SEH İstismarı

written by Mert SARICA | 14 December 2010

Rahmetli milw0rm ve veliahtı olan Exploit-DB sitelerine bakacak olursanız çoğu istismar aracının SEH (structured exception handler)'i yani türkçe meali ile yapılandırılmış özel durum işlemesini istismar ettiğini görebilirsiniz. Sayının fazla olmasının nedeni olarak tespit edilmesinin ve istismar edilmesinin kolay olduğunu söyleyebilirim. Modern windows işletim sistemlerinde (Vista ve sonrası) yer alan istismar önleyici korumalar (SEHOP, ASLR vs.) SEH istismarını zorlaştırmaktadır. Windows 7 kullanıyorum o halde rahatım dememeniz için ufak bir ekleme yapayım, (default) varsayılan olarak kurulan bir Windows 7 işletim sisteminde DEP özelliği Windows XP işletim sisteminde olduğu gibi sadece windows'un kendi programlarını ve servislerini korumakta, SEH istismarını zorlaştıran SEHOP özelliği ise devre dışı olarak gelmektedir bu nedenle modern windows işletim sistemi kullanıyorsanız sıkılaştırmanız yararınıza olacaktır.

Programlama ile içli dışlı olanlar bilirler, kimi programlama dilinde (C ne yazıkki bunlardan bir tanesi değil) try & catch, try & except gibi hata yakalamak amacıyla kullanılan özel durum işlemeleri (bloklar) bulunmaktadır. Bu blokların amacı içlerinde gerçekleşen işlemlerde bir hatanın ortaya çıkması durumunda kullanıcıyı uyarmak ve işlemin devam etmesini durdurmaktır aksi durumda bu hata, sistem üzerinde istenmeyen sonuçlara yol açabilmektedir.

Geliştirilen bir programda, hata yakalamak için kullanılan bu bloklara yer verilmemesi veya bu blokların oluşan hatayı yakalayamaması durumunda işletim sisteminin hata yakalama bloğu olan Windows SEH (işletim sistemi seviyesi) duruma müdahale ederek hatayı yakalamaktadır.

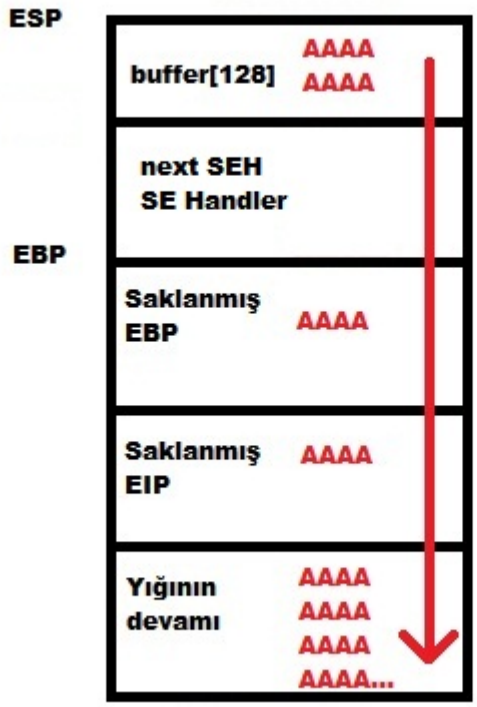
Bir programın hatayı yakalayabilmesi için her bir hata yakalama bloğunu işaret eden işaretci/göstergeç (pointer), yığında (stack) saklanmaktadır. Bir programda yer alan tüm hata yakalama blokları birbirlerine zincirdeki halkalar (SEH chain) gibi bağlıdırlar ve zincirin son halkasında Windows SEH yer alır.

SEH, bir sonraki hata yakalama bloğu işaretçisi (next seh) ve asıl hata yakalama bloğu işaretçisi (seh) olmak üzere 8 bayttan oluşmaktadır.

SafeSEH desteği ile geliştirilmiş bir program, Windows'daki özel durum işleme

mekanizmaları üzerinde ek denetimler gerçekleştirerek istismarı zorlaştırır. Yazımın ilerleyen kısmı, SafeSEH koruması devrede olmayan programlar, modüller ve DLL dosyaları için yapılandırılmış özel durum işlemesinin nasıl kötüye kullanılabilmesi üzerinedir.

