UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI

Information and Communication Technology Department

Final Report Part 2.1

Malware Analyst

Student Name: Đào Ngọc Tùng

Student ID: BA12-185

Ha Noi, 12 December 2024

Table Of Content

Basic Static Analysis

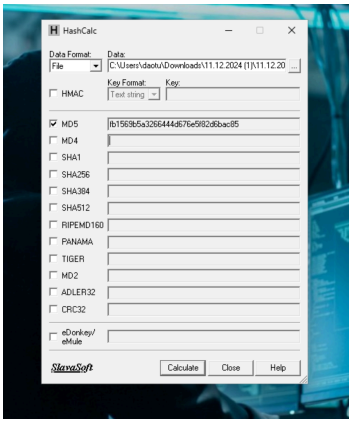
Purpose

The idea here is to glean any information from the file itself. This information is useful because it gives us an idea of what to expect when we run the executable, and may give us enough information to tweak our tools to get more out of the following basic dynamic analysis.

File information

File:part2.exe

MD5: fb1569b5a3266444d676e5f82d6bac85. Using HashClac to find MD5



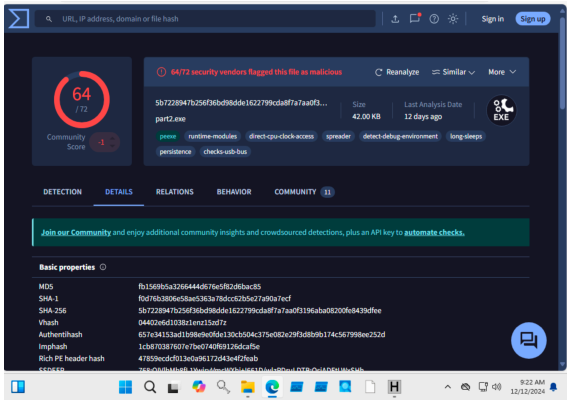
Size File: 42KB.

VirusTotal

Detection ratio: 64/72

Packers Identified: Ofuscation tools

Creation Time: 2004-01-27 11:22:58 UTC



Interesting Sections:

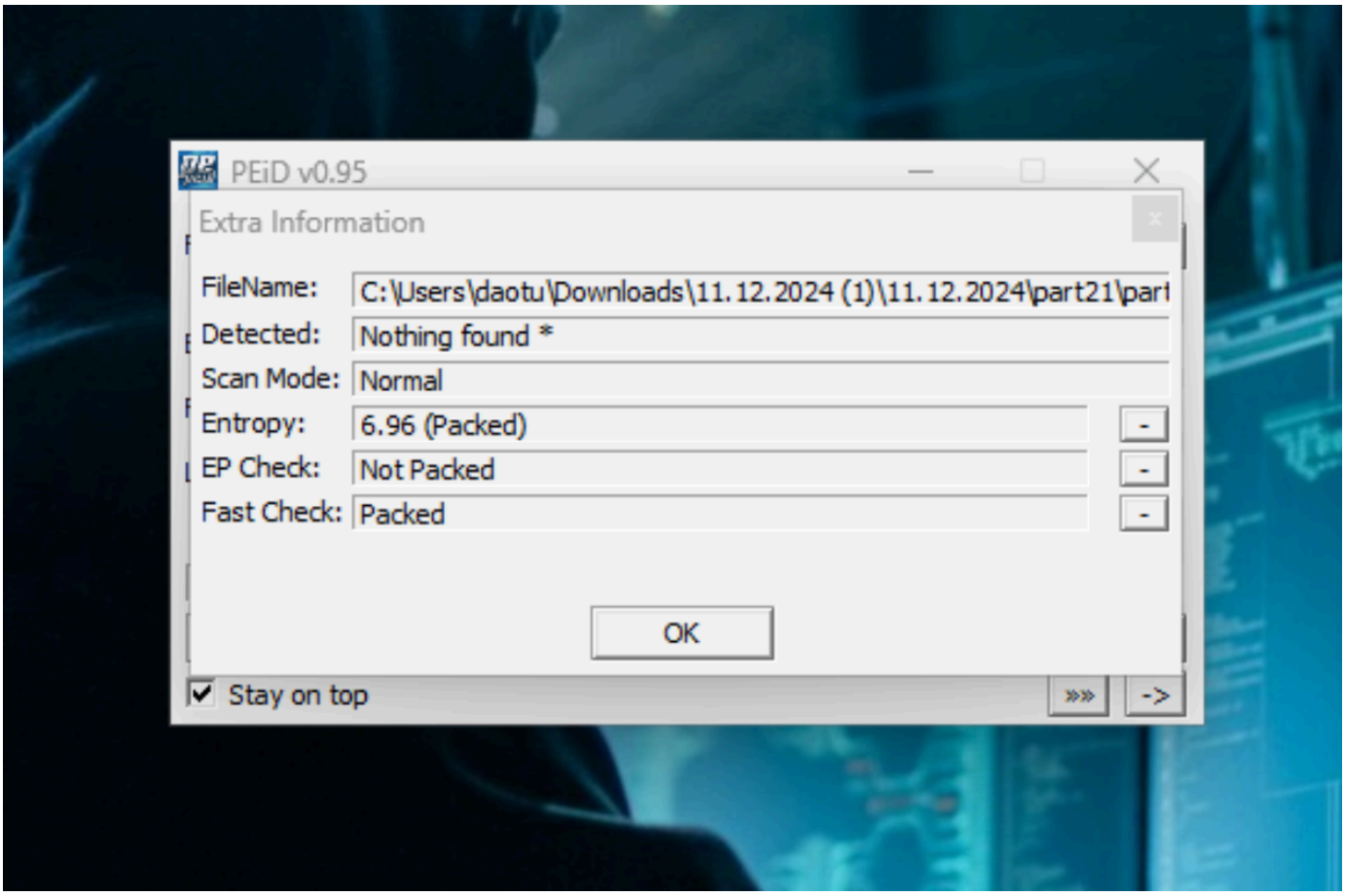
Sections

Name	Virtual Address	Virtual Size	Raw Size	Entropy	MD5	Chi2
.text	4096	41150	41472	6.91	4805651edeb048c0a10119c85da40482	541927.69
.data	49152	164	512	3.18	6931df9319eb6f5553c9c3791451bc66	61520

