

UPX Packing

Bu yazıda C dilinde hazırlanan basit uygulamaların yaygın kullanılan paketleme yazılımlarından biri olan UPX paketleyiciyle paketlenmelerinde nasıl görüldükleri incelenmeye çalışılacaktır. Analiz süresince Assembly dili üzerinde çalışmalar yapılacağı için Assembly diline hakim değilseniz bu yazıdan önce **“Assembly”** dizini altındaki yazılarıma göz atmanızı tavsiye ederim.

Hazırlanan kaynak kod derlendikten sonra UPX paketleme aracıyla paketleyebilmek için kullanılan komut satırında **“upx <Command> <Options> <Input File Name> <Output File Name>”** formatında paketlenen veya paket dışına çıkarılacak dosyanın belirtilmesi gerekiyor (UPX paketleme aracına kendi sitesinden kullandığınız işletim sistemine uygun halini indirebilirsiniz). Bunun için UPX dosyasının bulunduğu dizine giriş yapılarak **“upx -9 -o <Output File Name And Directory> <Output File Name And Directory>”** komutu kullanılarak dosyalar UPX aracıyla paketlenmiştir.

```
C:\Users\VMUser\Downloads\upx-4.2.2-win64>upx.exe -9 -o .\..\..\Desktop\Packed_Basic-Structures.exe ".\..\..\Desktop\Basic-Structures.exe"

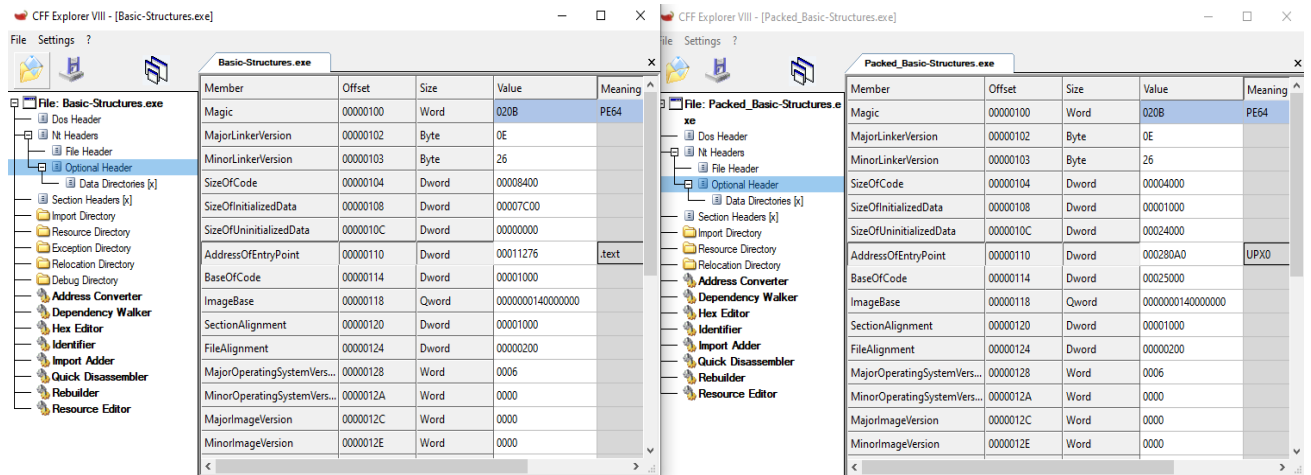
Ultimate Packer for eXecutables
Copyright (C) 1996 - 2024
UPX 4.2.2      Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 3rd 2024

-----
File size      Ratio      Format      Name
-----
64512 ->      15360      23.81%      win64/pe      Packed_Basic-structures.exe

Packed 1 file.
```

