



Dynamic Malware Analysis

Sample Name: ASKBot.exe

Analysis Type: Behavioral Analysis using Different Tools

Environment: Isolated Virtual Machine (Windows 10, No Internet Access)

PRELIMINARY SETUP :

The analysis was performed in a fully isolated virtual environment to ensure system integrity and avoid unintentional malware propagation. The Windows 10 VM was configured with HOA. Shared folder access, drag-and-drop, and clipboard sharing were disabled. System protections like Windows Defender were turned off to ensure malware behavior could be observed unimpeded. A snapshot of the VM was taken

Table of Contents

1. Introduction

- Environment Setup
- Briefly introduces ASKBot.exe
- objectives of the analysis, and tools used.

2. Task 3.1 – Run and Observe

- Documents dynamic analysis using Procmon, Wireshark etc.
- monitor real-time behavior of the malware.

3. Task 3.2 – Debugging with x64dbg

- Involves stepping through the binary in x64dbg to uncover internal logic.
- API calls, and hidden behavior.

1. Introduction

This report presents a detailed dynamic analysis of a suspicious binary file named **ASKBot.exe**, which is believed to demonstrate malicious behavior. The primary goal of this analysis is to investigate how the malware interacts with the operating system, including changes to the file system, registry

The tools employed include **Procmon** for monitoring file and registry event, **Wireshark** for inspecting network traffic, and **x64dbg** for low-level debugging.

1.1 Purpose

The purpose of this report is to conduct a **practical, tool-based investigation** of the ASKBot malware. It aims to **identify any malicious behavior** through observable file system, registry, and network activity. The report will also determine whether the malware uses **evasion techniques**, attempts persistence, or initiates any C2 communications.

2. Task 3.1 – Run and Observe

In this section, the sample will be analyzed using various dynamic analysis tools such as **Procmon**, **Regshot**, and others.

2.1 Regshot Analysis

Regshot is a lightweight tool that lets you take a snapshot of the **Windows registry and file system**,

Regshot taken before running exe file

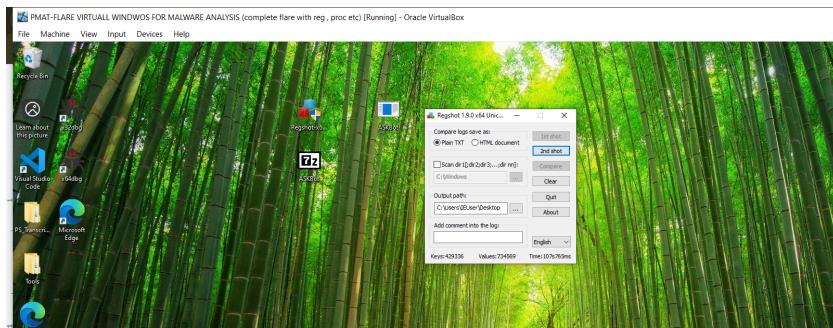


Figure 1

2nd regshot taken after running

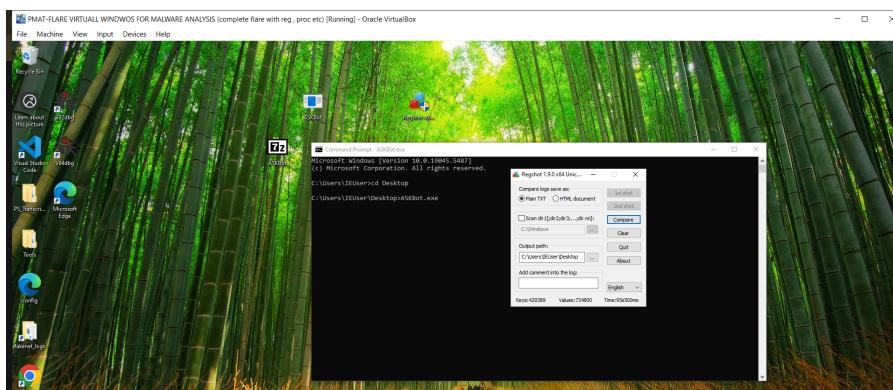


Figure 2

Compare:

Key Changes: So total 351 changes happened

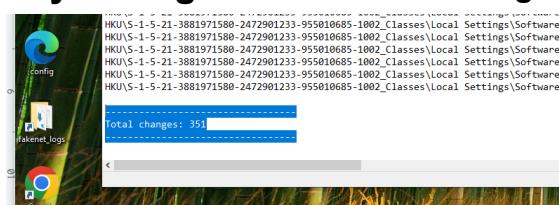


Figure 3

→ Keys deleted: 2

→ Keys added: 55

→ Values deleted: 2

Meaningfull Findings

1:

Figure 4

The malware created multiple entries under **Group Policy ServiceInstances** in above (Figure 4), it suggesting it may be setting up a form of system-level persistence.