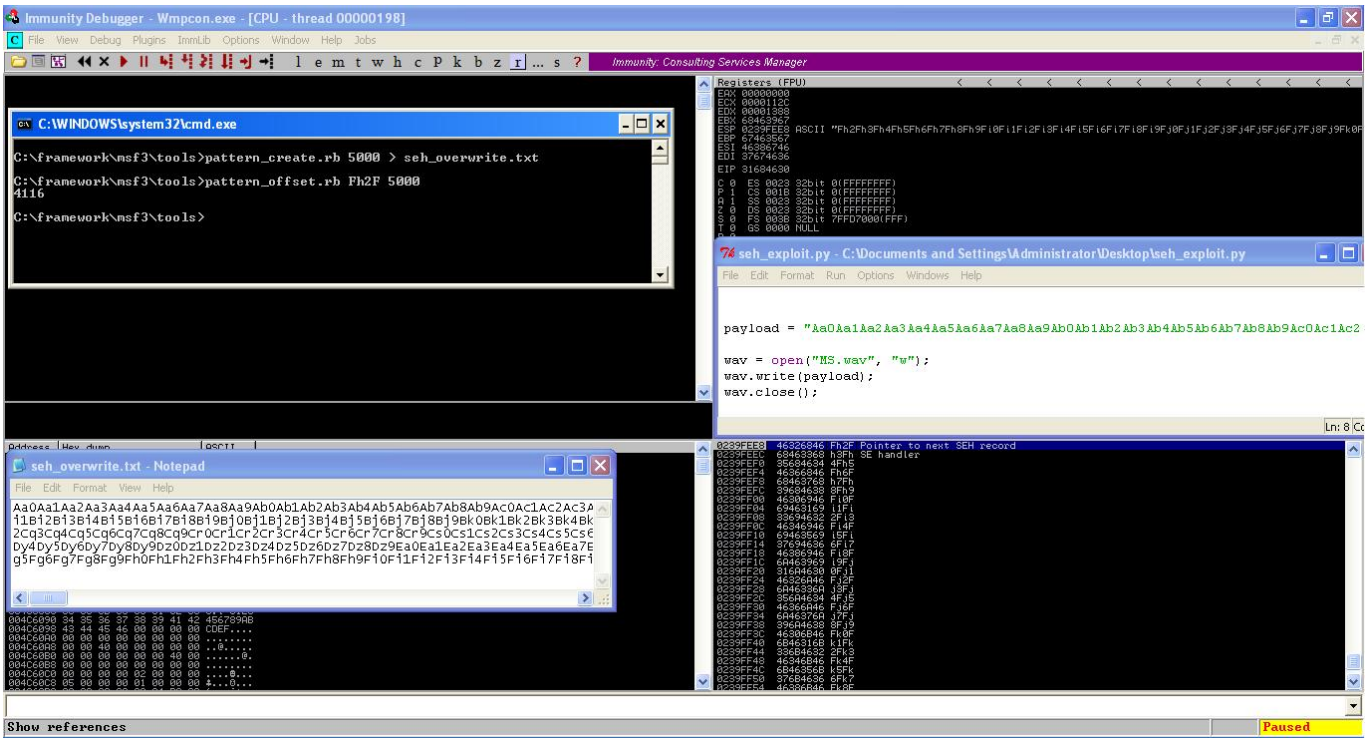
SEH istismarı kısaca ve kabaca arabellek taşmasında olduğu gibi dinamik bir değişkene kapasitesinden daha fazla veri kopyalanması ile SEH'in içinde yer alan işaretçilerin üzerine istenilen adreslerin yazılmasına ve programın akışının değiştirilmesine denir.