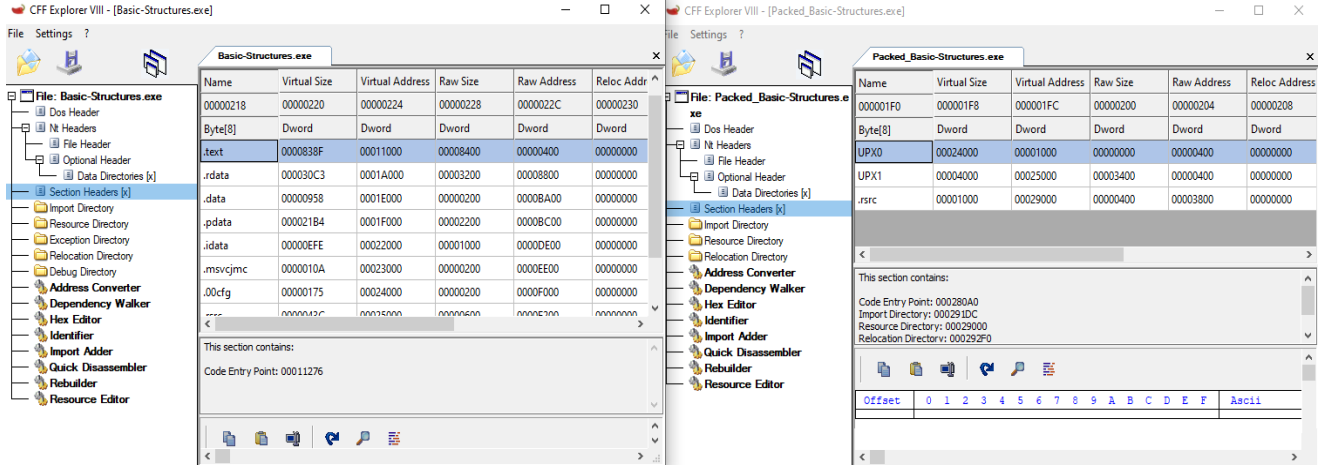
İkili dosyaları öncelikle CFF Explorer aracı üzerinde incelediğimizde;

- İlk göze çarpan kısmı “Optional Header” kısmındaki program kodlarının başlangıç adresini gösteren “AddressOfEntryPoint” değeri kodun bulunduğu “.text” sekmesini göstermesi gerekirken farklı bir alandan çalışmaya başlayacağını gösteriyor.

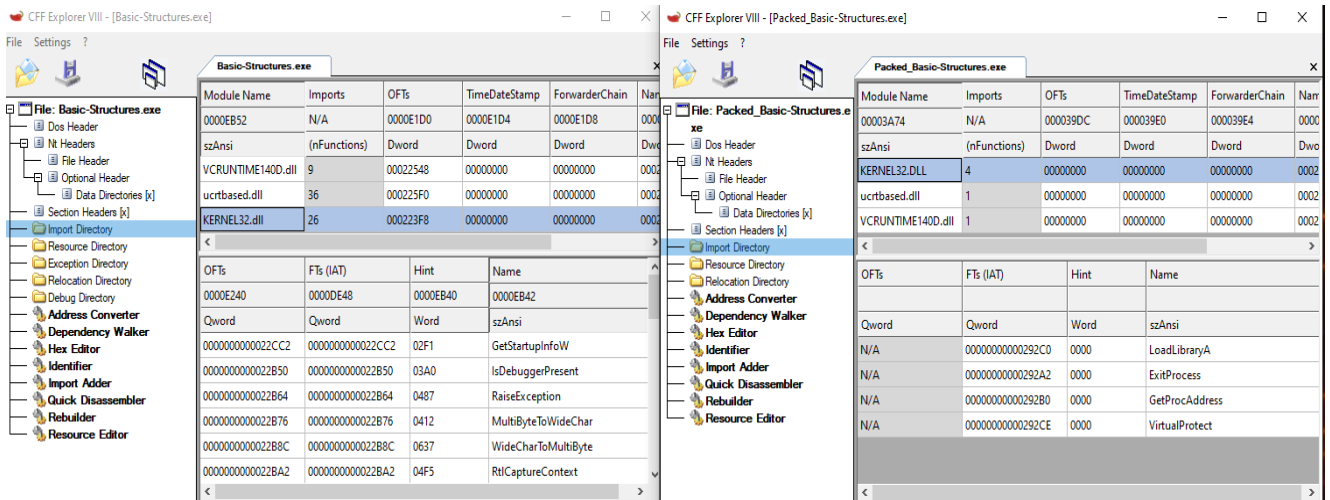


- Göze çarpan ikinci bir kısım ise “Section Header” alanındaki başlık bilgileri oldu. Burada Paketlenmemiş dosyada normal bir ikili dosyada olması gereken “.text”, “.data” gibi sekmeler bulunurken paketlenmiş ikili dosyada “UPX0”, “UPX1” gibi normalin dışında sekmeler görünür.

