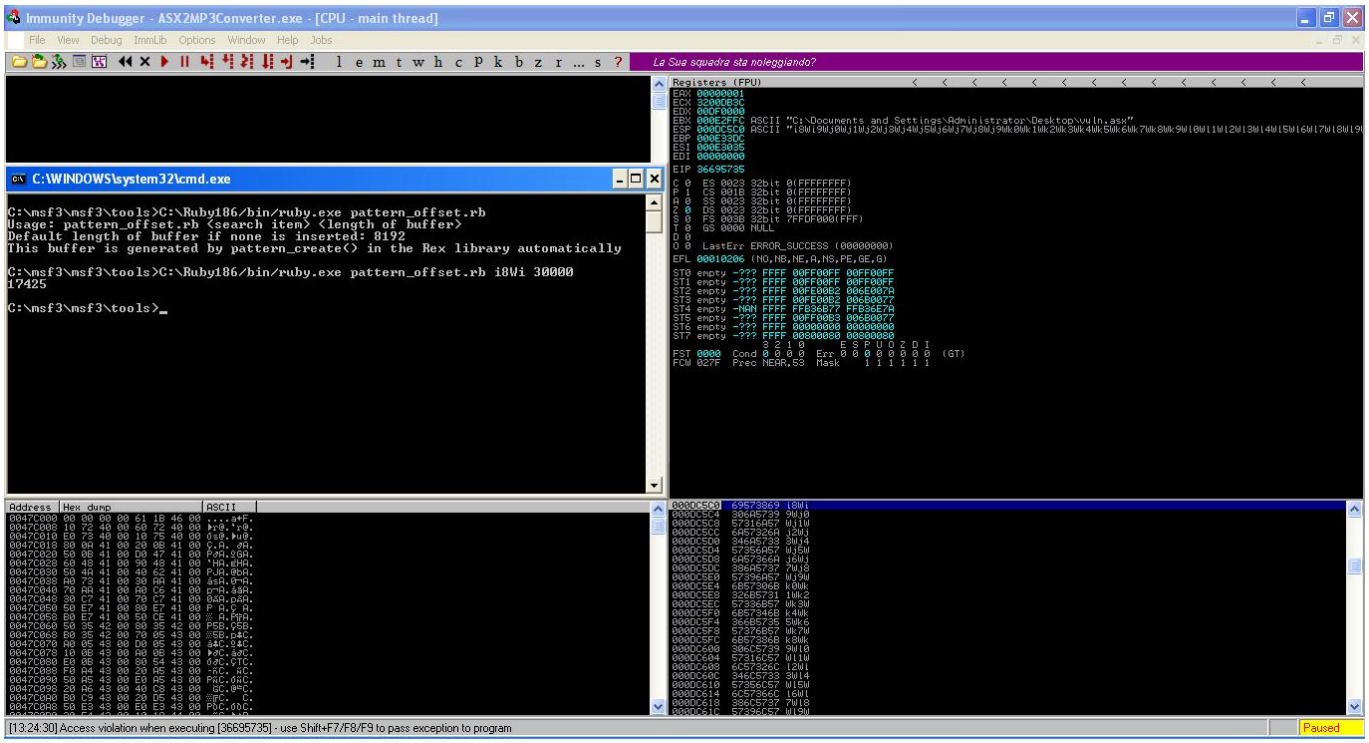



Fakat kaçıncı bayttan itibaren EIP kaydedicisine yazdığımızı deneme yanılma yolu ile tespit etmek zaman alacağı için hemen bu iş için tasarlanmış olan ve Metasploit aracı içinde yer alan pattern_create uygulamasına başvuruyoruz ve 30000 bayt büyüklüğünde bir dizi oluşturarak bu diziyi programımıza kopyalayarak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