Figure 1: Section information from VirusTotal

PEiD

PEiD shows the entropy packed

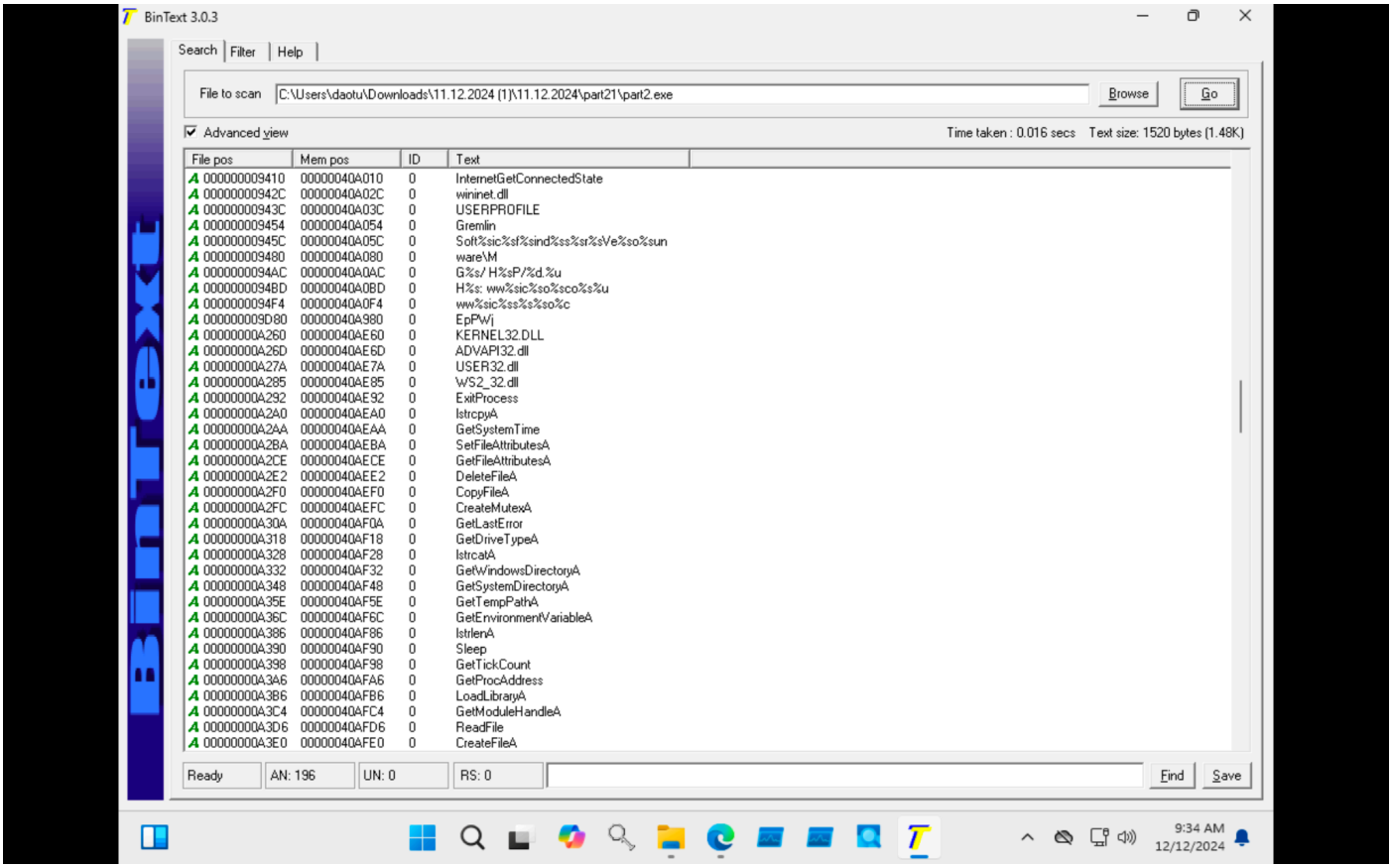


Dependency Walker

KERNEL32.DLL
ADVAPI32.DLL
SHELL32.DLL
WS2_32.DLL

A 000000009D80	00000040A980	0	EpPWj
A 00000000A260	00000040AE60	0	KERNEL32.DLL
A 00000000A26D	00000040AE6D	0	ADVAPI32.dll
A 00000000A27A	00000040AE7A	0	USER32.dll
A 00000000A285	00000040AE85	0	WS2_32.dll
A 00000000A292	00000040AE92	0	ExitProcess
A 00000000A2A0	00000040AEA0	0	lstrcpyA
A 00000000A2AA	00000040AEAA	0	GetSystemTime

Bintext



Some function-like names that aren't in the imports list.

+)KERNEL32.DLL

- ExitProcess
- lstrcpyA
- GetSystemTime
- SetFileAttributesA
- GetFileAttributesA
- DeleteFileA
- CopyFileA
- CreateMutexA
- GetLastError
- GetDriveTypeA
- lstrcatA
- GetWindowsDirectoryA
- GetSystemDirectoryA
- GetTempPathA
- GetEnvironmentVariableA
- lstrlenA
- Sleep
- GetTickCount
- GetProcAddress

LoadLibraryA
GetModuleHandleA
ReadFile
CreateFileA
GetModuleFileNameA
SetThreadPriority
GetCurrentThread
ExitThread
CreateThread
CloseHandle

+)ADVAPI32.DLL

GetUserNameA
RegCloseKey
RegOpenKeyA
RegSetValueExA

+)SHELL32.DLL

SHGetFileInfoA

+)WS2_32.DLL

closesocket
connect
gethostbyname
gethostname
htons
inet_addr
inet_ntoa
recv
send
Socket

Conclusion

The Total virus report has enough basic information for static analysis because the information is not packed.