- Burada dikkat edilmesi gereken ikinci bir detay ise **Raw_Size** (Section'ın disk üzerindeki boyutunu gösterir) sütunundaki değerler olmalıdır. Paketlenmemiş ikili dosyada **.text** kısmında bu değer sıfırdan farklıyken paketlenmiş ikili dosyada **UPX0** (.test/.code bölümü) kısmı için bu değer sıfırdır. Bu durumdan **Raw_Size** ile **Virtual_Size** arasındaki değer farkının yüksek olması ikili dosyanın paketlenmiş olma ihtimaline bir işaret olabileceğini gösterir.
 - UPX0 kısmı orijinal kodu yüklemek için ayrılmış alandır. UPX1 kısmında ise Stub ve orijinal kodun sıkıştırılmış hali bulunur. UPX0 alanının Raw_Size değerinin sıfır görünme nedeni, program çalıştırıldığında sıkıştırılmış haldeki kod parçasının orijinal hale getirilerek buraya yüklenecek olmasıdır.



- İkili dosyanın paketlenmiş olabileceğini gösteren ikinci bir işaret ise program çalıştırıldığında içe aktarılan DLL dosyalarının ve bu DLL dosyalarının içerisinde kullanılan fonksiyonların miktarı olacaktır. Paketlenmemiş ikili dosyada bu değerler çok daha yüksek olurken paketlenmiş ikili dosyada çok daha az miktardadır.
 - Burada dikkat edilmesi gereken ikinci bir ipucu ise paketlenmiş ikili dosyada **KERNEL32.DLL** kütüphanesi içinde **“LoadLibraryA”** ve **“GetProcAddress”** fonksiyonlarının kullanılmasıdır.



İkili dosyaları PE Studio aracı üzerinde incelediğimizde;

- CFF Explorer aracından farklı olarak Section kısımlarının yetkileri de dikkat çekiyor. Paketlenmemiş ikili dosyada olması gerektiği gibi sadece **.text** alanının çalıştırma yetkisi, **.data** alanının yazma yetkisi bulunurken, paketlenmiş ikili dosyada UPX0 alanının da UPX1 alanının da yama ve çalıştırma yetkileri bulunuyor.
 - o CFF Explorer aracından görüldüğü gibi UPX0 alanının bellekteki boyutu 0 bayt görünüyor.
 - o Alanların içindeki karakterler arası ilişki/anlamlılık yüzdesi olarak ifade edilebilecek Entropy değerlerindeki fark da göze çarpıyor. Paketlenmiş ikili dosyada paketlenmemiş ikili dosyaya kıyasla içerisinde bulundurduğu karakterler arası daha az ilişkiye/anlamlı yapılara sahip olduğu görülebilir. Bunun nedeni paketlenmiş ikili dosyada orijinal kod parçasının sıkıştırılmış durumda saklanmasıdır.

pestudio 9.58 - Malware Initial Assessment - www.winitor.com (read-only)

file settings about

c:\users\vmuser\desktop\malware analysis\bi

indicators (sections > writable)

footprints (count > 17)

virustotal (sample > unknown)

dos-header (size > 64 bytes)

dos-stub (size > 168 bytes)

rich-header (tooling > Visual Studio 2015)

file-header (executable > 64-bit)

optional-header (subsystem > console)

directories (count > 7)

sections (characteristics > self-modifying)

libraries (count > 3)

imports (flag > 71)

exports (n/a)

thread-local-storage (n/a)

.NET (n/a)

resources (signature > manifest)

strings (count > 813)

debug (streams > 2)

property	value	value	value	value
section	section[0]	section[1]	section[2]	section[3]
name	.textbss	.text	.data	.data
footprint > sha256	n/a	FD9A8AEB6D59BD8FB65CD...	48DFF3919C023D96D9BC598...	70F3A9DF
entropy	n/a	3.631	2.258	0.548
file-ratio (98.41%)	n/a	52.38 %	19.84 %	0.79 %
raw-address (begin)	0x00000000	0x00000400	0x00008800	0x0000BA00
raw-address (end)	0x00000000	0x00008800	0x0000BA00	0x0000BC00
raw-size (63488 bytes)	0x00000000 (0 bytes)	0x00008400 (33792 bytes)	0x00003200 (12800 bytes)	0x00000200 (512 bytes)
virtual-address	0x00001000	0x00011000	0x0001A000	0x0001E000
virtual-size (128928 bytes)	0x00010000 (65536 bytes)	0x0000838F (33679 bytes)	0x000030C3 (12483 bytes)	0x000009F0 (1600 bytes)
characteristics	0xE00000A0	0x60000020	0x40000040	0xC0000000
write	x	-	-	x
execute	x	x	-	-
share	-	-	-	-
self-modifying	x	-	-	-
virtual	x	-	-	-