2:

Figure 5

New entries under **Shell\BagMRU** and **Shell\Bags** indicate folder view changes, possibly used to track **user activity or disguise malicious file placement**. Such behavior is often linked to stealth tactics in file-based malware

3:

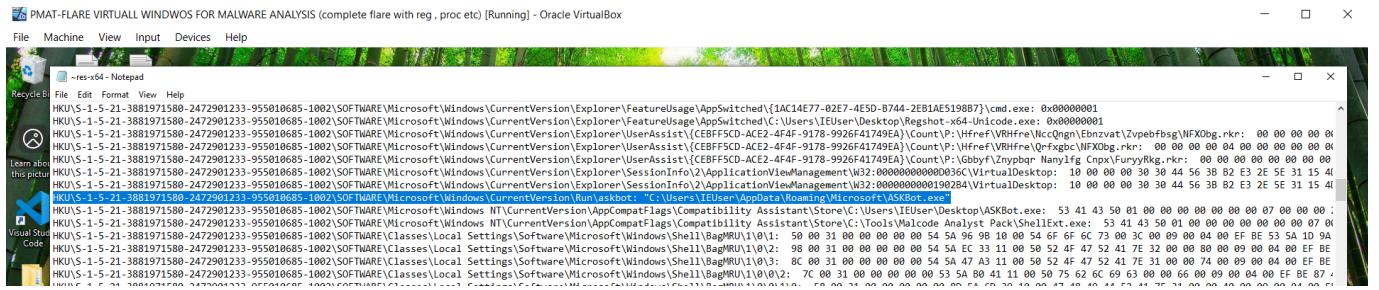


Figure 6

One of the most critical entries:(above Figure 6)

This confirms persistence — the malware is configured to execute automatically at startup for the current user by using the Run key.

4:

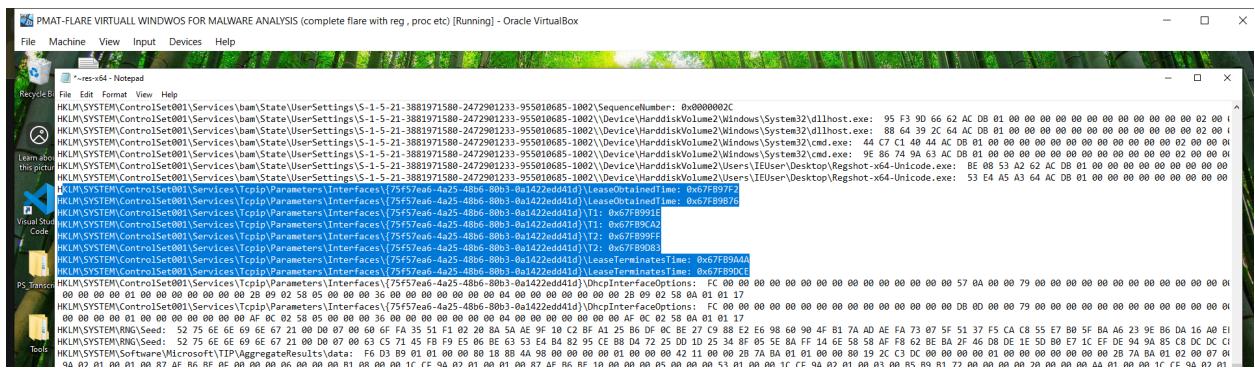


Figure 7

DHCP and lease-related values changed:

- LeaseObtainedTime, T1, T2, LeaseTerminatesTime

Implies potential manipulation or monitoring of **network configurations** or **DHCP behavior**. Changes to network-related registry keys (e.g., DHCP settings) might point to attempts to reroute or intercept network traffic.

2.2 wirshark Analysis

Network communication indications through strings

```
PAT-M-FLARE VIRTUAL WINDWOS FOR MALWARE ANALYSIS (complete flare with reg , proc etc) [Running] - Oracle VirtualBox  
File Machine View Input Devices Help  
# 1495 matches found... C:\Users\flUser\Desktop\ASXBot.exe  
Find All Save As Hex Size [+] Rescan [cancel] < Prev Page Next Page > More ...  
  
0000A8E6 api-ms-win-crt-edio-l1-1-0.dll  
0000A906 api-ms-win-crt-heap-l1-1-0.dll  
0000A926 api-ms-win-crt-math-l1-1-0.dll  
0000A948 api-ms-win-crt-convert-l1-1-0.dll  
0000A964 api-ms-win-crt-math-l1-1-0.dll  
0000A970 api-ms-win-crt-locale-l1-1-0.dll  
0000A97C api-ms-win-crt-time-l1-1-0.dll  
0000A980 GetProcAddress  
0000A9C8 GetCurrentThreadId  
0000A9F4 GetSystemTimeAsFileTime  
0000B014 InitializeListHead  
0000B024 RtlVirtualAllocEx  
0000B038 RtlLookupFunctionEntry  
0000B052 RtlVirtualAlloc  
0000B068 RtlVirtualAllocEx  
0000B072 UnhandledExceptionFilter  
0000B094 SetUnhandledExceptionFilter  
0000B0A0 RtlVirtualAllocEx  
0000B0D0 memcmp  
0000B0D4 _dclass  
0000B0D8 _dtor  
0000B250 Software\Microsoft\Windows\CurrentVersion\Run  
0000B280 APFDATA  
0000B290 cryptsp.dll  
0000B298 cryptbase.dll  
0000B2A4 exit  
0000B2B0 cryptui.dll  
0000B2B8 C:\WINDOWS\SYSTEM32\cmd.exe  
0000B2D0 Connection: close  
0000B2D8 cryptbase.dll  
0000B300 Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36  
0000B378 ntdll.dll  
0000B380 cryptui.dll  
0000B3A4 false  
0000B3AC %.17g  
0000B3B0 true  
0000B3E0 null  
0000B3E0 true  
0000B3F0 null  
0000B3F8 uv04  
0000B400 uv05g  
0000B408 null  
0000B410 %.4i.%i  
0000B418 null  
0000B490 ewogICAgImRtbil6ICjhc2tib3QukNwiCn0#  
0000B4B8 <xml version='1.0' encoding='UTF-8' standalone='yes'?>
```