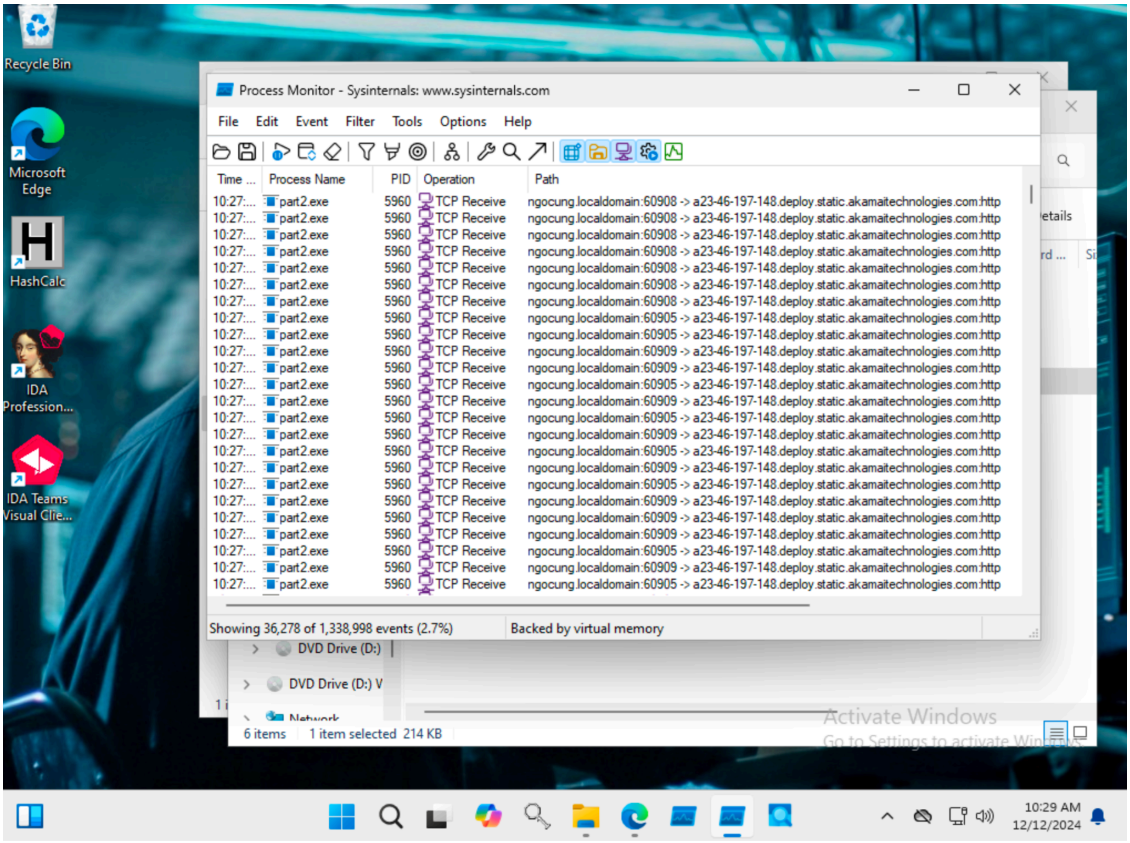
Basic Dynamic Analysis

Purpose

By running the program and analyzing its effects on the system we can get a first-hand look of the malware in action. This gives us some context when we disassemble the executable during advanced static analysis. It also allows us to develop host or network based signatures to identify the presence of this malware on a system in the future.

ProcessMonitor

Access crypto libs a lot, probably for the unpacking procedure.

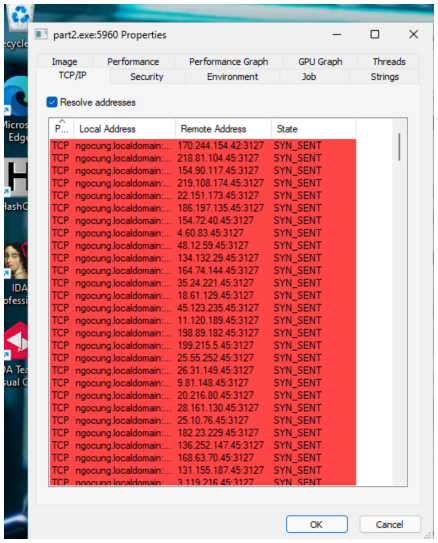


The file contains **4 types of operations (`Operation`)**. Here's the list of operation types and their general purposes:

1. TCP Receive
   - Description: Represents the activity of receiving data over a TCP network connection.
   - Purpose: Used to gather data sent from a server or another device through a TCP link.
2. TCP Connect
   - Description: Represents the establishment of a new TCP connection between two devices (e.g., client and server).
   - Purpose: Allows a process to connect to a server or another service over the network.
3. TCP Send
   - Description: Represents sending data over a TCP connection to the network.
   - Purpose: Facilitates data transmission from the current process to a server or another device.
4. TCP Reconnect
   - Description: Represents re-establishing a TCP connection, usually after a previous connection was interrupted.
   - Purpose: Maintains communication with a server or another device following a lost connection.