Unpack

pestudio 9.58 - Malware Initial Assessment - www.winitor.com (read-only)

file settings about

c:\users\vmuser\desktop\malware analysis\bi

indicators (sections > writable)

footprints (count > 8)

virustotal (sample > unknown)

dos-header (size > 64 bytes)

dos-stub (size > 168 bytes)

rich-header (tooling > Visual Studio 2015)

file-header (executable > 64-bit)

optional-header (subsystem > console)

directories (count > 5)

sections (characteristics > self-modifying)

libraries (count > 3)

imports (flag > 6)

exports (n/a)

thread-local-storage (n/a)

.NET (n/a)

resources (signature > manifest)

strings (count > 526)

debug (n/a)

manifest (level > aslInvoker)

version (n/a)

notification (n/a)

property	value	value	value	value
section	section[0]	section[1]	section[2]	section[3]
name	UPX0	UPX1	.rsrc	
footprint > sha256	n/a	2D833E407682D42668BDC72...	C26361172E4C5F7C4F3E20D...	
entropy	n/a	7.789	4.021	
file-ratio (93.33%)	n/a	86.67 %	6.67 %	
raw-address (begin)	0x00000400	0x00000400	0x00003800	
raw-address (end)	0x00000400	0x00003800	0x00003C00	
raw-size (14336 bytes)	0x00000000 (0 bytes)	0x00003400 (13312 bytes)	0x00000400 (1024 bytes)	
virtual-address	0x00001000	0x00025000	0x00029000	
virtual-size (167936 bytes)	0x00024000 (147456 bytes)	0x00004000 (16384 bytes)	0x00001000 (4096 bytes)	
characteristics	0xE0000080	0xE0000040	0xC0000040	
write	x	x	x	
execute	x	x	-	
share	-	-	-	
self-modifying	x	x	-	
virtual	x	-	-	

Packed

- String.exe gibi ikili dosya içerisindeki değerleri ASCII veya Unicode karakterlere çeviren araçlar kullanılarak dosya içeriğinde kullanılan kütüphane, fonksiyon isimleri görüntülenebiliyor. Bu nitelik de paketlenmemiş bir dosyada paketlenmiş bir ikili dosyaya kıyasla çok daha fazla sayıda anlamlı metinleri görüntülenebildiği görülüyor.

- Buradaki çıktılar doğrultusunda ikili dosyanın paketlenme süreci hakkında detaylı bilgi edinilebiliyor. Örnek olarak aşağıdaki çıktıda **GetProcAddress**, **LoadLibrary**, **UPX0**, **UPX1** gibi kısımlar verilebilir.

