

WhiteHats

Amit Malik (DouBle_Zer0)

"There is no security on this earth, there is only opportunity."

-- General Douglas MacArthur

[INJECTOR MASK OR A TOOL]

Injector is a little tool that will inject your code into your target process. The code injection concept is not new but the thinking behind injector is new and powerful. The purpose of this document is to explain the concept behind injector and to provide a powerful technique to bypass anti viruses

Introduction:

Injector is a little tool that will inject your code into your target process. The code injection concept is not new but the thinking behind injector is new and powerful. The purpose of this document is to explain the concept behind injector and to provide a powerful technique to bypass anti viruses. Current state of technique is only valid with post exploitation phase, social engineering (eg. Execute SFX archive in silent mode with proper parameter to injector with little variation in interface –PID issues) or physical access in a Penetration test. (viruses and worms can use with variation in interface^(G)).

<u>Antivirus:</u>

Antivirus (or **anti-virus**) Software is used to prevent, detect, and remove, malware, including computer viruses, worms, and Trojan horse. Such programs may also prevent and remove adware, spyware, and other forms of malware. – Wikipedia

Antivirus play a major role for the security of a system. But for hackers/ pentesters , it creates some big problems. During the post exploitation phase we have some sort of excess on the victim machine and generally we want to upload some tools on the victim machine for batter control but here antivirus play with our tools and detect them as a malicious file and delete them. Now it may be possible if you are using your own tool then antivirus may fail to detect it but this is rare situation during a pentest. All in all if we use publicly exposed tools then there is higher probability of getting caught by antivirus.

A simple example: suppose we have a reverse shell now we know that this is a single connection to our machine from our victim and we don't want to loose it at any condition. So we upload some tools on the victim machine like netcat or other executable (may be generated from metasploit) to get a backup connection etc.. may be to upgrade our shell to merepreter shell. But our victim is using a anti virus and anti virus delete the uploaded files. Now we need something else means either bypass AV or write our own code.

But the question How antivirus detect our executables (.exe).

Executables (.EXE):

Basically exe files not only contain your code but also contain some important data that is only meaningful for a Operating system loader.

And technical name for that data is PE(portable executable) file header/ PE file format. I am not explaining pe file format here. But here is snapshot:

	0	1	2	з	-4	5	6	7	8	9	A	в	C	D	E	F	10	01234 5 6789ABCDEF0
00000000	4 D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	в8	MZ
00000011	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	00	00	@ []
00000022	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	<u></u>
00000033	00	00	00	00	00	00	00	00	00	80		00		0E	1 F	BА	ΟE	••••••
00000044	00	B4	09	CD	21	вз	01	4C	CD	21	54	68	69	73	20	70	72	IL. This pr
00000055	6 F	67	72	61	6 D	20	63	61	6 E	6 E	6 F	74	20	62	65	20	72	ogram cannot be r
00000066	75	6 E	20	69	6 E	20	44	4 F	53	20	6 D	6 F	64	65	2 E	OD.	OD	un in DOS mode
00000077	DА	24		00		00		00	00	50	45	00		4C	01	07		.Ş <mark>PE</mark> L
00000088	B 1	1E	67	4C	00	26	00	00	D2	01	0.0	00	ΕO	00	07	01	OВ	gL.&
00000099	01	02	38	00	0A	00	00	00	12	00	00	00	02	00	00	20	12	8
AAOOOOOOA	00	00	00	10	00	00	00	20	00	00	00	00	40	00	00	10	00	
OOOOOOBB	00	00	02	00	00	04	00	00	00	01	00	00	00	04	00	00	00	
000000000	00	00	00	00	00	BO	00	00	00	04	00	00	59	36	01	00	03	<u>.</u>
OOOOOODD	00	00	00	00	00	20	00	00	10	00	00	00	00	10	00	00	10	
OOOOOOEE	00	00	0.0	00	00	00	10	00	00	00	00	00	00	00	00	00	00	
OOOOOOFF	00	00	50	00	00	AC	02	00	00	00	00	00	00	00	00	00		P
00000110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000121	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000132	00	00	0.0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000143	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000154		00		00		00		00		00		00		00		00		
00000165	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
🛗 shellcodete																		

PEview:

& PEview - F:\Documents and Settings\Administrator\Desktop\shellcodes\nasm.exe												
File View Go Help												
		Data	Description	1 Mehre								
	pview	Data	Description									
MR DOB Chuk Dramman	00000000	0106	Magic Maine Linksey Manaine	IMAGE_NT_OPTIONAL_HDR32_MAGIC								
	00000002	02	Major Linker Version									
B-IMAGE_NI_HEADERS	00000003	30	Minor Linker Version									
	00000004	0003A000	Size of Code									
	00000000	000000000	Size of Uninitialized Data									
IMAGE SECTION HEADED toxt	00000000	00004600	Address of Entry Doint									
IMAGE SECTION HEADER data	00000010	00001130	Address of Entry Point									
IMAGE SECTION HEADER Idata	00000014	00030000	Base of Data									
IMAGE SECTION HEADER has	00000010	00030000	Image Bace		_							
IMAGE SECTION HEADER idata	00000010	00400000	Section Alignment		-							
SECTION text	00000020	00001000	File Alignment									
SECTION data	00000024	0000200	Major O/S Version									
	00000028	0004	Minor O/S Version									
B-SECTION idata	00000020	0001	Major Image Version									
- IMPORT Directory Table	0000002E	0000	Minor Image Version									
- IMPORT Name Table	0000030	0004	Maior Subsystem Version									
IMPORT Address Table	0000032	0000	Minor Subsystem Version									
IMPORT Hints/Names & DLL Names	0000034	00000000	Win32 Version Value									
	00000038	00094000	Size of Image		_							
	0000003C	00000400	Size of Headers									
	00000040	0008D405	Checksum									
	00000044	0003	Subsystem	IMAGE SUBSYSTEM WINDOWS CUI								
	00000046	0000	DLL Characteristics									
	00000048	00200000	Size of Stack Reserve									
	0000004C	00001000	Size of Stack Commit									
	0000050	00100000	Size of Heap Reserve									
	00000054	00001000	Size of Heap Commit									
	00000058	00000000	Loader Flags									
	0000005C	00000010	Number of Data Directories									
	00000060	00000000	RVA	EXPORT Table								
	00000064	00000000	Size									
	00000068	00093000	RVA	IMPORT Table								
	_0000006C	000007AC	Size									
	00000070	00000000	RVA	RESOURCE Table	₩.							
< · · · >	<				>							

Viewing IMAGE_OPTIONAL_HEADER

So when we execute exe, windows loader first read the PE header and on the basis of header information loader loads the file into memory.

Executable Detection (top level view):



⁽Antivirus scan the file if(malicious) then delete/etc.)

So when we copy the file on system then antivirus scan the file (check for known signatures) and if the file is malicious then delete the file.

Encrypter/packer/protector:

The above technologies (Encrypter/packer/protector) are not only to evade anti viruses. People also use them for ethical work. Basically encryptor/packers encrypt/pack the file and add the decryption/unpacking routine into the file. So when we execute the file windows loader loads the file on the basis of PE header (encryptor/packer does not pack the PE header of a file they only change the necessary values in PE header. Eg. Address of entry point, sections etc..). so the unpacking work something like this:



So unpacking/decryption routine unpack the code and then jump on the Original Entry Point (on our real code freshly recovered from unpacking routines).

So if we pack the exe with any packer then Antivirus should not be able to detect the exe?. Ok now its time to do some practical.

Pack any malicious file with UPX (a freeware packer) and then test again with Antivirus. You should see that your AV is again popping up the alert and telling that hey don't try to fool me it is malicious.. But our file was packed how AV still know that it is a malicious file and the answer is AV has also signature for UPX, so what AV is doing it loads the file detect the packer then decrypt/unpack(AV also know how to unpack UPX packed files ©) the file and then test the file, And this is the reason that AV still know that file is malicious. But hey where is proof that AV is doing all this shit to test file.

Ok to prove just open the same packed file in a hex editor and just overwrite the starting bytes with some garbage values and then again test it with your AV. Now you should see that AV is saying that file is dean (Reason: Because file is corrupted and AV have no option to unpack/decrypt it) fantastic that's all we want.

The Concept of Injector (Ingeneric way to bypass AV):

Because exe files are going to detected by AVs(at least if you pack them with the publicly exposed packers/encryptors). So we have to think in a another way..

And the another way is: split the exe into two parts (not physically)

- 1) The core code (the actual code that performs a specific task for eg. Bind shell)
- 2) The interface a mechanism that will inject the code into memory and execute that code.



So the functioning is something like this:

Note that from the above explanation we know that shellcode/code into a file is not going to be detected by AV because AV don't know how to decode shellcode. (Don't talk about Heuristic, I know AV vendors are joking ©)

Important Note: you may be thinking that why I am saying encoded shellcode because if you use metasploit shellcodes there signatures may be in AVs. If you encode the shellcode with any available encoder in metasploit then AVs not able to decode it in a file and not able to detect it (if you don't understand it read the whole stuff again ©).

Although in some cases (Eg. Avast may be with others also) AV not alert if you use shellcodes that are not encoded because AV think that txt file are lame files. But if you force fully scan the file than AV alert.

Second part of the concept is the interface that will inject the code into a process. Code injection is not a new concept (dll injection is one of the most popular example).

Note: All the things are generic and are not specific to any tool or shellcodes. Metasploit and shellcodes are used only to demonstrate the concept. You can also inject your codes "that are detectable to AV in exe mode" with this method and can bypass AV.

Things that you can do with this method:

- 1) Can backdoor a process
- 2) Can provide many backup shells (every type)

Key Benefit: we can use publically available tool (malicious) without fear.

And many-2 others things (that depends on imagination).

Work Flow of Injector



Note: With variations we can accomplish many things \circledast

[Download Injector]

https://sites.google.com/site/mamit30/home/injector

[Video Demonstration]

http://vimeo.com/14139105

References:

- 1. Metasploit http://www.metasploit.com
- 2. UPX <u>http://upx.sourceforge.net/</u>
- 3. PEview <u>http://www.magma.ca/~wjr/</u>
- 4. An In-depth Look into the Win32 Portable Executable File Format <u>http://msdn.microsoft.com/en-us/magazine/cc301805.aspx</u>
- 5. Three Ways to Inject Your Code into Another Process http://www.codeproject.com/KB/threads/winspy.aspx
- 6. Fsecure Malware Analysis Course (free) <u>https://noppa.tkk.fi/noppa/kurssi/t-110.6220/luennot</u>