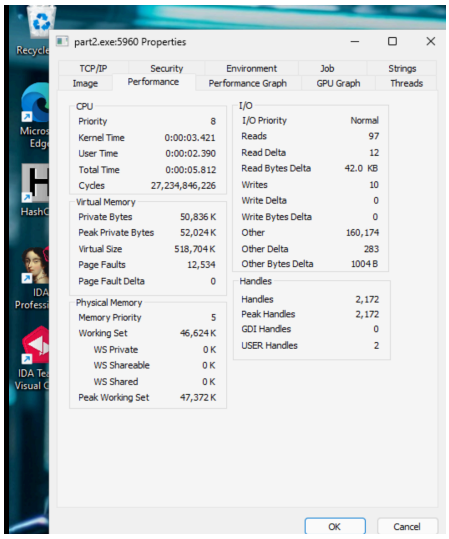
ProcessExplorer

1. TCP/IP connections originating from `ngocung.localdomain` to various remote addresses on port 3127. All connections are in the `SYN_SENT` state, indicating that the system is attempting to establish TCP connections but has not yet received responses. This behavior could indicate network scanning, botnet activity, or malware attempting to contact remote servers. Further investigation is recommended.



2. Performance for the process, detailing resource usage. Key points include:
   - CPU: Low usage with a total time of 5.8 seconds and priority 8.
   - Memory: Moderate use of virtual memory (518,704 K virtual size) with 46,624 K in the working set.
   - I/O: 160,174 "Other" I/O operations suggest unusual activity.
   - Handles: 2,172 handles, which is relatively high.

This behavior could indicate potentially abnormal or resource-intensive activity. Further analysis or sandbox testing is recommended.



Conclusion

The file contains seven operations: TCP Receive, TCP Connect, TCP Send, Thread Create, Thread Exit, ReadFile, and TCP Reconnect. Most involve network communication (*Receive, Send, Connect, Reconnect*), while others manage threading (*Create, Exit*) or file access (*ReadFile*). Network operations dominate, indicating a system focused on data transmission and connectivity.