Figure 8

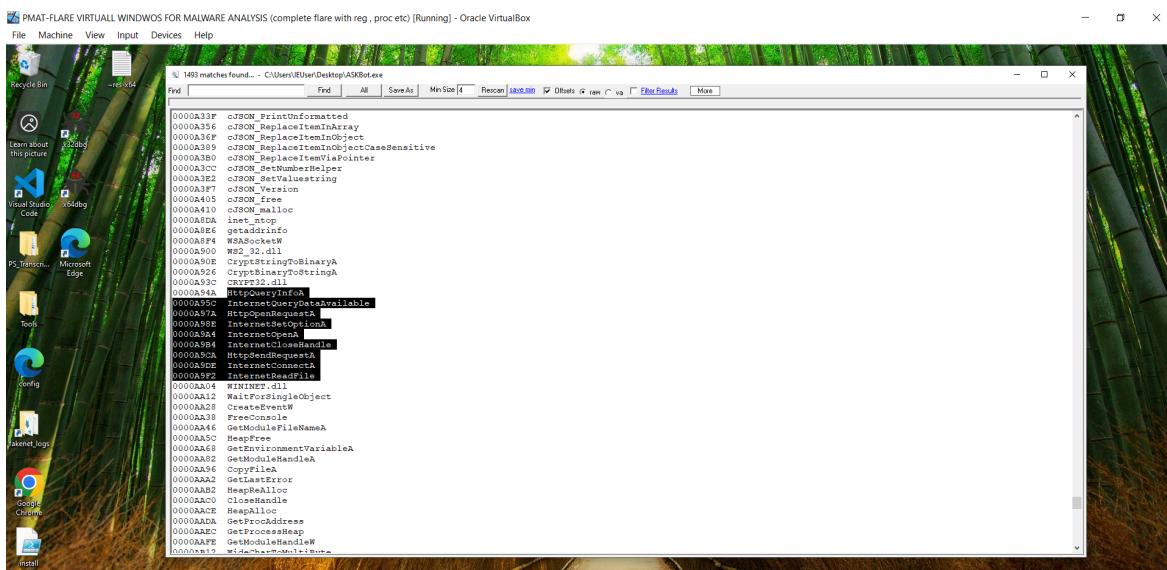


Figure 9

These API calls are used by malware to connect to the internet, send requests, and read responses as you can see in above **Figure 9**, And Also These show it's using raw HTTP communication etc.

Setup the FakeNet (INETSIM) ON Remnux

Firstly i make the separate adapter for separate connection from the real internet

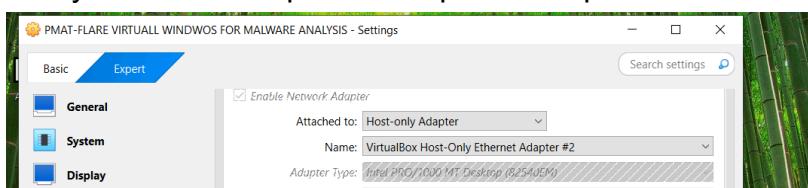


Figure 10

Then paste the address of the remnux in the dns of ipv4 for the fake dns service

```

PMAT-remnux [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Activities Terminal Apr 14 08:53
remnux@remnux: ~
inet 10.1.1.25/24 brd 10.1.1.255 scope global dynamic enp0s3
    valid_lft 547sec preferred_lft 547sec
inet6 fe80::a00:27ff:fe00:e992/64 scope link
    valid_lft forever preferred_lft forever
remnux@remnux: $ inetsim
INetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg
Using log directory:      /var/log/inetsim/
Using data directory:     /var/lib/inetsim/
Using report directory:   /var/log/inetsim/report/
Using configuration file: /etc/inetsim/inetsim.conf
Parsing configuration file.
Configuration file parsed successfully.
== INetSim main process started (PID 1486) ==
Session ID: 1486
Listening on: 10.1.1.25
Real Date/Time: 2025-04-14 08:52:49
Fake Date/Time: 2025-04-14 08:52:49 (Delta: 0 seconds)
  Forking services...
* dns_33_tcp_udp - started (PID 1490)
* smtp_25_tcp - started (PID 1494)
* smtsp_25_tcp - started (PID 1495)
* pop3_110_tcp - started (PID 1495)
* pop3s_995_tcp - started (PID 1496)
* ftp_21_tcp - started (PID 1497)
* ftpts_990_tcp - started (PID 1498)
* http_80_tcp - started (PID 1491)
* https_443_tcp - started (PID 1492)
done.
Simulation running.