Unpack							
encoding (2)	size (bytes)	location	flag (7)	label (83)	group (9)	technique (5)	value
ascii	19	section:idata	-	import	synchronization	-	InitializeListHead
unicode	35	section:rdata	-	file	registry	-	api-ms-win-core-registry-l1-1-0.dll
ascii	12	section:rdata	-	-	registry	-	RegOpenKeyEx
ascii	15	section:rdata	-	-	registry	T1012 Query Registry	RegQueryValueEx
ascii	11	section:rdata	-	-	registry	-	RegCloseKey
ascii	17	section:idata	-	import	reconnaissance	T1082 System Information Discovery	IsDebuggerPresent
ascii	25	section:idata	-	import	reconnaissance	-	IsProcessorFeaturePresent
ascii	23	section:idata	-	import	reconnaissance	-	QueryPerformanceCounter
ascii	19	section:idata	x	import	reconnaissance	T1057 Process Discovery	GetCurrentProcessId
ascii	14	section:idata	-	import	reconnaissance	-	GetStartupInfo
ascii	16	section:idata	-	import	memory	-	RtlVirtualUnwind
ascii	9	section:idata	-	import	memory	-	HeapAlloc
ascii	8	section:idata	-	import	memory	-	HeapFree
ascii	14	section:idata	-	import	memory	-	GetProcessHeap
ascii	12	section:idata	-	import	memory	T1055 Process Injection	VirtualQuery
ascii	6	section:idata	-	-	memory	-	memcpyp
ascii	23	section:idata	-	import	file	T1124 System Time Discovery	GetSystemTimeAsFileTime
ascii	18	section:idata	x	import	execution	T1057 Process Discovery	GetCurrentThreadId
ascii	17	section:idata	-	import	execution	-	RtlCaptureContext
ascii	22	section:idata	x	import	execution	-	RtlLookupFunctionEntry
ascii	17	section:idata	x	import	execution	T1057 Process Discovery	GetCurrentProcess
encoding (1)	size (bytes)	location	flag (1)	label (13)	group (3)	technique (2)	value
ascii	14	section:rsrsc	x	import	memory	T1055 Process Injection	VirtualProtect
ascii	6	section:rsrsc	-	-	memory	-	memcpyp
ascii	11	section:rsrsc	-	import	execution	-	ExitProcess
ascii	14	section:rsrsc	-	import	dynamic-library	-	GetProcAddress
ascii	11	section:rsrsc	-	import	dynamic-library	T1106 Execution through API	LoadLibrary
ascii	4	-	-	utility	-	-	UPX0
ascii	4	-	-	utility	-	-	UPX1
ascii	6	section:UPX1	-	format-string	-	-	o %s,
ascii	3	section:UPX1	-	format-string	-	-	%lu
ascii	4	section:UPX1	-	file	-	-	.pdb
ascii	12	section:rsrsc	-	file	-	-	KERNEL32.DLL
ascii	13	section:rsrsc	-	file	-	-	ucrtbased.dll
ascii	17	section:rsrsc	-	file	-	-	VCRUNTIME1400.dll
ascii	40	dos-stub	-	dos-message	-	-	!This program cannot be run in DOS mode.
ascii	4	rich-header	-	-	-	-	Rich
ascii	5	-	-	-	-	-	.rsrsc
ascii	4	-	-	-	-	-	4.22
ascii	4	-	-	-	-	-	UPX!
ascii	4	section:UPX1	-	-	-	-	vmM26
ascii	3	section:UPX1	-	-	-	-	7??
ascii	4	section:UPX1	-	-	-	-	*!#N
ascii	3	section:UPX1	-	-	-	-	15m

Manual Unpacking

İkili dosyanın paketlenildiği anlaşıldıktan sonra sıkıştırılmış orijinal kod parçasını dışarı çıkarabilmek gerekiyor. Bunun için kodun bir disassembler üzerinde açıp analiz edilmesi gerekiyor. Analiz sürecinde aranması gereken ipuçları listelendiğinde;

-

- UPX (Ultimate Packer for Executables),
 - o <https://upx.github.io/>

Kaynaklar

- <https://dlnhxyz.medium.com/manually-unpacking-a-upx-packed-binary-with-radare2-part-1-7039317c2ed8>
- <https://dlnhxyz.medium.com/manually-unpacking-a-upx-packed-binary-with-radare2-part-2-be00860b5eac>
- <https://www.youtube.com/watch?v=Npm5tuy1Pp4>
- <https://tech-zealots.com/reverse-engineering/dissecting-manual-unpacking-of-a-upx-packed-file/>
- <https://www.travismathison.com/posts/Manually-unpacking-malware/>
- <https://medium.com/@0x8080/how-to-manually-unpack-a-binary-e56e18732b3b>
- <https://www.manrajbansal.com/post/manually-unpacking-a-upx-packed-binary>
- <https://www2.cs.arizona.edu/people/debray/Publications/static-unpacking.pdf>
- <https://www.youtube.com/watch?v=J9l-Sewaihs>
- <https://www.ired.team/offensive-security/defense-evasion/t1045-software-packing-upx>
- <https://tech-zealots.com/reverse-engineering/dissecting-manual-unpacking-of-a-upx-packed-file/>
- <https://www.ired.team/offensive-security/defense-evasion/t1045-software-packing-upx>
- <https://sam0x90.blog/2020/06/06/unpacking-binary-101/>