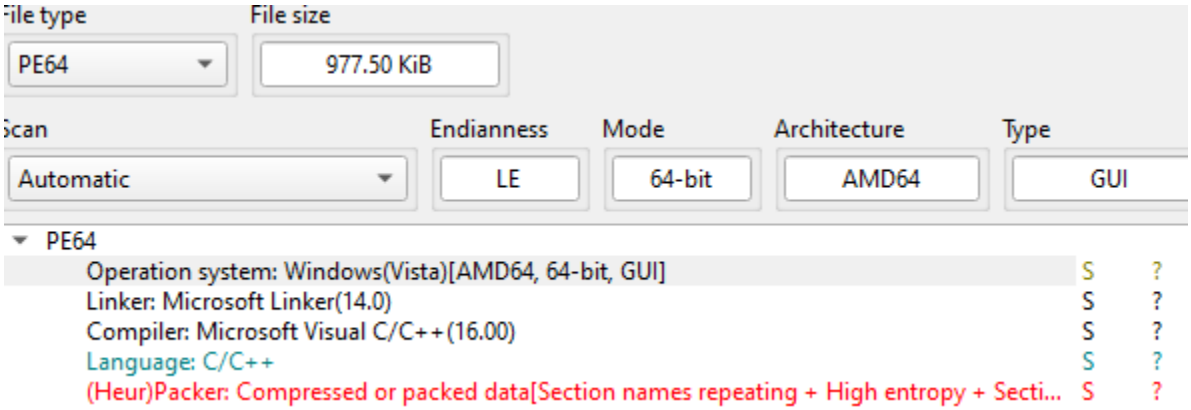
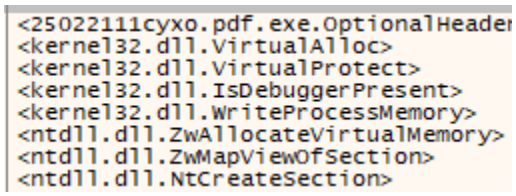
Unpacking Process Injection - DarkCloud
5537D725807584D2B92E993A1B5F5430



Breakpoints set.



First VirtualAlloc followed by A call to `VirtualAlloc` was observed, which allocated a region of memory within the target process's address space. This was immediately followed by a call to `WriteProcessMemory`, where the `lpBuffer` parameter pointed to data that was written into the memory region returned by `VirtualAlloc`.

This behavior is characteristic of code injection techniques, where memory is first allocated with `VirtualAlloc` (typically with `PAGE_EXECUTE_READWRITE` or `PAGE_READWRITE` protection), and then malicious or arbitrary code is written into that memory using `WriteProcessMemory`.

The key detail is that the **`lpBaseAddress`** argument in `WriteProcessMemory` matched the **address returned by `VirtualAlloc`**, indicating a deliberate setup for injecting or staging code. The `lpBuffer` in this case pointed to the source of the payload, either from the current process or a loaded buffer.

```
RIP RAX  00007FFDF391BCB0  48:FF25 69580400   jmp qword ptr ds:[<WriteProcessMemory>]   WriteProcessMemory
00007FFDF391BCD0  48:FF25 B1630400   jmp qword ptr ds:[<ZombifyActCtx>]   ZombifyActCtx
qword ptr ds:[00007FFDF3961820 <kernel32.WriteProcessMemory>]=<kernelbase.WriteProcessMemory>

.text:00007FFDF391BCB0 kernel32.dll:$3BCB0 #3B0B0 <WriteProcessMemory>
```

Dump 1 | Dump 2 | Dump 3 | Dump 4 | Dump 5 | Watch 1 | [x=] Locals | Struct

Address	Hex	ASCII
0000020A68AF0000	0B 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000020A68AF0010	00 00 00 00 00 00 00 00 00 00 00 00 7F 02 00 00	.....
0000020A68AF0020	00 00 00 00 FF FF 00 00 00 00 00 00 00 00 00 00	....yy.....
0000020A68AF0030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000020A68AF0040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000020A68AF0050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000020A68AF0060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000020A68AF0070	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000020A68AF0080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000020A68AF0090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000020A68AF00A0	00 00 00 00 00 B0 8F 00 00 00 00 00 00 00 00 00	.....'.....
0000020A68AF00B0	3A 9A 61 00 00 00 00 00 50 50 0D 77 23 00 00 00	:.a....PP.w#...
0000020A68AF00C0	02 02 00 00 38 FD 7A 00 28 00 00 00 00 00 00 00	....8yz.+.....
0000020A68AF00D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000020A68AF00E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000020A68AF00F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....