```

Figure 11

Run the INETSIM

Now everything is done so now open the wireshark then start the **ASKBot.exe** then observe the traffic in the wireshark

capture:

So i see the lot of traffic in wireshark after running the exe file

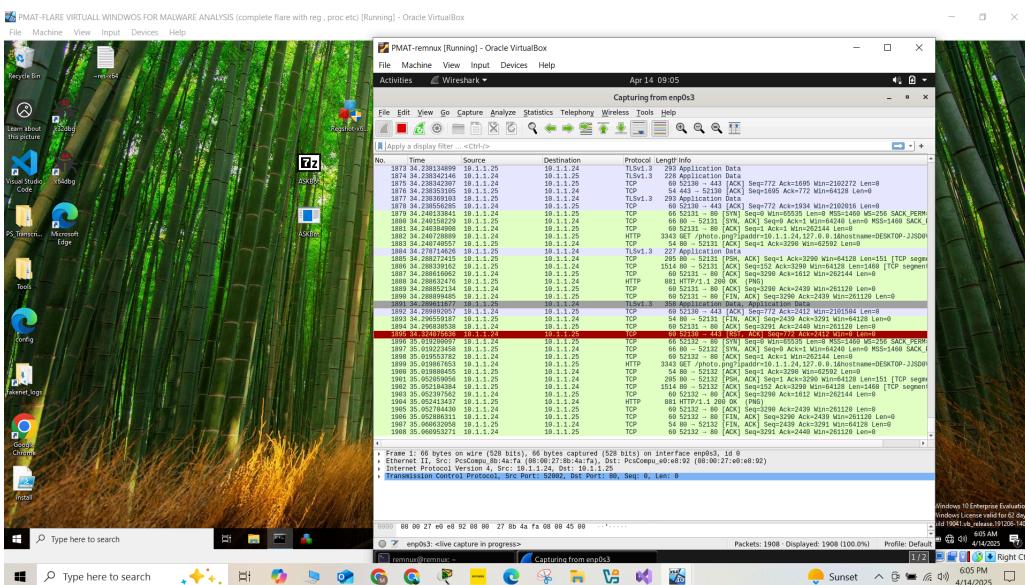


Figure 12

Let analyze the meaningfull traffic in it

First of all i use filter (**http**) because we now that we have the indicator of http in string

```
00009220 %s?ipaddr=%s&hostname=%s&procs=%s
00009248 photo.png
00000100 DDCD
```

It indicate that it can downloads the photo from the internet using http

```
00009268 C:\Windows\system32\cmd.exe
000092D8 Connection: close
000092F0 HTTP/1.1
00009300 Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36
00008378 ntdll.dll
00008388 NTQuerySystemInformation
```

In wireshark

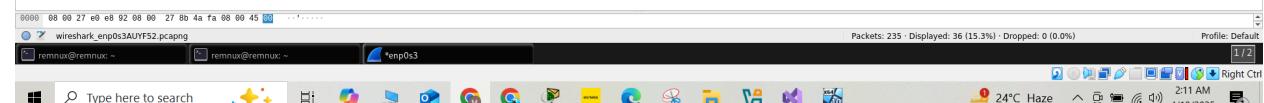
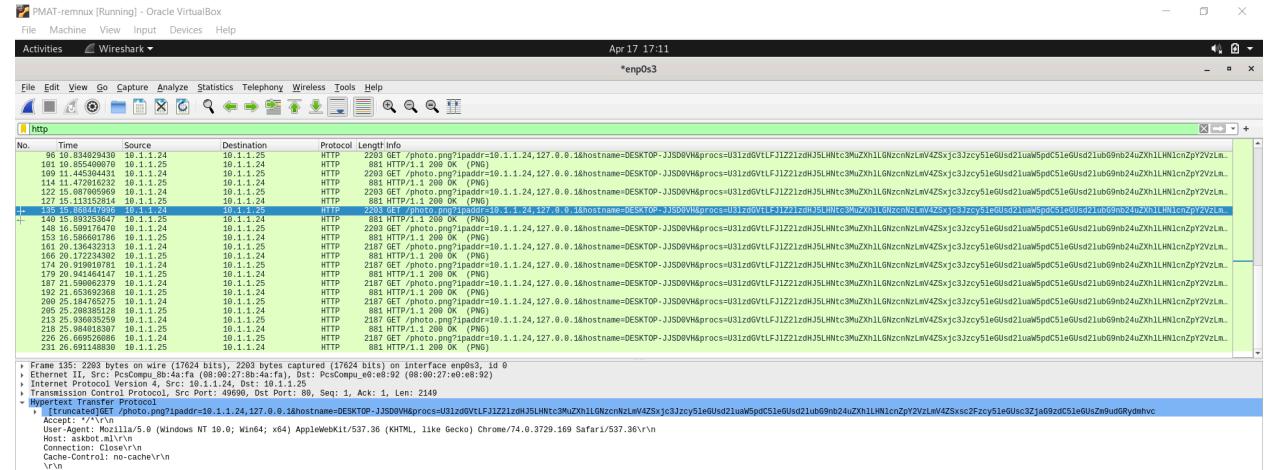


Figure 13

Look **Figure 13** that http request for downloading from malware but my Inetsim give him fake response

