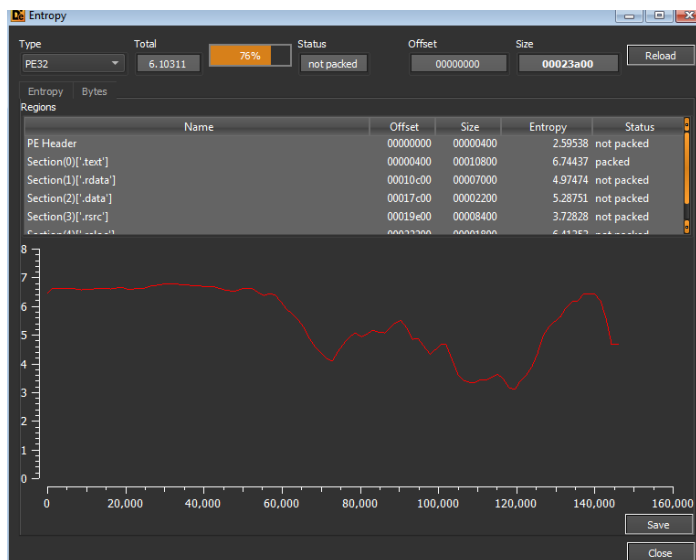


Initially we load the sample in DetectItEasy and here we can note that entropy is fairly high so sample might be packed.



Now we upload the sample in x32 dbg and apply breapoints on VirtualAlloc,VirtualProtect,IsDebuggerPresent,CreateProcessInternalW,NtResumeThread for unpacking.

Type	Address	Module/Label/Exception	State	Disassembly	Hits
Software	76371836	<kernel32.dll.VirtualAlloc>	Enabled	mov edi,edi	0
	76374317	<kernel32.dll.VirtualProtect>	Enabled	mov edi,edi	0
	76374415	<kernel32.dll.IsDebuggerPresent>	Enabled	jmp <JMP.IsDebuggerPresent>	0
	76383848	<kernel32.dll.CreateProcessInternalW>	Enabled	push esi	0
	77280058	<ntdll.dll.NtResumeThread>	Enabled	mov eax,4F	0

After hitting VirtualAlloc third time we analyse the memory return by it dump3 and we can note that it is containing a PE file compressed with aplib (M8Z), so here we will use aplib-ripper to extract pe.

Disassembly:

```

push ebp
mov ebp,esp
pop ebp
jmp <JMP.VirtualProtect>
mov ecx,dword ptr ds:[esi]
mov dword ptr ds:[eax],ecx
mov ecx,dword ptr ds:[esi+4]
mov dword ptr ds:[eax+4],ecx
jmp kernel32.76372077
mov eax,dword ptr ss:[ebp-130]
call kernel32.76372077
call <kernel32.RegKrnGetGlobalState>
jmp kernel32.76380033
nop
nop
nop
nop
nop
nop
push ebp
mov ebp,esp
push esi
xor esi,esi
push dword ptr ss:[ebp+10]
xor esi,esi
push dword ptr ss:[ebp+C]
inc esi
push dword ptr ss:[ebp+8]
call dword ptr ds:[<&NtFlushInstructionCache>]
test eax,eax