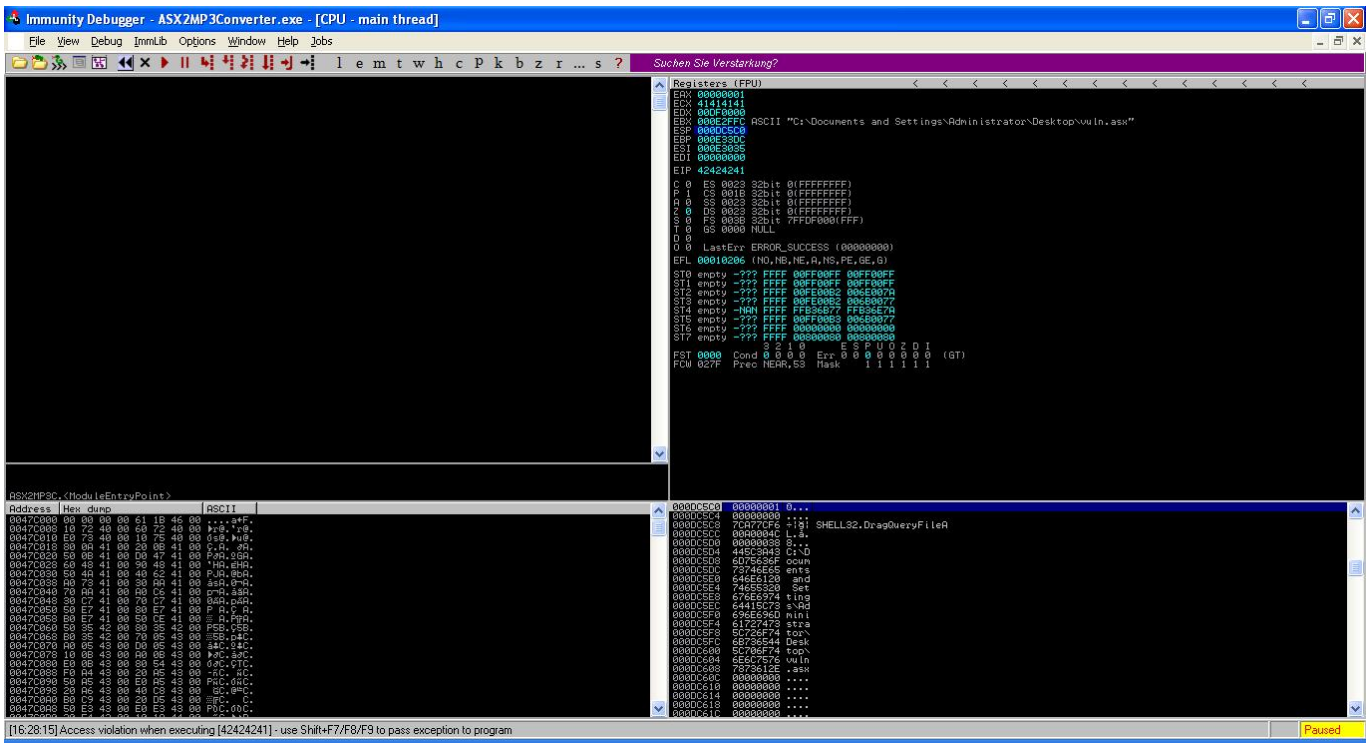

C:\WINDOWS\system32\cmd.exe
C:\msf3\msf3\tools>C:\Ruby186/bin/ruby.exe pattern_create.rb
Usage: pattern_create.rb length [set a] [set b] [set c]
C:\msf3\msf3\tools>C:\Ruby186/bin/ruby.exe pattern_create.rb 30000 > bof.txt
C:\msf3\msf3\tools>