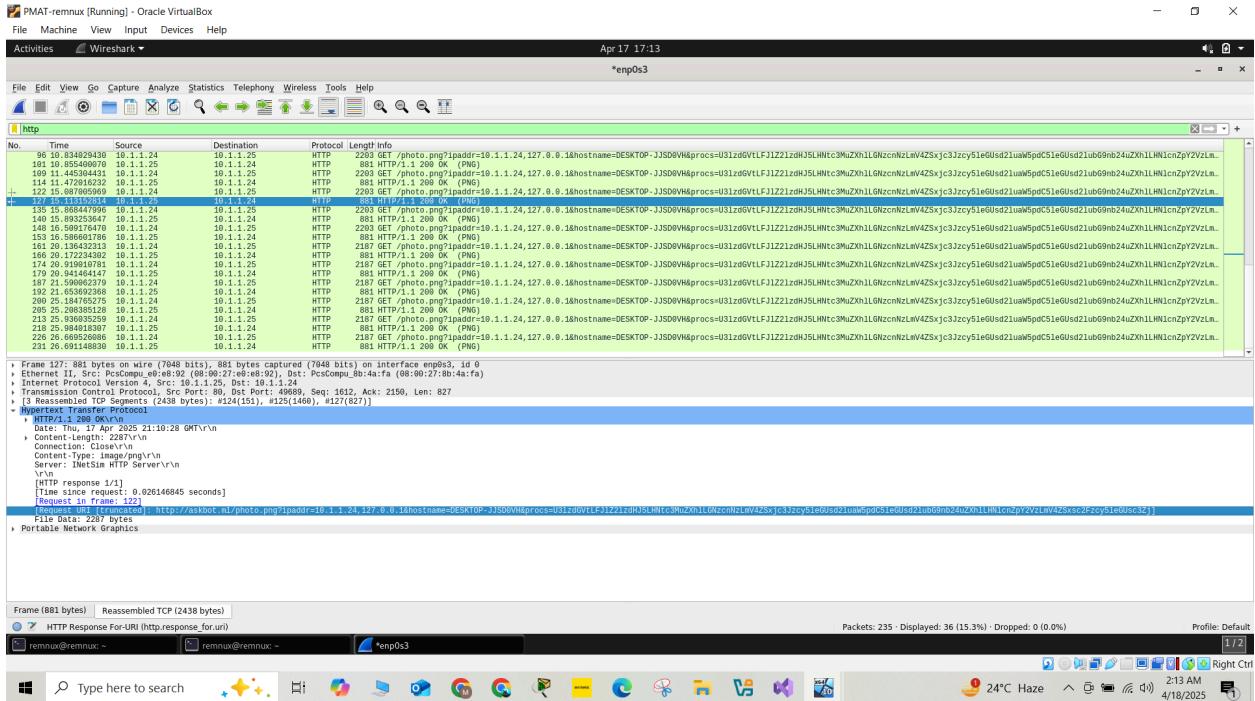


Figure 14

Next filter i use is “**http.user_agent contains "Mozilla/5.0"**” as you can see Figure 20 Because this indicator was given in the strings , its means that i also get request from the mozilla browser for png photo

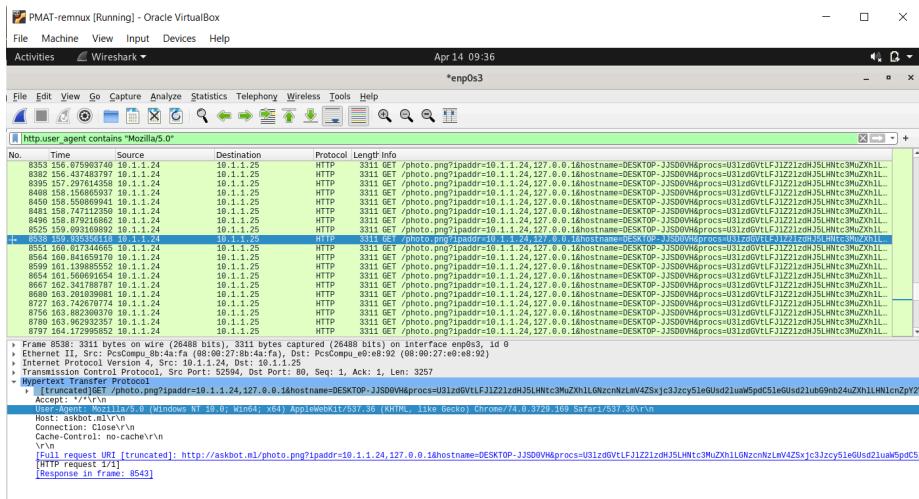


Figure 15

Summary

The malware attempted multiple outbound connections to suspicious IP addresses over HTTP.Unusual DNS queries were observed,

2.3 PROCMON Analysis

The purpose of using Process Monitor (Procmon) in this analysis is to observe and capture real-time system-level events generated by the execution of **ASKBot.exe**. This includes monitoring file system activity, registry modifications, and process behavior

Procmon Filter 1: (Process Name is ASKBot.exe)