0000008C2618EF38	00007FF6D6DFB86F	return to 25022111
0000008C2618EF40	0000000000000000	
0000008C2618EF48	00007FF6D6DFD732	return to 25022111
0000008C2618EF50	0000020A68AF0000	
0000008C2618EF58	0000000000001000	
0000008C2618EF60	0000000000000000	
0000008C2618EF68	0000008C2618F5B0	
0000008C2618EF70	00007FF6D6E23CD9	25022111cyxo.pdf.0
0000008C2618EF78	00007FF6D6DF722E	return to 25022111
0000008C2618EF80	00007FFDF38F7034	return to kernel32
0000008C2618EF88	00007FFDF5804A24	ntdll.NlsMbCodePag
0000008C2618EF90	0000008C2618F050	
0000008C2618EF98	0000008C2618F550	
0000008C2618EFA0	0000008C2618F5B0	
0000008C2618EFA8	0000008C2618F5A8	
0000008C2618EFB0	0000000000000000	
0000008C2618EFB8	0000008C2618EFC0	
0000008C2618EFC0	0000008C2618F568	
0000008C2618EFC8	0000008C2618F560	
0000008C2618EFD0	0000000000000000	

Parameters of WriteProcessMemory from ChatGPT

c

Copy Edit

```c
BOOL WriteProcessMemory(
  HANDLE  hProcess,
  LPVOID  lpBaseAddress,
  LPCVOID lpBuffer,
  SIZE_T  nSize,
  SIZE_T  *lpNumberOfBytesWritten
);
```

🧠 Parameter Breakdown:

Parameter	Description
`hProcess`	Handle to the **target process** (must have `PROCESS_VM_WRITE` and `PROCESS_VM_OPERATION` rights)
`lpBaseAddress`	Pointer to the **target address in the remote process** where you want to write
`lpBuffer`	Pointer to the **local buffer** containing the data you want to write
`nSize`	Number of bytes to write
`lpNumberOfBytesWritten`	Optional pointer to receive how many bytes were written (can be `NULL`)

lpBuffer parameter points to the address on the allocated memory from VirtualAlloc

Current register Values. RSI is pointing to MSBuild.exe

```
RAX   00007FFDF391BCB0    <kernel32.WriteProcessMemory>
RBX   0000000055862785
RCX   00000000000000B0    '°'
RDX   0000000000400000
RBP   0000008C2618F7A0
RSP   0000008C2618EF38
RSI   0000008C2618F4E0    "KC:\\Windows\\Microsoft.NET\\Framework\\v4.0.30319\\MSBuild.exe"
RDI   00000000A608C8E3

R8    0000020A68A33E80
R9    0000000000001000
R10   00007FFDF572D304    ntdll.00007FFDF572D304
R11   0000000000000246    L'Ƶ'
R12   0000000094374349
R13   00000000B6F832C6
R14   0000000000000001
R15   000000002B430F34

RIP   00007FFDF32A05F0    <kernelbase.WriteProcessMemory>

RFLAGS   0000000000010202
ZF 0  PF 0  AF 0
OF 0  SF 0  DF 0
CF 0  TF 0  IF 1

LastError   00000000 (ERROR_SUCCESS)
LastStatus  C0000139 (STATUS_ENTRYPOINT_NOT_FOUND)

GS 002B  FS 0053
ES 002B  DS 002B
CS 0033  SS 002B
```