```

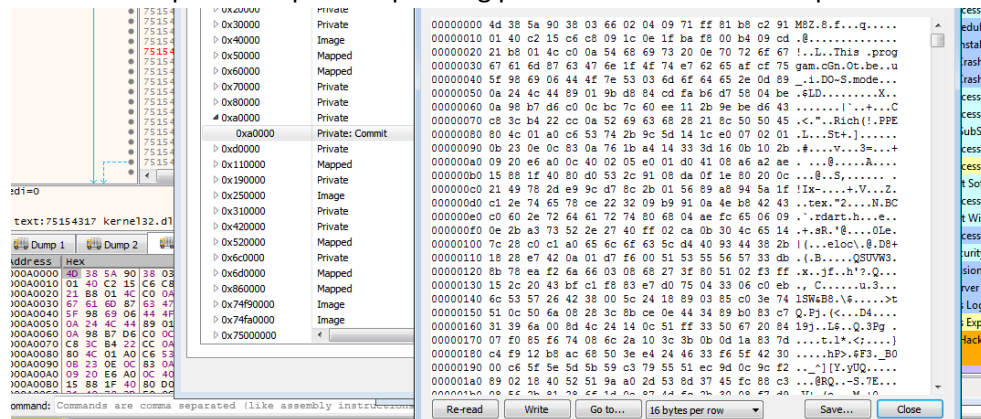
Memory Dump 3:

```

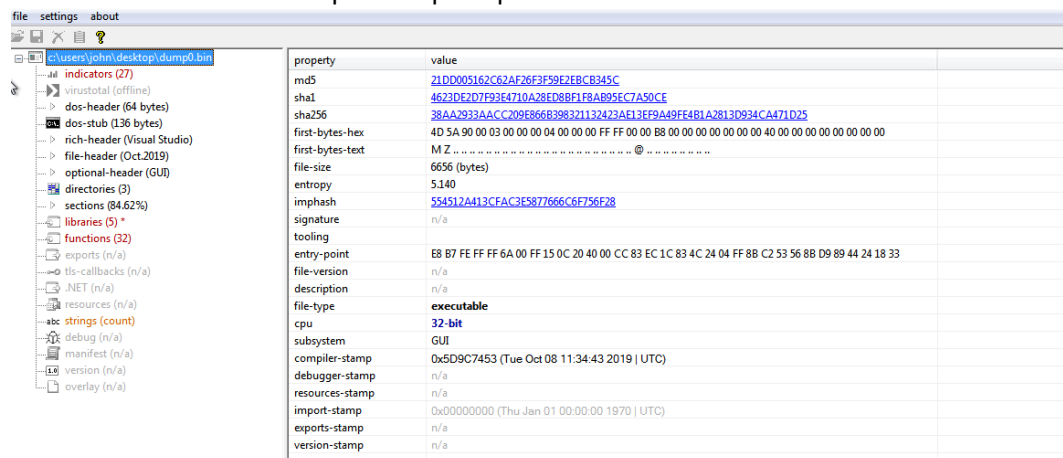
Address Hex
00110000 4D 38 5A 90 38 03 66 02 04 09 71 FF 81 B8 C2 91 MZ
00110001 01 04 C2 15 C6 C8 09 1C 0E 1F BA F8 00 B4 09 CD 0A
00110002 21 B8 01 4C C0 0A 54 68 69 73 20 0E 70 72 6F 67 .LA.This.prog
00110003 67 63 62 87 63 47 6E 1F 4F 74 67 62 65 AF CF 75 gah.cgn.Orcpe.Tu
00110004 5F 98 69 06 44 4F 7E 53 03 6D 6F 64 65 2E 00 89 .1.D0-S.mode..
00110005 0A 24 4C 44 89 01 9B D8 84 CD FA 86 D7 5B 04 BE .S.D..0.1.unk.M
00110006 0A 98 B7 D6 C0 0A 9C 7C 6D E5 11 28 9E 8E D6 43 .0A..1..XOC
00110007 C8 3C 84 22 CC 0A 52 69 63 68 28 21 8C 50 50 45 Ec"1.Rich(.PPE
00110008 80 4C 01 A0 C6 53 74 2B 9C 50 14 1C E0 07 02 01 .L.AStc.).B...
00110009 08 23 0E 0C 83 0A 76 1B A4 14 33 30 16 08 10 28 .W...V..H..3...+
0011000A 09 20 E6 A0 0C 40 02 05 E0 01 D0 41 08 A6 A2 AE .e..8..a.DA.4e
0011000B 15 88 1F 40 8D D0 53 2C 91 08 DA 0F 1E 80 20 0C ...B.D3...U...