bof.txt - Notepad
File Edit Format View Help
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac
i1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5
2Cq3Cq4Cq5Cq6Cq7Cq8Cq9Cr0Cr1Cr2Cr3Cr4Cr5Cr6Cr7Cr8Cr9Cs0Cs1Cs2Cs3Cs4Cs5Cs6C
Dy4Dy5Dy6Dy7Dy8Dy9Dz0Dz1Dz2Dz3Dz4Dz5Dz6Dz7Dz8Dz9Ea0Ea1Ea2Ea3Ea4Ea5Ea6Ea7Ea
g5Fg6Fg7Fg8Fg9Fh0Fh1Fh2Fh3Fh4Fh5Fh6Fh7Fh8Fh9Fi0Fi1Fi2Fi3Fi4Fi5Fi6Fi7Fi8Fi9
6Go7Go8Go9Gp0Gp1Gp2Gp3Gp4Gp5Gp6Gp7Gp8Gp9Gq0Gq1Gq2Gq3Gq4Gq5Gq6Gq7Gq8Gq9Gr0G
Hw8Hw9Hx0Hx1Hx2Hx3Hx4Hx5Hx6Hx7Hx8Hx9Hy0Hy1Hy2Hy3Hy4Hy5Hy6Hy7Hy8Hy9Hz0Hz1Hz
e9Jf0Jf1Jf2Jf3Jf4Jf5Jf6Jf7Jf8Jf9Jg0Jg1Jg2Jg3Jg4Jg5Jg6Jg7Jg8Jg9Jh0Jh1Jh2Jh3
0Kn1Kn2Kn3Kn4Kn5Kn6Kn7Kn8Kn9Ko0Ko1Ko2Ko3Ko4Ko5Ko6Ko7Ko8Ko9Kp0Kp1Kp2Kp3Kp4K
Lv2Lv3Lv4Lv5Lv6Lv7Lv8Lv9Lw0Lw1Lw2Lw3Lw4Lw5Lw6Lw7Lw8Lw9Lx0Lx1Lx2Lx3Lx4Lx5Lx
d3Nd4Nd5Nd6Nd7Nd8Nd9Ne0Ne1Ne2Ne3Ne4Ne5Ne6Ne7Ne8Ne9Nf0Nf1Nf2Nf3Nf4Nf5Nf6Nf7
4Ol5Ol6Ol7Ol8Ol9Om0Om1Om2Om3Om4Om5Om6Om7Om8Om9On0On1On2On3On4On5On6On7On8C
Pt6Pt7Pt8Pt9Pu0Pu1Pu2Pu3Pu4Pu5Pu6Pu7Pu8Pu9Pv0Pv1Pv2Pv3Pv4Pv5Pv6Pv7Pv8Pv9Pw
b7Rb8Rb9Rc0Rc1Rc2Rc3Rc4Rc5Rc6Rc7Rc8Rc9Rd0Rd1Rd2Rd3Rd4Rd5Rd6Rd7Rd8Rd9Re0Re1
8sj9sk0sk1sk2sk3sk4sk5sk6sk7sk8sk9sl0sl1sl2sl3sl4sl5sl6sl7sl8sl9sm0sm1sm2s
Ts0Ts1Ts2Ts3Ts4Ts5Ts6Ts7Ts8Ts9Tt0Tt1Tt2Tt3Tt4Tt5Tt6Tt7Tt8Tt9Tu0Tu1Tu2Tu3Tu
a1Va2Va3Va4Va5Va6Va7Va8Va9Vb0Vb1Vb2Vb3Vb4Vb5Vb6Vb7Vb8Vb9Vc0Vc1Vc2Vc3Vc4Vc5
2wi3wi4wi5wi6wi7wi8wi9wj0wj1wj2wj3wj4wj5wj6wj7wj8wj9wk0wk1wk2wk3wk4wk5wk6w
Xq4Xq5Xq6Xq7Xq8Xq9Xr0Xr1Xr2Xr3Xr4Xr5Xr6Xr7Xr8Xr9Xs0Xs1Xs2Xs3Xs4Xs5Xs6Xs7Xs
y5Yy6Yy7Yy8Yy9Yz0Yz1Yz2Yz3Yz4Yz5Yz6Yz7Yz8Yz9Za0Za1Za2Za3Za4Za5Za6Za7Za8Za9
7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1A
Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br
x0Cx1Cx2Cx3Cx4Cx5Cx6Cx7Cx8Cx9Cy0Cy1Cy2Cy3Cy4Cy5Cy6Cy7Cy8Cy9Cz0Cz1Cz2Cz3Cz4
1Ef2Ef3Ef4Ef5Ef6Ef7Ef8Ef9Eg0Eg1Eg2Eg3Eg4Eg5Eg6Eg7Eg8Eg9Eh0Eh1Eh2Eh3Eh4Eh5E
Fn3Fn4Fn5Fn6Fn7Fn8Fn9Fo0Fo1Fo2Fo3Fo4Fo5Fo6Fo7Fo8Fo9Fp0Fp1Fp2Fp3Fp4Fp5Fp6Fp
v4Gv5Gv6Gv7Gv8Gv9Gw0Gw1Gw2Gw3Gw4Gw5Gw6Gw7Gw8Gw9Gx0Gx1Gx2Gx3Gx4Gx5Gx6Gx7Gx8
5Id6Id7Id8Id9Ie0Ie1Ie2Ie3Ie4Ie5Ie6Ie7Ie8Ie9If0If1If2If3If4If5If6If7If8If9I
Jl7Jl8Jl9Jm0Jm1Jm2Jm3Jm4Jm5Jm6Jm7Jm8Jm9Jn0Jn1Jn2Jn3Jn4Jn5Jn6Jn7Jn8Jn9Jo0Jc
t8Kt9Ku0Ku1Ku2Ku3Ku4Ku5Ku6Ku7Ku8Ku9Kv0Kv1Kv2Kv3Kv4Kv5Kv6Kv7Kv8Kv9Kw0Kw1Kw2
9Mc0Mc1Mc2Mc3Mc4Mc5Mc6Mc7Mc8Mc9Md0Md1Md2Md3Md4Md5Md6Md7Md8Md9Me0Me1Me2Me3M