Default (x64 fastcall) 5

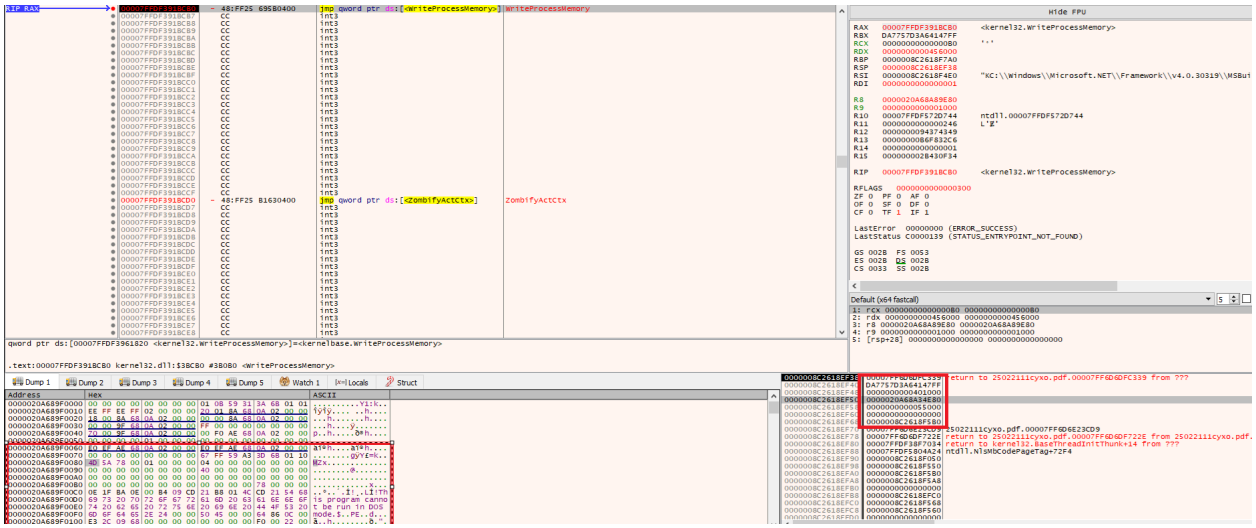
Apparently MSBuild is already running. I forgot to put the breakpoints for Process Creation (CreateProcessA, CreateProcessW, CreateProcessInternalW etc)

svchost.exe	1560		2.59 MB	NT AUTHORITY\SYSTEM	Host Proce
sihost.exe	4036		6.8 MB	DESKTOP-26...	Shell Infrast
x64dbg.exe	1276	0.07	43.89 MB	DESKTOP-26...	x64dbg
25022111CY...	4044		2.38 MB	DESKTOP-26...	Microsoft V
MSBuild....	1136		804 kB	DESKTOP-26...	MSBuild.ex