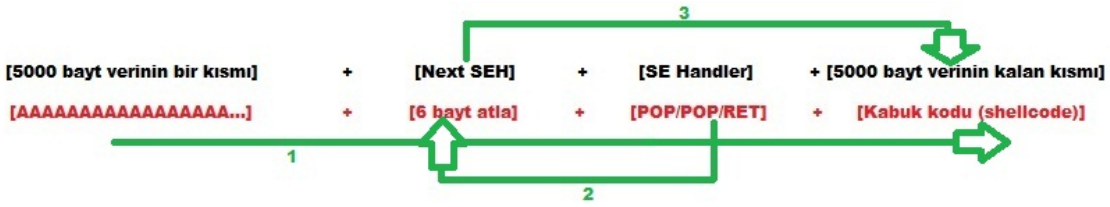
Daha net anlaşılabilmesi adına ufak bir örnek üzerinden gidecek olursak SEH istismarına imkan tanıyan ve arabellek taşması zafiyetine sahip olan Free WMA MP3 Converter v1.1 aracını inceleyelim.

Free WMA MP3 Converter, WMA, WAV ve MP3 uzantılı dosyaları birbirilerine çevirmeye yaran basit bir programdır. Programdaki zafiyetin varlığını teyit etme adına öncelikle bir .wav uzantılı bir dosya oluşturmamız gerekmektedir. Bunun için bir önceki yazımda da kısaca bahsetmiş olduğum Metasploit'in pattern_create aracından faydalanabiliriz. Bu araç ile oluşturduğumuz 5000 karakterden oluşan diziyi WAV uzantılı dosyaya kopyalayalım. Programı Immunity Debugger ile çalıştırdıktan sonra "WAV to MP3" menüsüne tıkladığımızda bizden herhangi bir WAV dosyasını girdi olarak vermemizi istemektedir. Bunun içinde bir adım evvel yaratmış olduğumuz WAV dosyasını kullandığımızda next SEH ve SE Handler'ın üzerine başarıyla yazabildiğimizi

görebiliriz. Metasploit'in pattern_offset aracı ile kaçıncı baytın Next SEH'e denk geldiğine baktığımızda ise 4116. bayt olduğunu görebiliriz. Ufak bir hesaplamadan sonra (FFFC – FEF0 = 268) SE Handler'dan sonraki 268 baytın üzerine başarıyla istediğimiz veriyi yazabildiğimizi görebiliyoruz.



Kısaca ortaya çıkan durumu ve hemen altında istismar aracımızı ne şekilde oluşturmamız gerektiğine bakacak olursak;