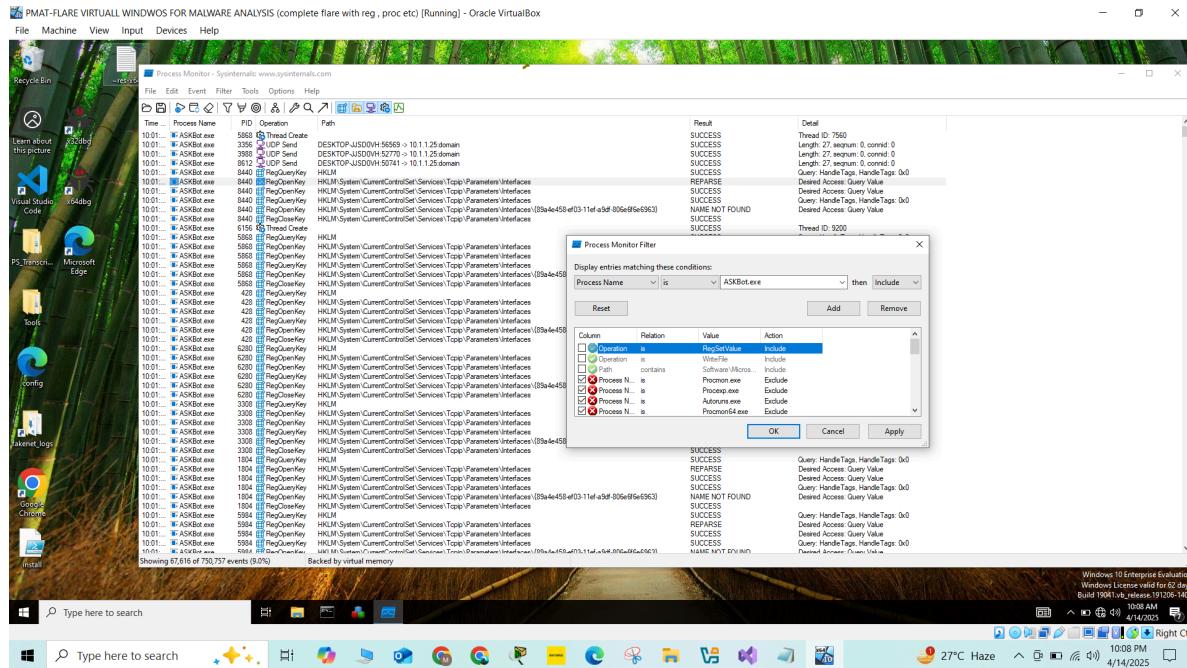


Figure 16

In **Figure 16** This filter isolates only the events generated by the malware process. It helps remove noise from other system processes. It ensures a focused view on the target executable's behavior

Procmon Filter 2: (Operation is CreateFile)

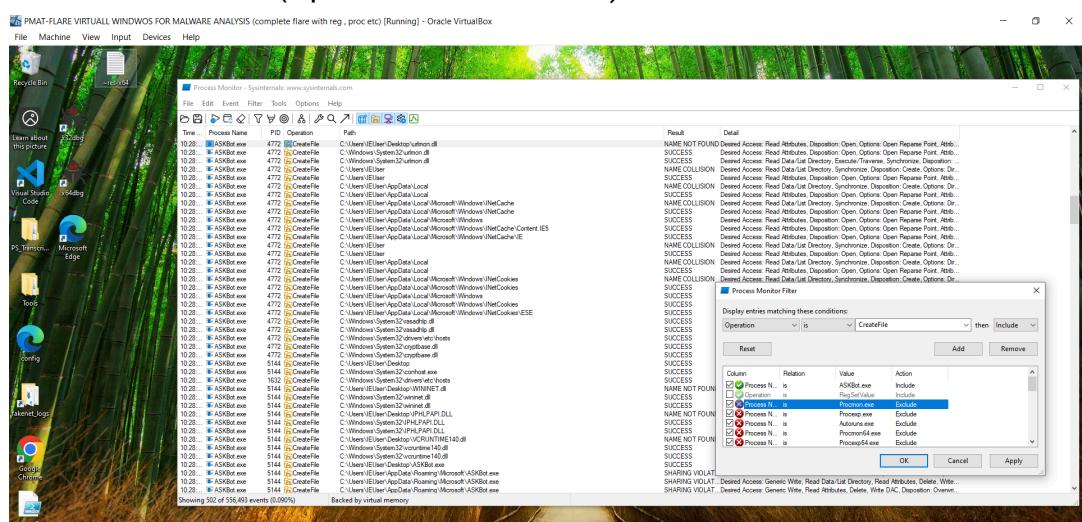


Figure 17

As you can see above **Figure 17** Filtering by **CreateFile** helps us **identify which files and locations the malware tries to access or manipulate**, including DLLs, config files, user data, or logs.

In screenshot, ASKBot.exe is attempting to access:

- System DLLs like **urmon.dll, cryptbase.dll, user32.dll**
- Internet cache directories like **INetCache, IE, Cookies**

This indicates that the malware is:

- **Loading dependencies dynamically**
- **Checking or modifying user-specific cache or cookie data, persistence**
- possibly creating **malicious file** and save into image

Procmon Filter 3: (Operation is RegSetValue)

below **Figure 18** Detects registry modifications – a key persistence method. Reveals changes to startup keys or malware configs. Helps identify how malware maintains control after reboot.