```

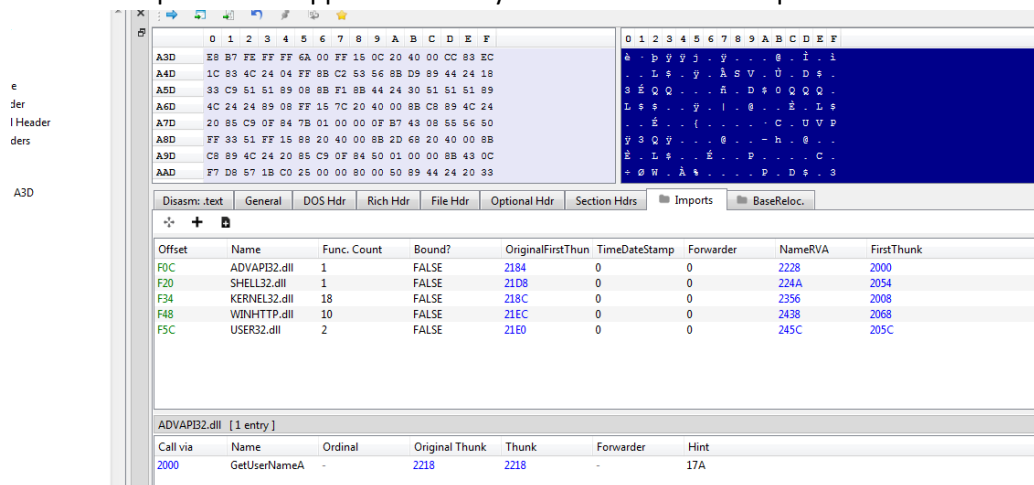
Here we dump the compressed pe using process hacker and decompress it.



Now we loaded the decompressed pe in pestudio.



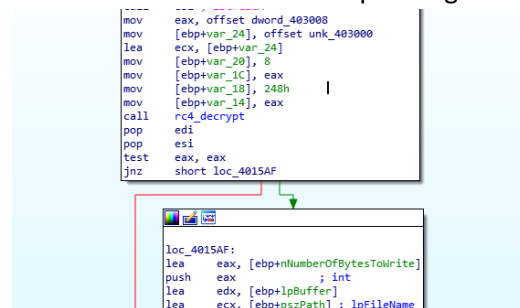
Here as sample is not mapped in memory we can view all the imports are intact.



Here we can note suspicious APIs it might be trying to make network communication to C2 server.

HeapReAlloc	-	-	kernel32.dll
wsprintfA	-	-	user32.dll
wsprintfW	-	-	user32.dll
WriteFile	x	-	kernel32.dll
VirtualProtect	x	-	kernel32.dll
WinHttpCloseHandle	x	-	winhttp.dll
WinHttpSetOption	x	-	winhttp.dll
WinHttpOpenRequest	x	-	winhttp.dll
WinHttpSendRequest	x	-	winhttp.dll
WinHttpQueryHeaders	x	-	winhttp.dll
WinHttpOpen	x	-	winhttp.dll
WinHttpReceiveResponse	x	-	winhttp.dll
WinHttpQueryDataAvailable	x	-	winhttp.dll
WinHttpConnect	x	-	winhttp.dll
WinHttpReadData	x	-	winhttp.dll

In IDA we can note that arguments to rc4_decrypt are passed and one argument is pointing to location 403000 and another two are pointing to the location 403008 so, 403000 can contain key of rc4_crypt.



```

from arc4 import ARC4
import pefile

def config_extract(file):
    pe = pefile.PE(file)
    for section in pe.sections:
        if b".data" in section.Name:
            return section.get_data()

def rc4_decrypt(key, data):
    cipher = ARC4(key)
    decrypt = cipher.decrypt(data)
    return decrypt.replace(b'\x00', b'')

def main():
    file = input("file: ")
    data = config_extract(file)
    key = data[:8]
    data = data[8:592]
    print(rc4_decrypt(key, data))

if __name__ == "__main__":
    main()

```

```

b'\xfb3=\x1e\x02/index.php\x13boldidiotruss.xyz\x0fnizaoplov.xyz\x0f153ishak.best\x10ilu21plane.xyz'

```