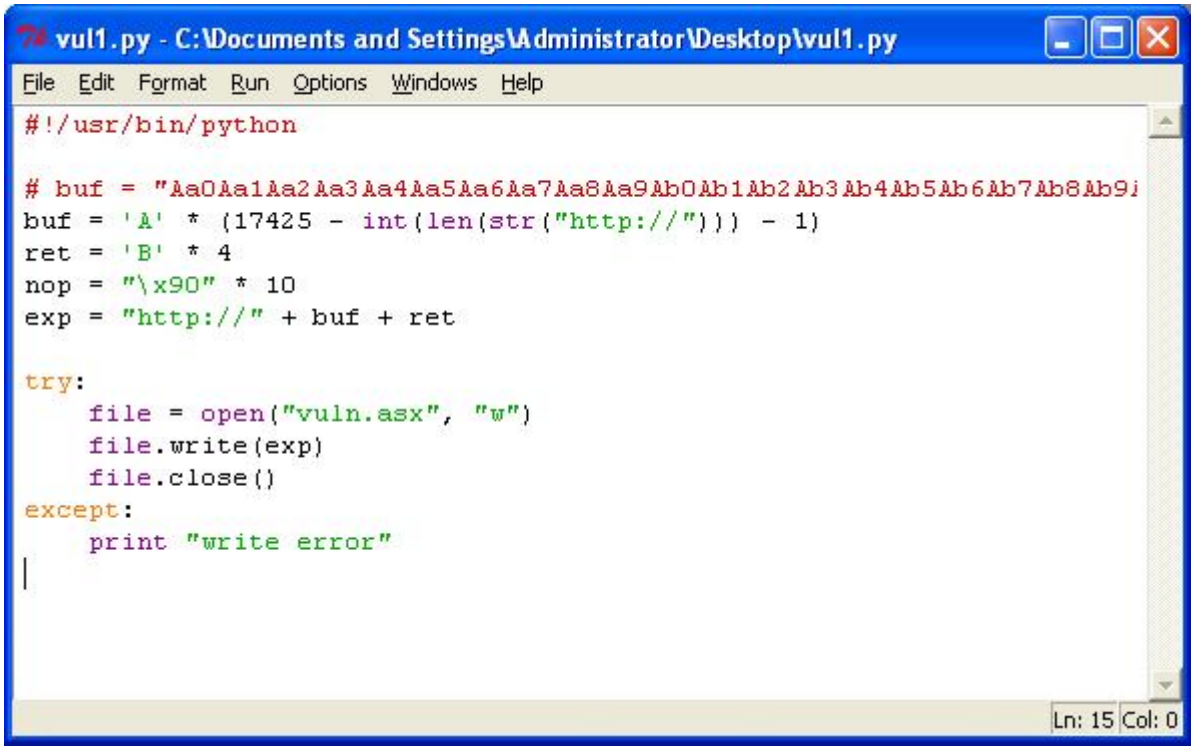


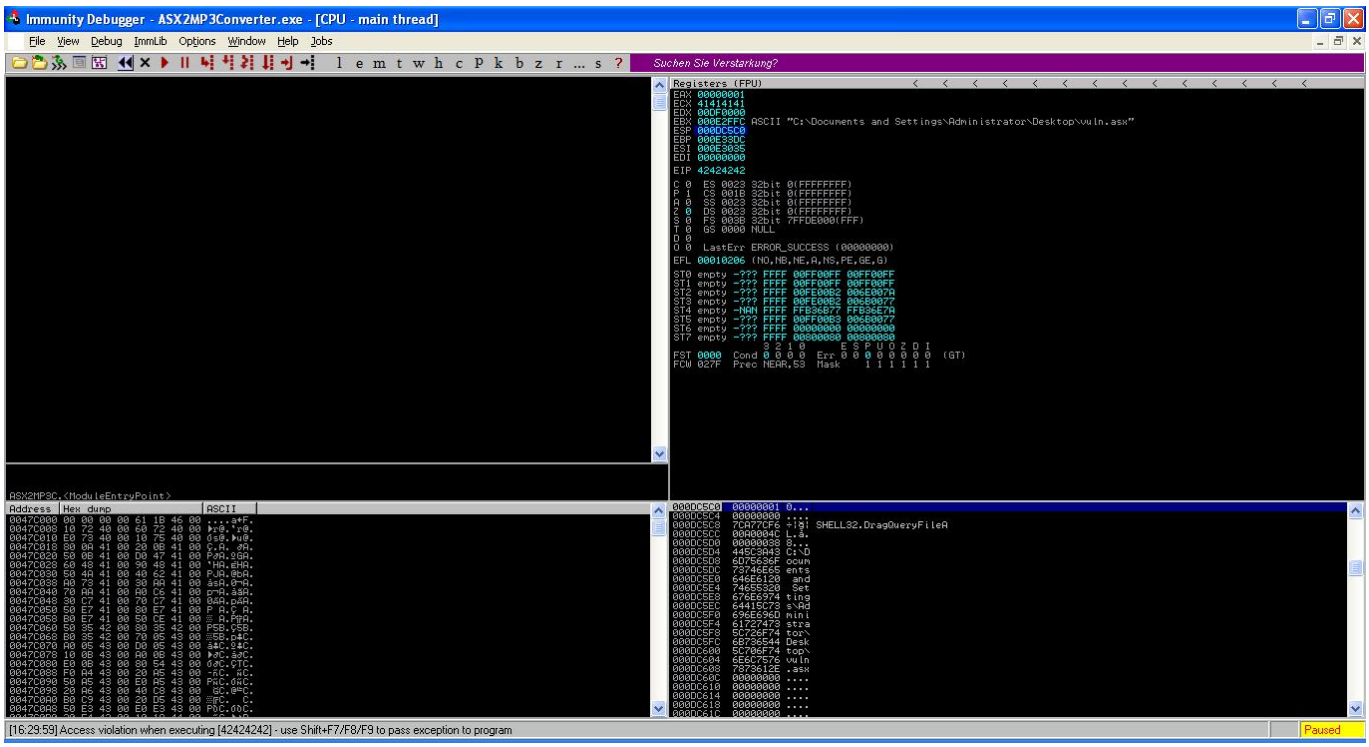




Programımızı aşağıdaki şekilde güncelledikten sonra çalıştırdığımızda EIP kaydedicisini tekrar kontrol ediyoruz.