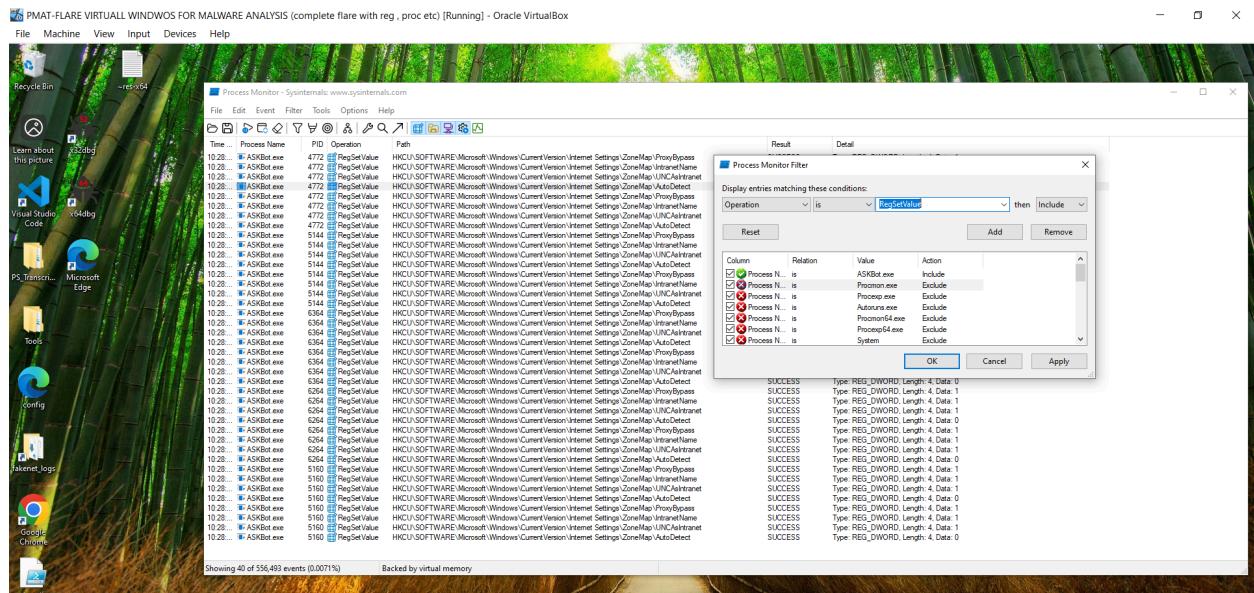


Figure 18

3.Task 3.2 – Debugging with x64dbg

Purpose of Debugger Analysis:

The purpose of using a debugger like x64dbg is to observe the real-time execution flow of the malware and understand its internal logic. It allows us to step through each instruction, analyze function calls, and monitor how the malware interacts with system components such as files, processes, and the registry.. Overall, debugger-based inspection provides deeper behavioral insights for reverse engineering and incident reporting.

Open in x64dbg

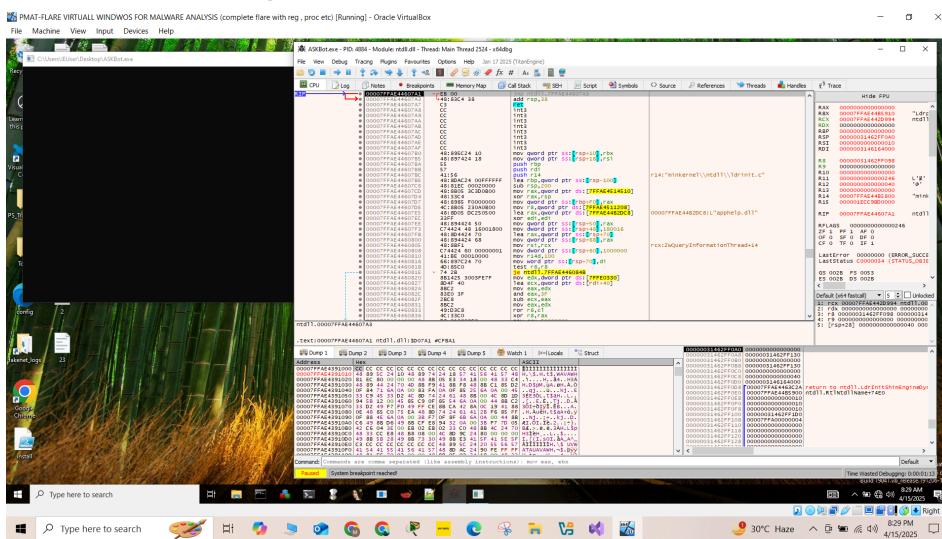


Figure 19

So after running and lot of the struggle i found the “entry point” was packed because there some reason

Checked in the x64dbg:



Figure 20

So in **Figure 20** when i open this sample the debugger automatically stop in this point and In this screenshot we can clearly see that program **jump** to the some function and then addition in the register then finally **(ret)**, so after analysing i see in the the particular address where the program want to jump.