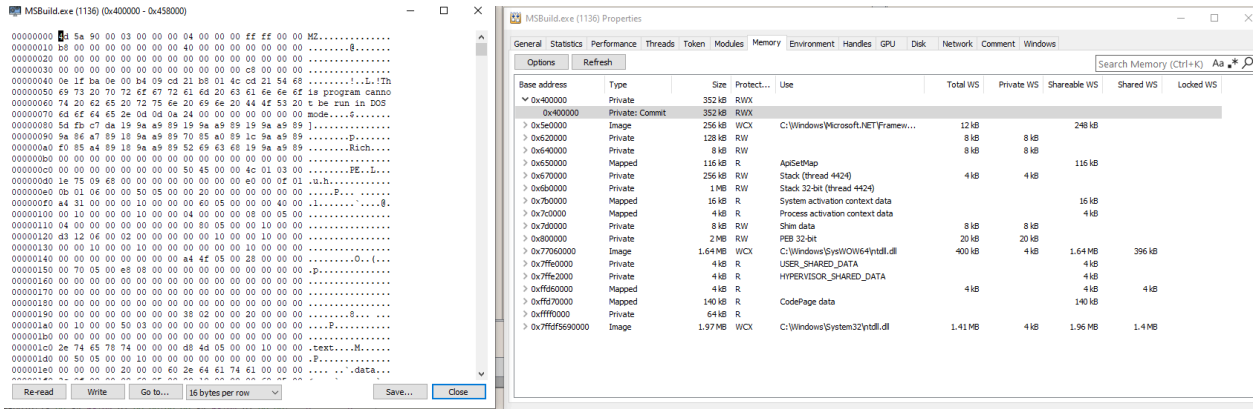
Process Writing to MSBuild.exe

```
00007FFDF32A05F0   48:8BC4          mov rax,rsp                          WriteProcessMemory
00007FFDF32A05F3   48:8958 08       mov qword ptr ds:[rax+8],rbx
00007FFDF32A05F7   4C:8948 20       mov qword ptr ds:[rax+20],r9
00007FFDF32A05FB   4C:8940 18       mov qword ptr ds:[rax+18],r8
00007FFDF32A05FF   48:8950 10       mov qword ptr ds:[rax+10],rdx
00007FFDF32A0603   55               push rbp
00007FFDF32A0604   56               push rsi                             rsi:"KC:\\Windows\\Microsoft.NET\\Framework\\v4.0.30319\\MSBuild.exe"
00007FFDF32A0605   57               push rdi
```

WriteProcessMemory is called again and the lpBuffer has the MZ signature header of 4D 5A indicating that an executable is being loaded into memory into MSBuild.exe