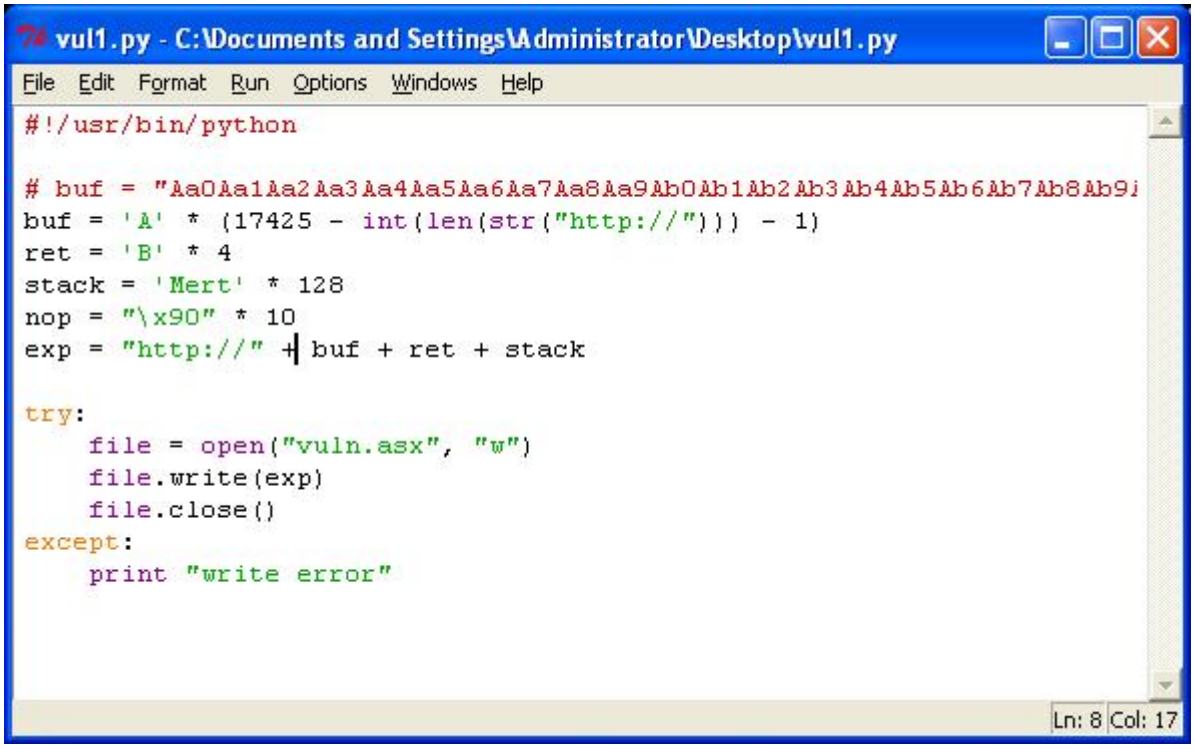



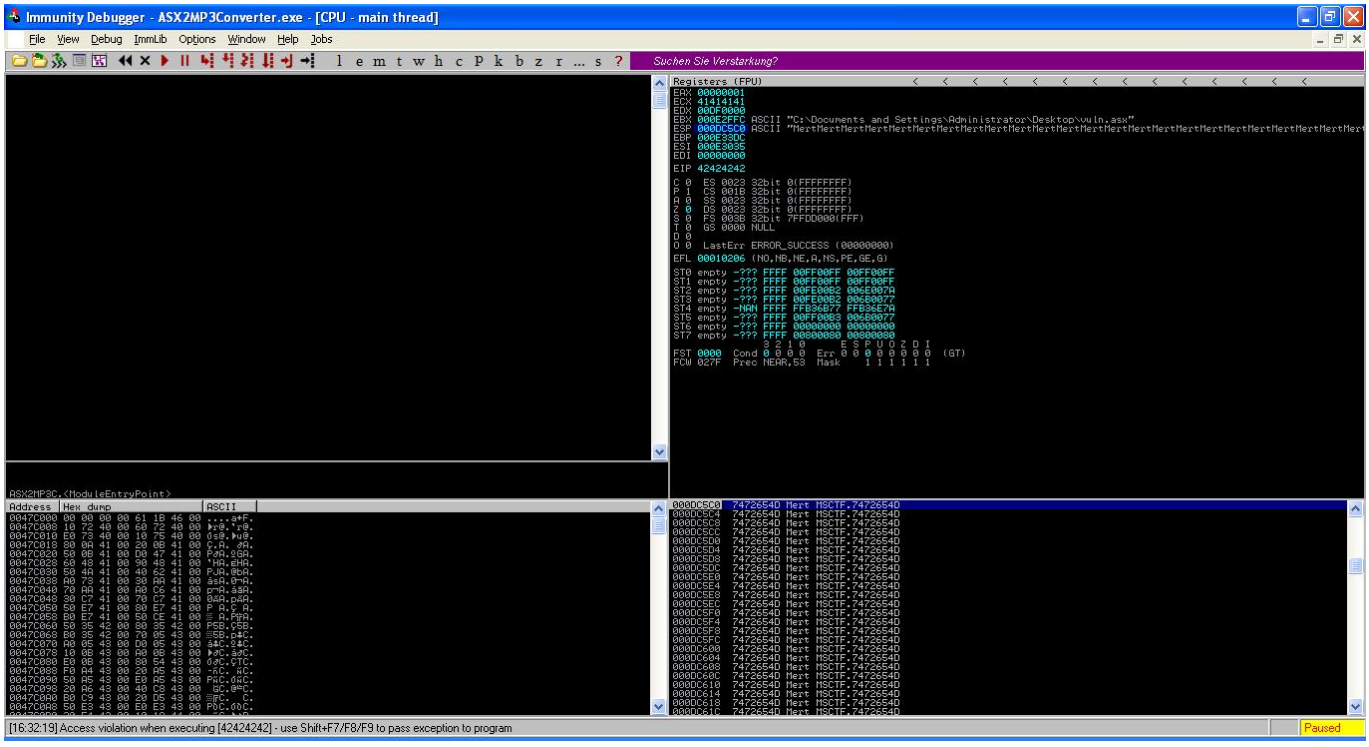

Fakat görünen o ki EIP kaydedicisine fazladan 1 bayt A gelmiş ve EIP kaydedicisi 42424242 (BBBB) yerine 42424241 (BBBA) değerine sahip olmuş. Programımızda son bir değişiklik yaptıktan sonra asx uzantılı dosyayı tekrar oluşturup programa yüklediğimizde bu defa başarıyla EIP kaydedicisi üzerine istediğimiz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