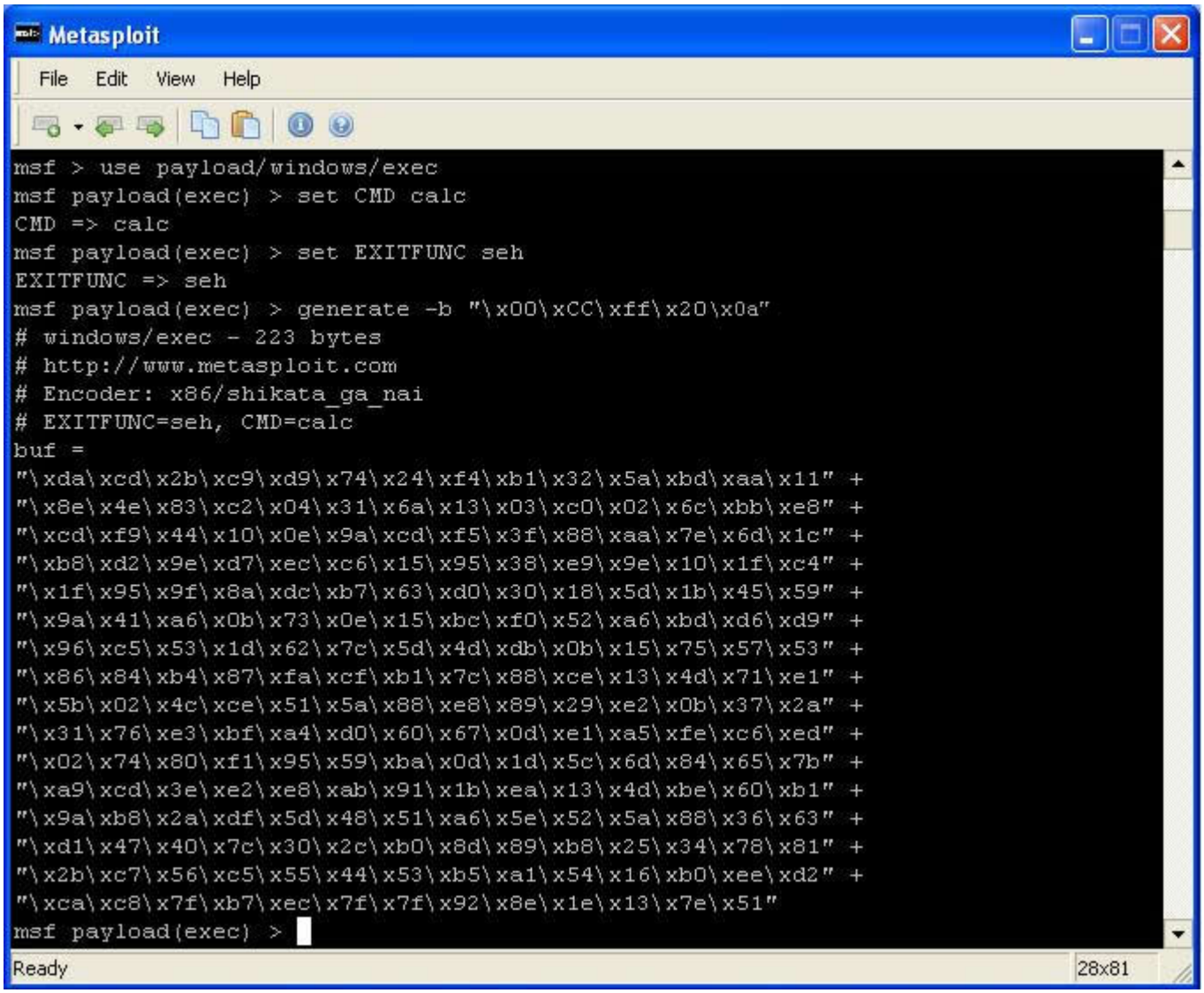


SE Handler'ın üzerine POP POP RET komutlarını kopyalamamızın amacı programda hataya (özel duruma) yol açmak ve bu sayede programın akışının Next SEH'e yönlendirilmesini sağlamaktır. Next SEH'te yer alan "6 bayt atla" komutu ile programın akışı SE Handler üzerinden 6 bayt atlayarak sistem üzerinde dilediğimiz işlemi gerçekleştirmemize imkan tanıyan kod parçasına yani kabuk koduna (shellcode) gidecek şekilde devam edecektir.

SE Handler'dan sonraki 268 bayta dilediğimizi veriyi yazabildiğimiz için kabuk kodumuzu buraya koymamız yeterli olacaktır.