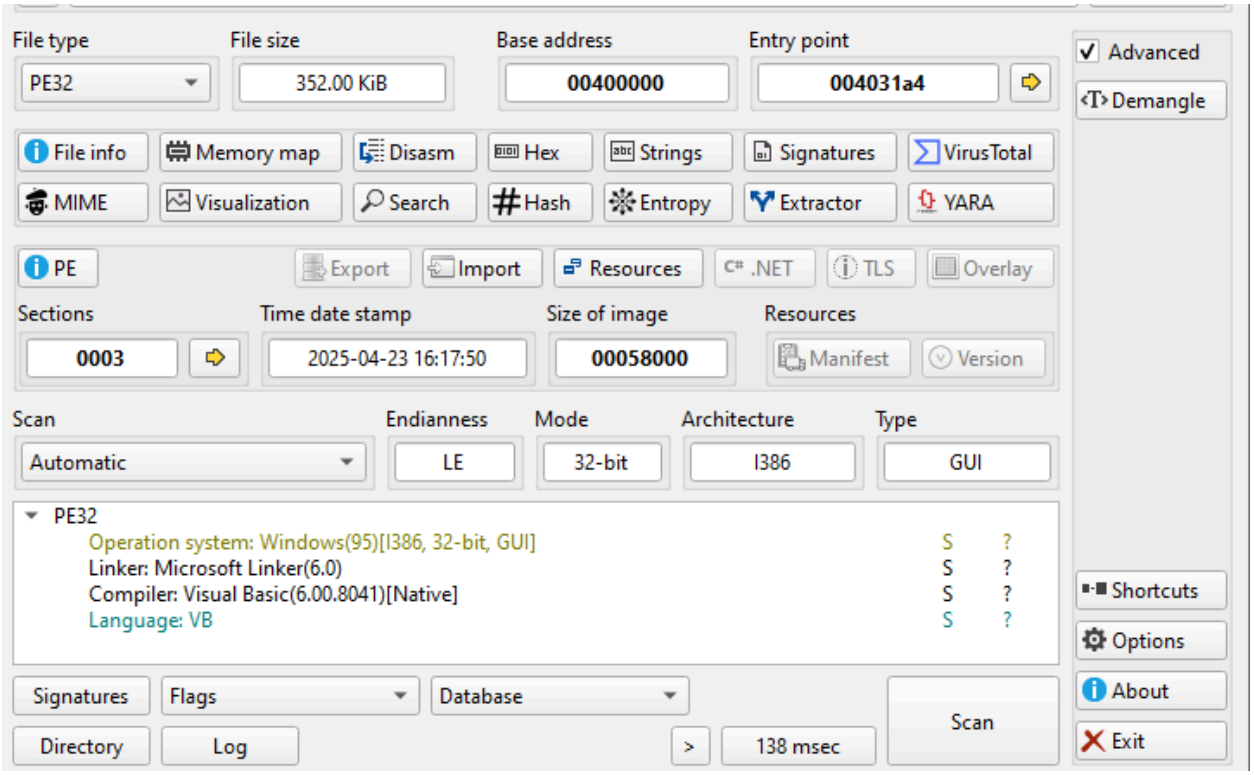


Looking into the memory of MSBuild.exe using SystemInformer shows that the executable is being loaded into it at the base address of 0x400000. Note in the WriteProcessMemory call above, lpBaseAddress is pointing to 0x401000 for the next run.

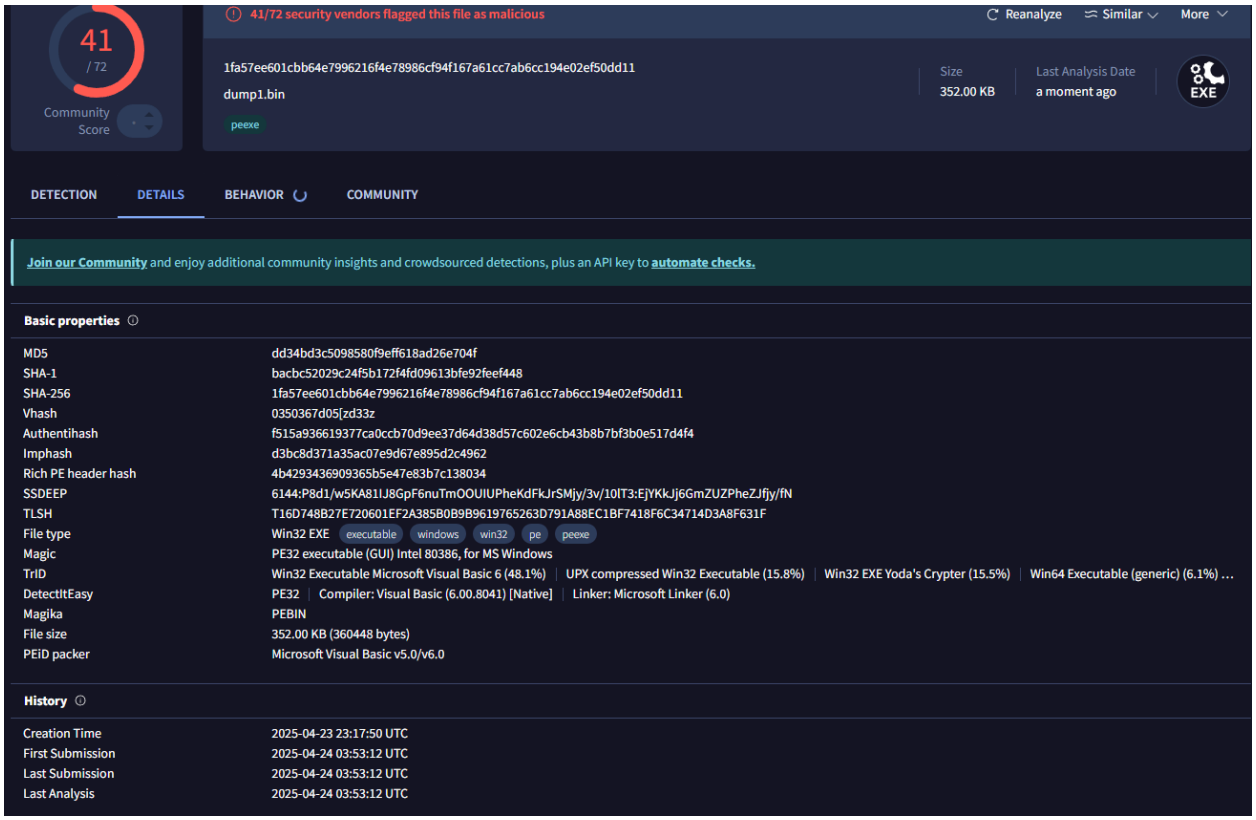


It's possible to save and dump the file from SystemInformer by clicking on save and creating a new file from that region of memory.