Network indicators:

```

00007FF8ABCED09C CC int3
00007FF8ABCED0D0 CC int3
00007FF8ABCED0E0 CC int3
00007FF8ABCED0F0 CC int3
00007FF8ABCED0F4 48:89C4 mov rax,rsp
00007FF8ABCED0F8 48:8958 08 mov qword ptr ds:[rax+8],rbx
00007FF8ABCED0F9 48:8950 10 mov qword ptr ds:[rax+10],rdp
00007FF8ABCED0FB 48:8970 18 push rdi
Breakpoint Not Set 57
00007FF8ABCED0F9 48:89C4 00 sub rso,80

00007FF8ABC95F6D CC int3
00007FF8ABC95F8E CC int3
00007FF8ABC95F9F CC int3
00007FF8ABC95900 EC:88DC mov r11,rsp
00007FF8ABC95903 55 push rbp
00007FF8ABC95905 53 push r13
00007FF8ABC95905 41:56 push r14
00007FF8ABC95907 48:89C4 90 lea rbp,qword ptr ss:[rsp-70]
00007FF8ABC95907 48:89C4 90 sub rbp,qword ptr ss:[rsp-70]

```

Figure 21

Above **Figure 21** These functions belong to the WinINet library, commonly used by malware for establishing connections to remote servers, sending **HTTP** requests. These api function related to **WinINet API calls** confirms that **ASKBot.exe** exhibits behavior consistent with **network-aware malware**

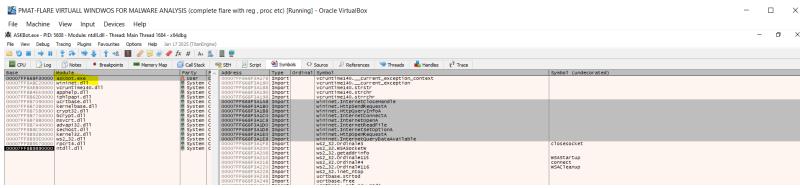


Figure 22

We should further see the **WinINet** for more network base call

WinINet

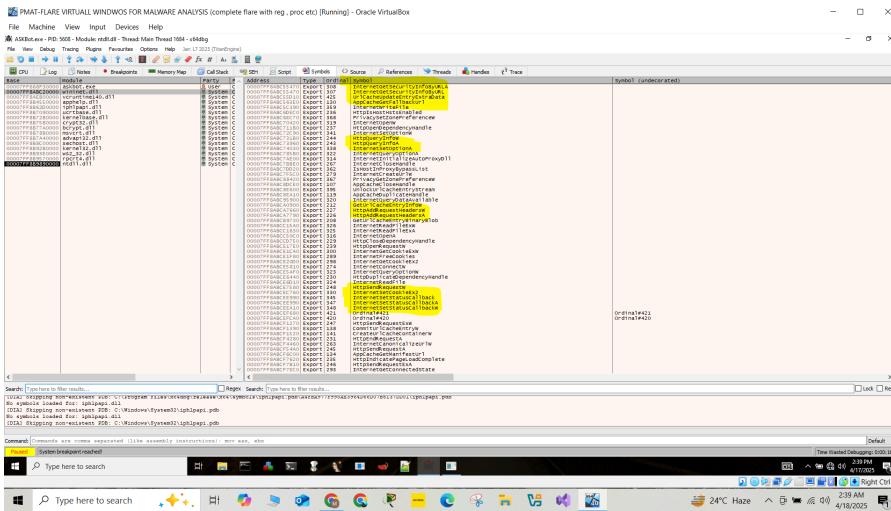


Figure 23

Like as you seen in **Figure 23** this we can clearly see in this that temptation of the connect for internet happend

presence of these WinINet API calls confirms that **ASKBot.exe** exhibits behavior consistent with network-aware malware such as:

Registry keys modification found :

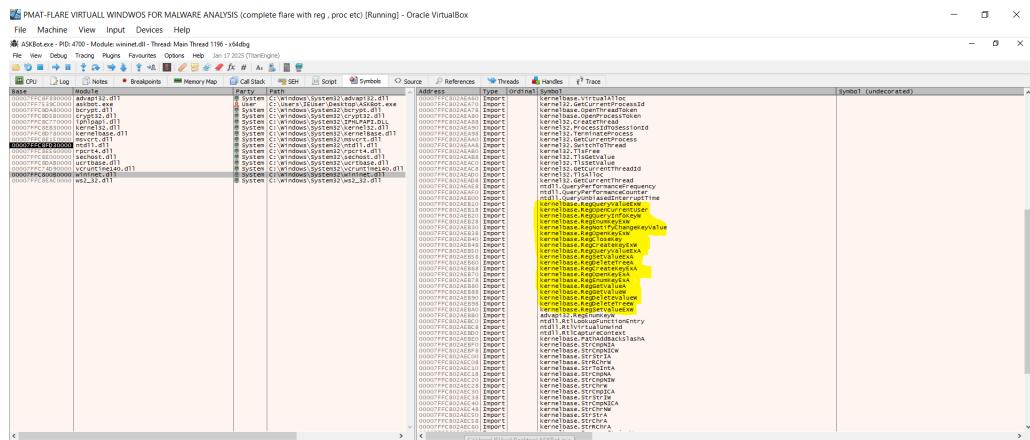


Figure 24