Advanced Static Analysis

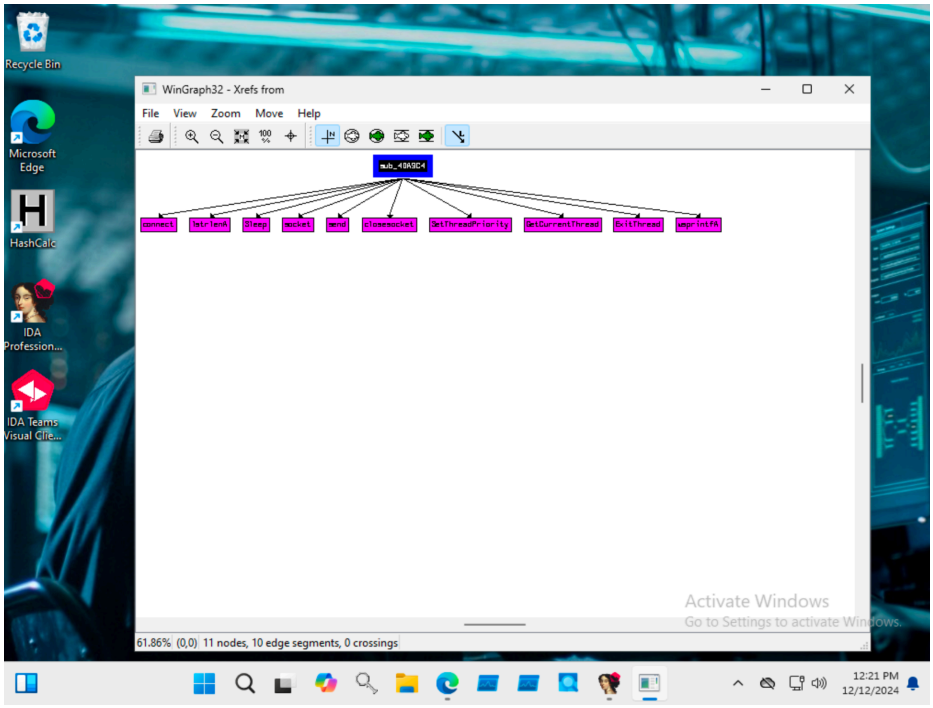
Purpose

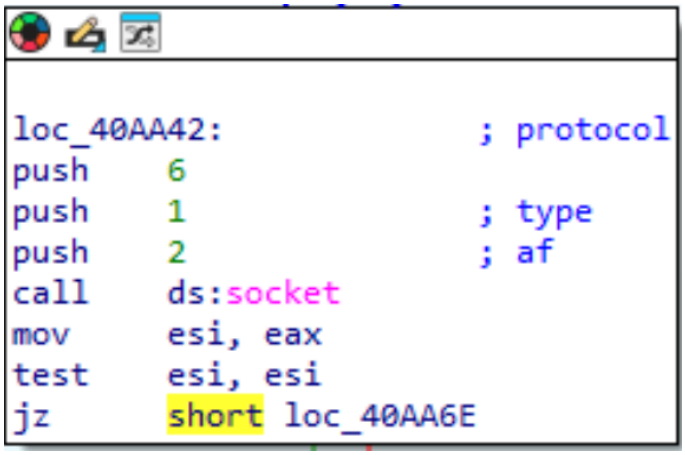
The purpose is to analyze malware functions, control flow analysis, and generate function and variable names.

IDA

+) Let's start with the *socket* function(0x0040AA42). First at the assembly code. If we review this function from Microsoft's website, we see it takes 3 parameters, which are all integers. These parameters are *af, stream socket,* and *protocol*.

Graph:



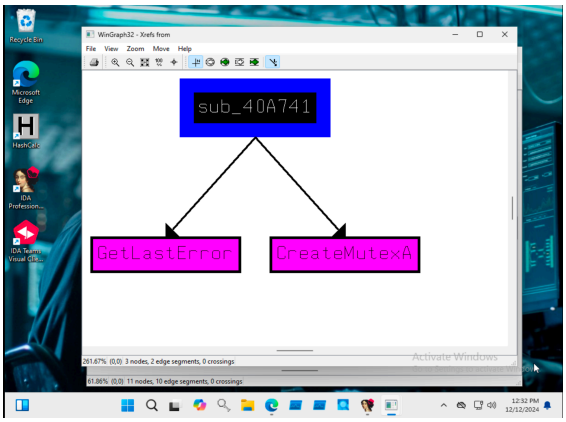


The code is creating a socket using the socket() function with the parameters:

- AF_INET (IPv4 address family),
- SOCK_STREAM (stream socket for TCP),
- IPPROTO_TCP (TCP protocol).

If the socket() call fails (returning -1 or 0), it redirects the flow to a failure-handling routine (loc 40AAGE).

+)CreateMutexA (0x0040A78B): A file is XOR-encrypted using the key 0x55 and uploaded to a distant server via a socket using the sub_402210 function (already examined).




```
var_35= byte ptr -35h
var_34= byte ptr -34h
var_33= byte ptr -33h
var_32= byte ptr -32h

push    ebp
mov     ebp, esp
sub     esp, 40h
and     [ebp+var_32], 0
lea     eax, [ebp+Name]
push    eax             ; lpName
push    1               ; bInitialOwner
push    0               ; lpMutexAttributes
mov     [ebp+Name], 73h ; 's'
mov     [ebp+var_3F], 79h ; 'y'
mov     [ebp+var_3E], 6Eh ; 'n'
mov     [ebp+var_3D], 63h ; 'c'
mov     [ebp+var_3C], 2Dh ; '-'
mov     [ebp+var_3B], 5Ah ; 'Z'
mov     [ebp+var_3A], 2Dh ; '-'
mov     [ebp+var_39], 6Dh ; 'm'
mov     [ebp+var_38], 74h ; 't'
mov     [ebp+var_37], 78h ; 'x'
mov     [ebp+var_36], 5Fh ; '_'
mov     [ebp+var_35], 31h ; '1'
mov     [ebp+var_34], 33h ; '3'
mov     [ebp+var_33], 33h ; '3'
call    ds:CreateMutexA
call    ds:GetLastError
sub     eax, 0B7h
neg     eax
sbb     eax, eax
inc     eax
leave
retn
sub_40A741 endp
```

This code performs the following steps:

- Initialize Mutex Name: It constructs the mutex name string "sync-Z-mtx_133" using ASCII values assigned to local variables.
- Create Mutex: The code calls CreateMutexA to create a mutex with the initialized name, granting the calling thread initial ownership.
- Error Checking: It calls GetLastError to check if there was an error during mutex creation and performs some calculations to handle the error code.
- Function Exit: The function cleans up the stack and exits.