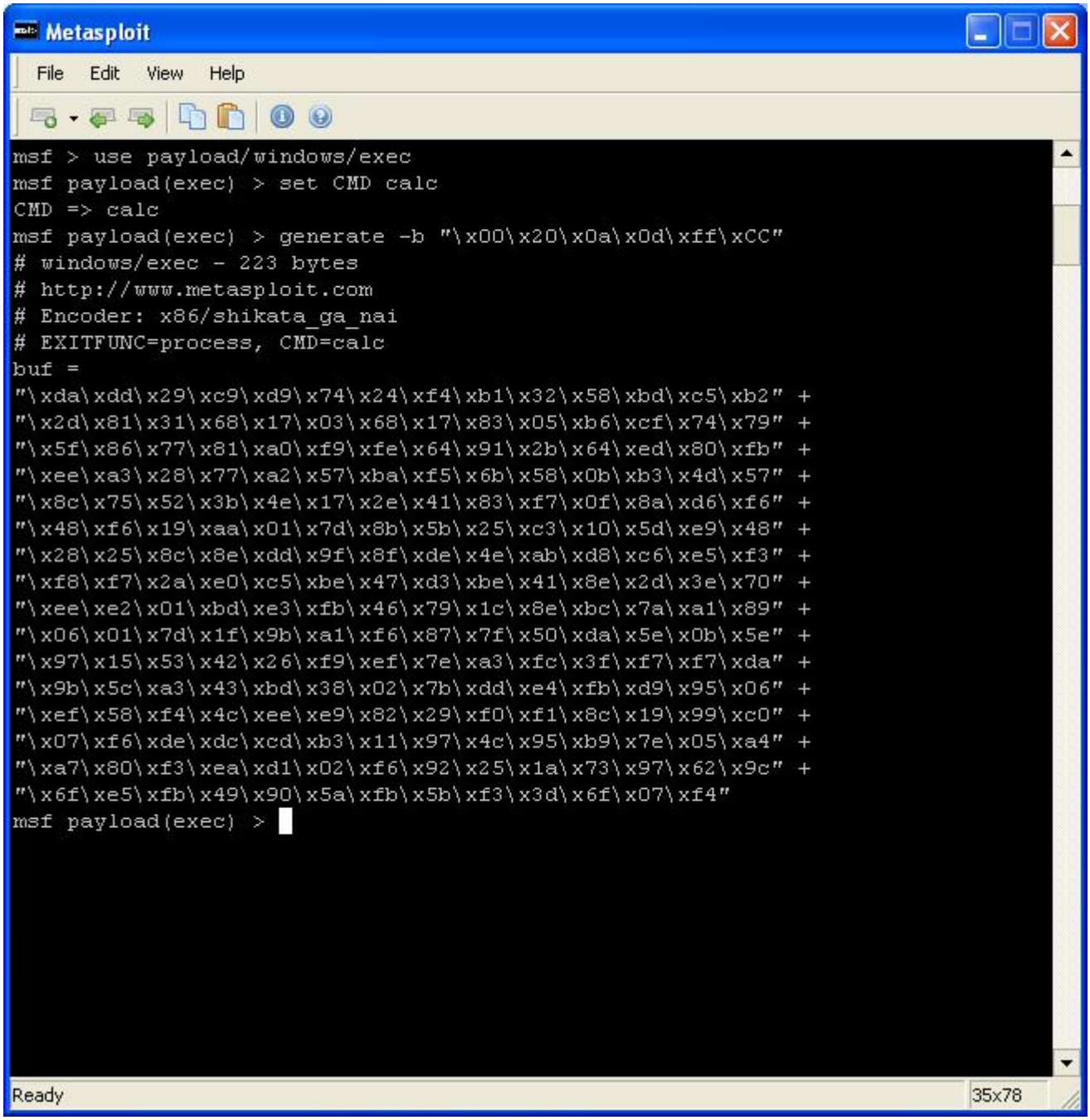

Arabellek taşması ile yığına yeterli miktarda veri yazıp yazamadığımızı kontrol etmek için (shellcode yığında yer alacağı için 400 bayt yeterli olacaktır) programımıza aşağıdaki gibi bir stack değişkeni ekliyor, tekrar çalıştırıyor ve yığına yazabildiğimizi görüyoruz.