Dumped file seems to be written in VB and is now 32bit instead of 64bit like the previous one.

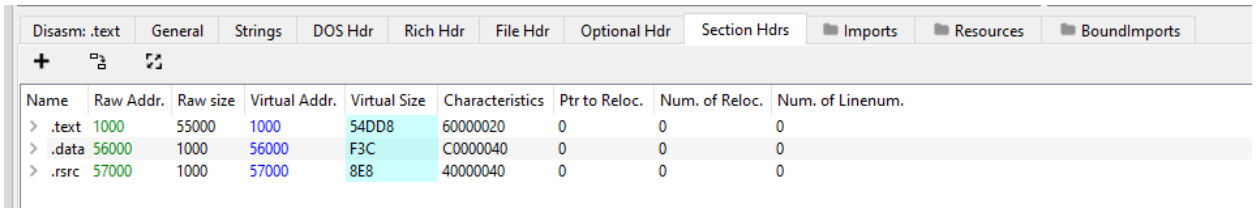


VT was able to analyze this file with a lot of detection and some vendors labelling it as Dark Cloud.



PE-Bear on the dumped file. To fix the import address table; the Raw address is initially changed to match the virtual address and subtracting the values of the raw addresses (Next raw address subtracted by the previous one) and applying the value to raw size. Afterwards, match the virtual size with the raw size and try to fill in the memory for the last slot in the raw size. The image base in the Optional Hdr tab may also be needed to match the address from where we dumped.

Looks like there is no need to repair the IAT.



Name	Raw Addr.	Raw size	Virtual Addr.	Virtual Size	Characteristics	Ptr to Reloc.	Num. of Reloc.	Num. of Linenum.
.text	1000	55000	1000	54DD8	60000020	0	0	0
.data	56000	1000	56000	F3C	C0000040	0	0	0
.rsrc	57000	1000	57000	8E8	40000040	0	0	0

Some clear text configurations can be found running Strings on the dumped file

```
00031BA0  rollandcraig32@gmail.com
00031BD8  order@oleonidas.gr
00031C04  +}bVmK6i0mnd
00031C24  mail.oleonidas.gr
```

```
00031C64  @StrBotToken
00031C84  @ChatID
00031C98  \Microsoft\Windows\Templates\firefly.exe
00031CF0  HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce\
00031D80  boos
00031D8C  RegWrite
00031F2C  http://showip.net
00031F54  http://www.mediacollege.com/internet/utilities/show-ip.shtml
00031FF0  ?id=
0003200C  Uploader
00032024  POST
```

Seems like in the original dump1.bin there were two executables inside. I split them and did a quick look and it looks like the payloads are packed by upx.

DumpDarkCloudMaybe.bin	4/23/2025 10:23 PM	BIN File	340 KB
DumpDLLMaybe.bin	4/23/2025 10:22 PM	BIN File	13 KB

```
PE32
    Operation system: Windows(2000)[I386, 32-bit, DLL]          S  ?
    Linker: Microsoft Linker(9.00.21022)                        S  ?
    Compiler: Microsoft Visual C/C++(15.00.21022)[C]            S  ?
    Language: C                                                 S  ?
    Tool: Visual Studio(2008)                                   S  ?
    Packer: UPX(3.03)[NRV,brute]                                S  ?
    (Heur)Packer: Compressed or packed data[Imports like UPX (v2.90-3.XX) + Sections like UP...  S  ?
    Overlay: Binary[Offset=0x00028600,Size=0x0002c6f0]
        Unknown: Unknown                                        S  ?
```

Strings content is still the same but Ida can interpret the dumped files properly now.