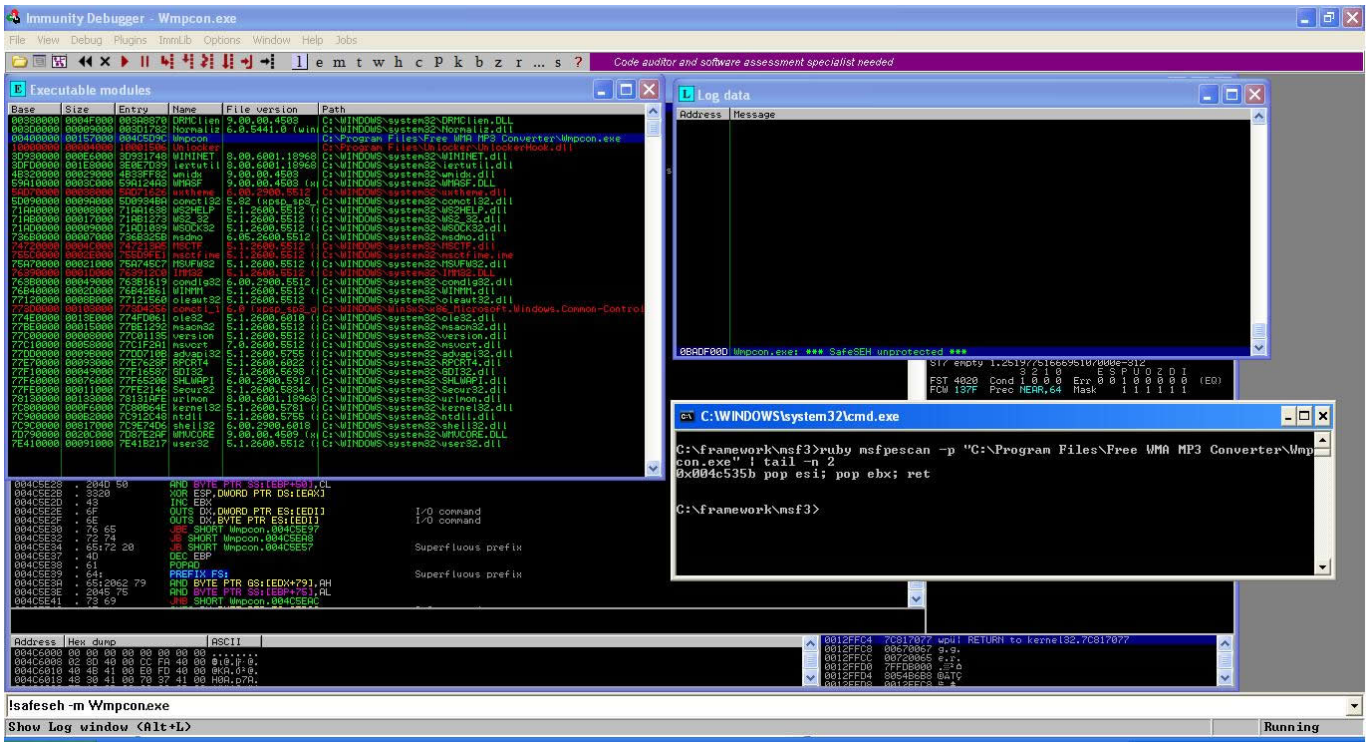
Kimi zaman SE Handler'dan sonraki alan kabuk kodumuz için yeterli olmayabilir bu durumda da kabuk kodu için en ideal yer yukarıdaki resimde yer alan 5000 baytlık ilk kısım olacaktır. SE Handler sonrasında yer alan adrese, geri X bayt zıpla komutu (JMP backward) vererek akışın kabuk kodumuza ilerlemesini sağlayabiliriz.

Öncelikle kabuk kodunu oluşturmamız gerekiyor bunun için Metasploit aracı ile calc.exe programını çalıştıran bir kabuk kodu oluşturalım.