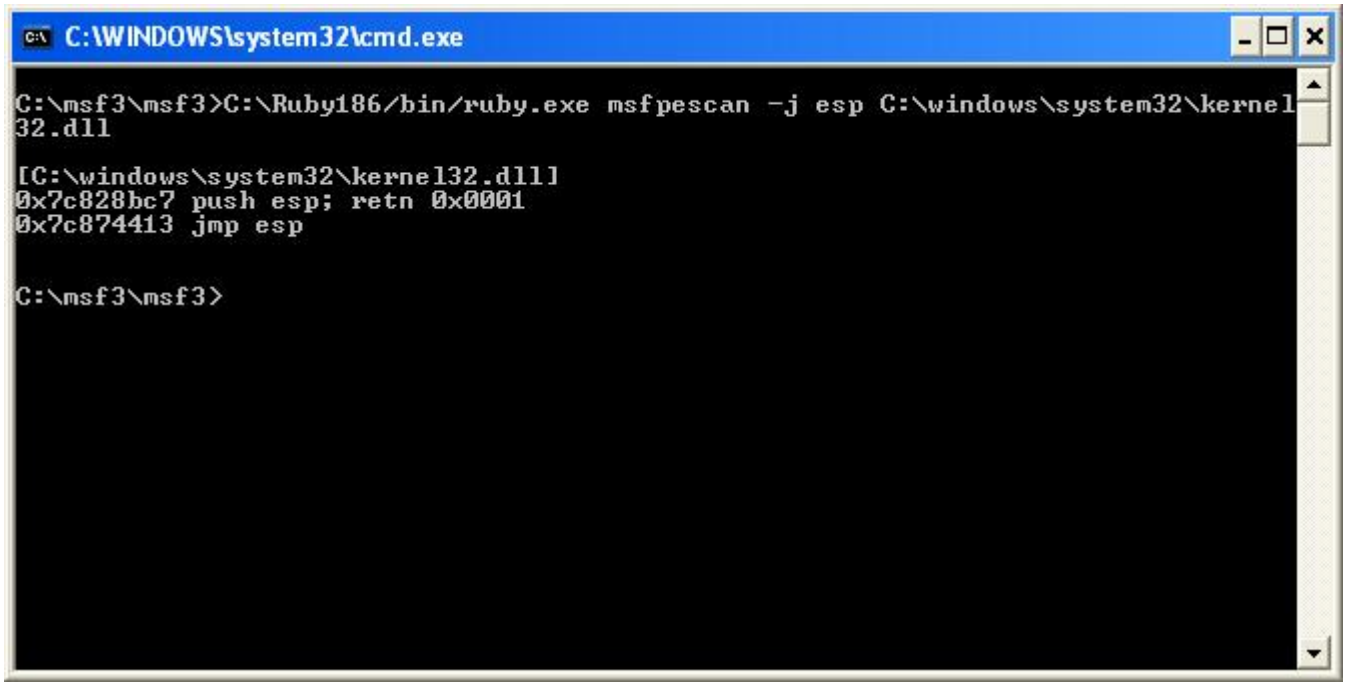


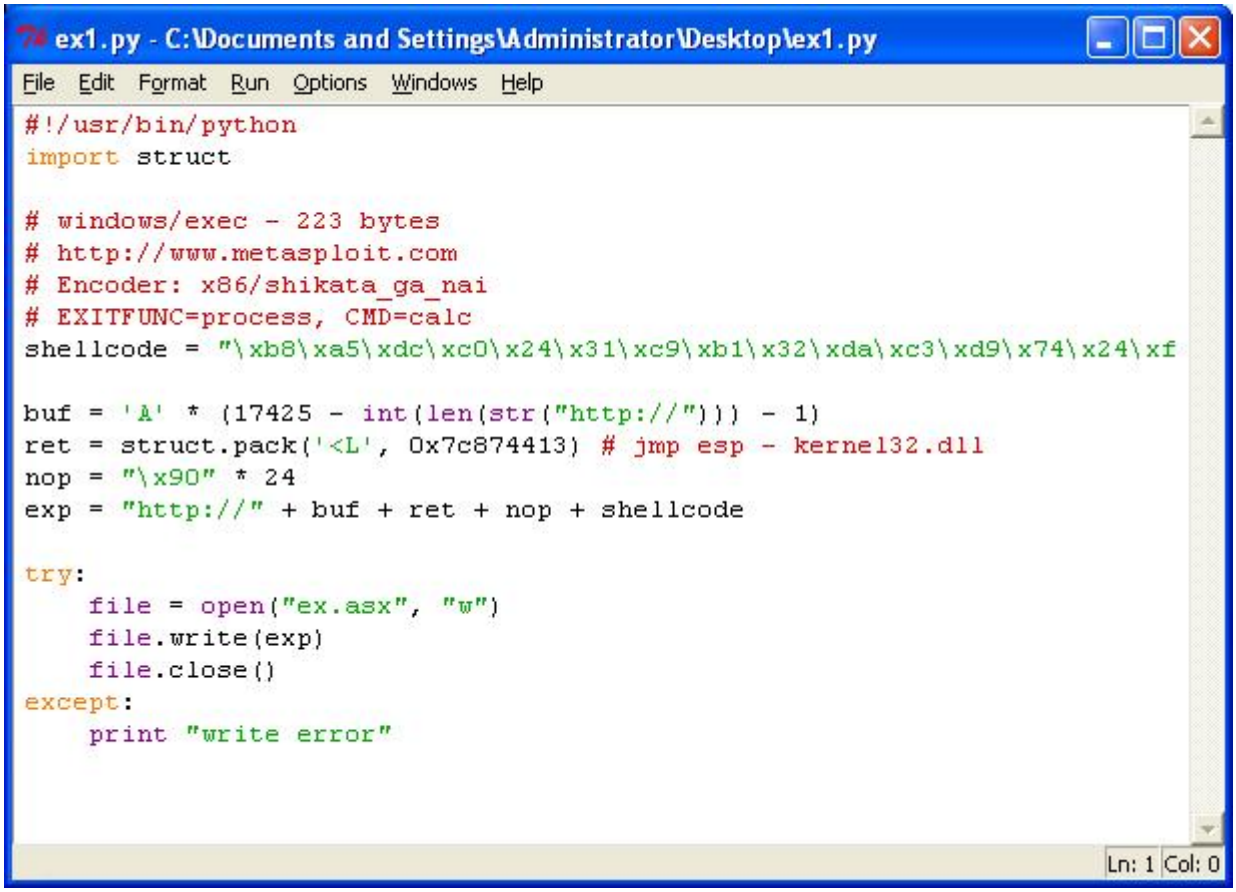




Amacımız hazırladığımız kodu (shellcode) yığına yazmak ve depolanmış, saklanmış EIP kaydedicisini yığına yönlendirmek (JMP ESP) olduğu için öncelikle Metasploit'in son sürümünün yüklü olduğu bilgisayarımızda (şuan için 3.5.0) hesap makinası uygulamasını çalıştıran kodu (shellcode) oluşturuyoruz. Ardından yığına yönlendirmek için kullanacağımız assembly komutunu yine diğer bir Metasploit uygulaması olan msfpescan ile örnek olarak kernel32.dll dosyası üzerinde aratıyoruz. Son olarak istismar aracımızı elde ettiğimiz bu bilgiler ile güncelledikten ve çalıştırdıktan sonra oluşturduğumuz asx dosyasını ASX to MP3 Convertor uygulamasına

yüklettiğimizde hesap makinası çalışıyor ve mutlu sona ulaşmış oluyoruz.






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

çoğu bir şekilde atlatılabildiği için istismarı zorlaştırmaktan öteye gidememişlerdir.

Bir sonraki yazıda görüşmek dileğiyle herkese güvenli günler dilerim…