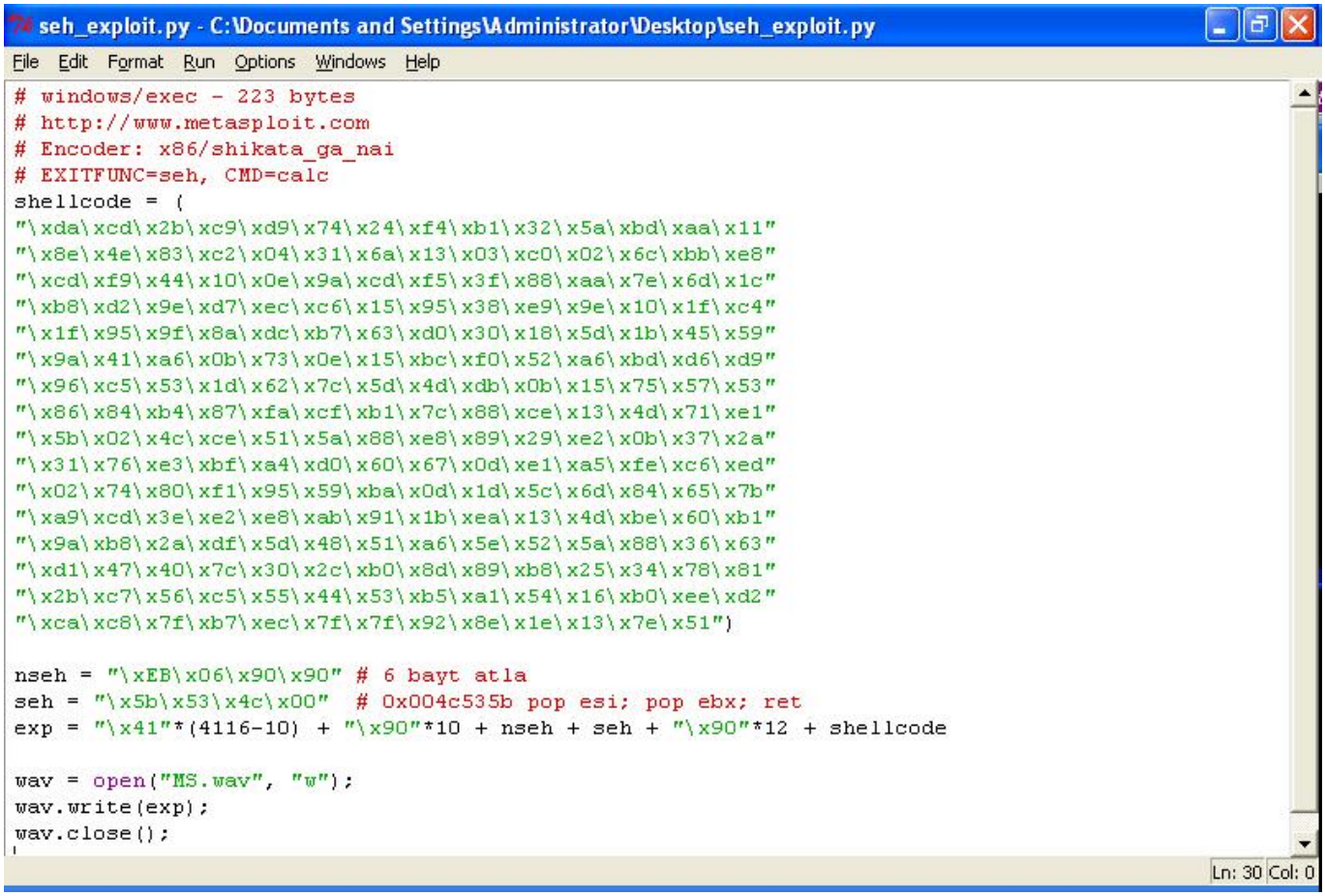


POP/POP/RET komutunu SafeSEH özelliğinin devre dışı olduğu ve program tarafından kullanılan herhangi bir program, modül veya DLL dosyasında aramamız gerektiği için öncellikle bunu bulmakla işe koyuluyoruz. Bunun için Immunity Debugger aracında yer alan SafeSeh scriptini kullanabiliriz. !safeseh scripti -m parametresi ile çalışıyor bu yüzden hemen Immunity Debugger'da yer alan "E" butonuna basarak "executable modules" penceresine hızlıca göz atıyor ve şansımızı ilk olarak programın kendisinden yana (Wmpcon.exe) kullandığımızda programın SafeSEH'i desteklemediğini öğreniyoruz

ve seviniyoruz :) Bir sonraki adımda, bu modülde/programda yer alan POP/POP/RET adresini aramamız gerekiyor. Bunun için Metasploit aracında yer alan msfpescan aracından faydalanabiliriz. Aracı aşağıdaki ekran görüntüsünde yer aldığı şekilde çalıştırdığımızda hemen istediğimiz adresi buluyoruz ve istismarı gerçekleştirmek için ihtiyaç duyduğumuz tüm bilgileri elde etmiş oluyoruz.



Son olarak istismar aracımızı elde ettiğimiz bu bilgiler ışığında güncelleyip çalıştırdığımızda SEH'i başarıyla istismar ederek Windows'un hesap makinası aracının (calc.exe) çalıştığını görebiliyoruz.



Bu defa video çekmemi isteyenlerin sesine kulak verdim ve konu ile ilgili ufak bir video çektim.

Bir sonraki yazıda görüşmek dileğiyle herkese güvenli haftalar diliyorum.