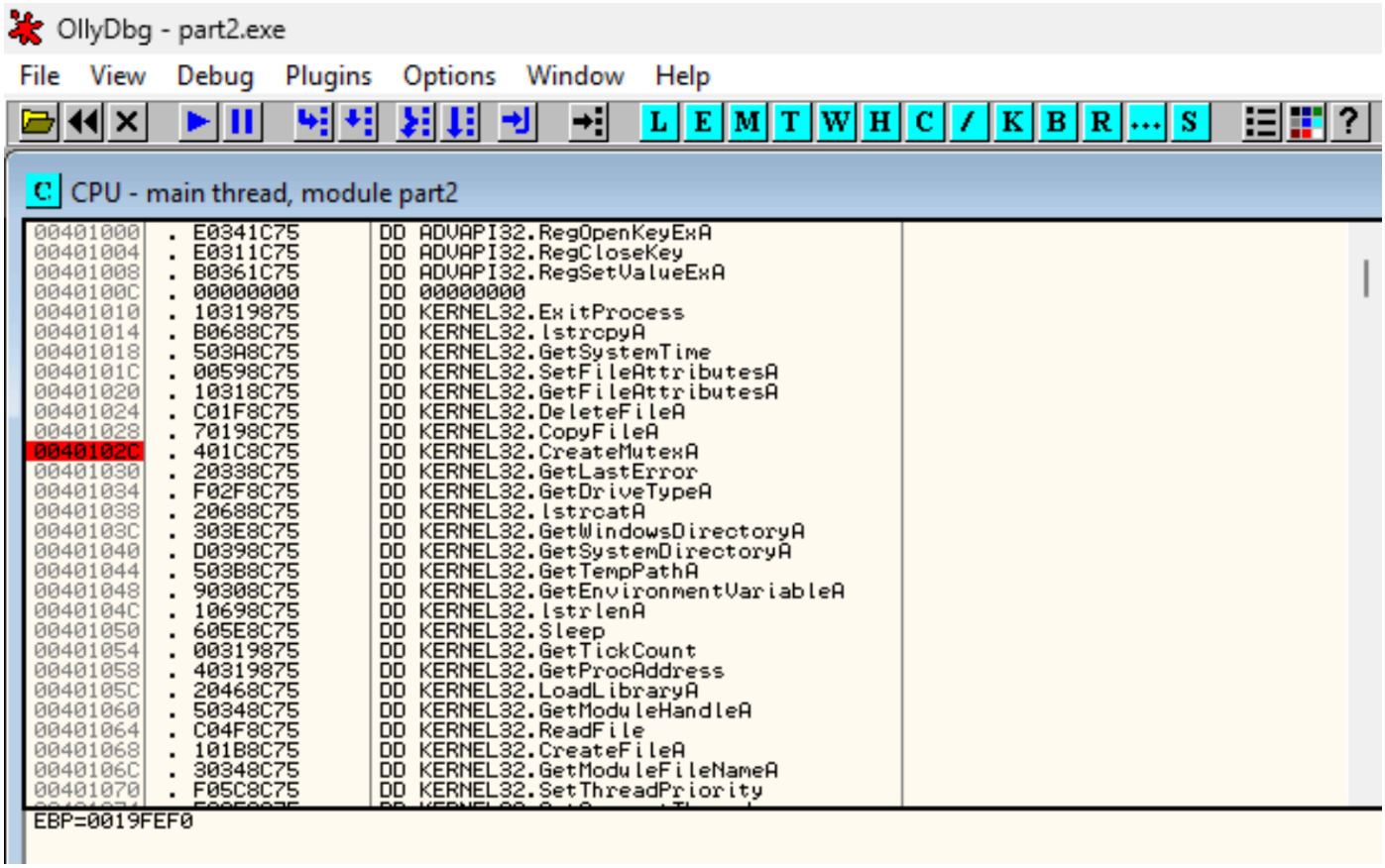
Conclusion

Create socket function to communication between two endpoints in networks is a Windows API function used to create or open a mutex

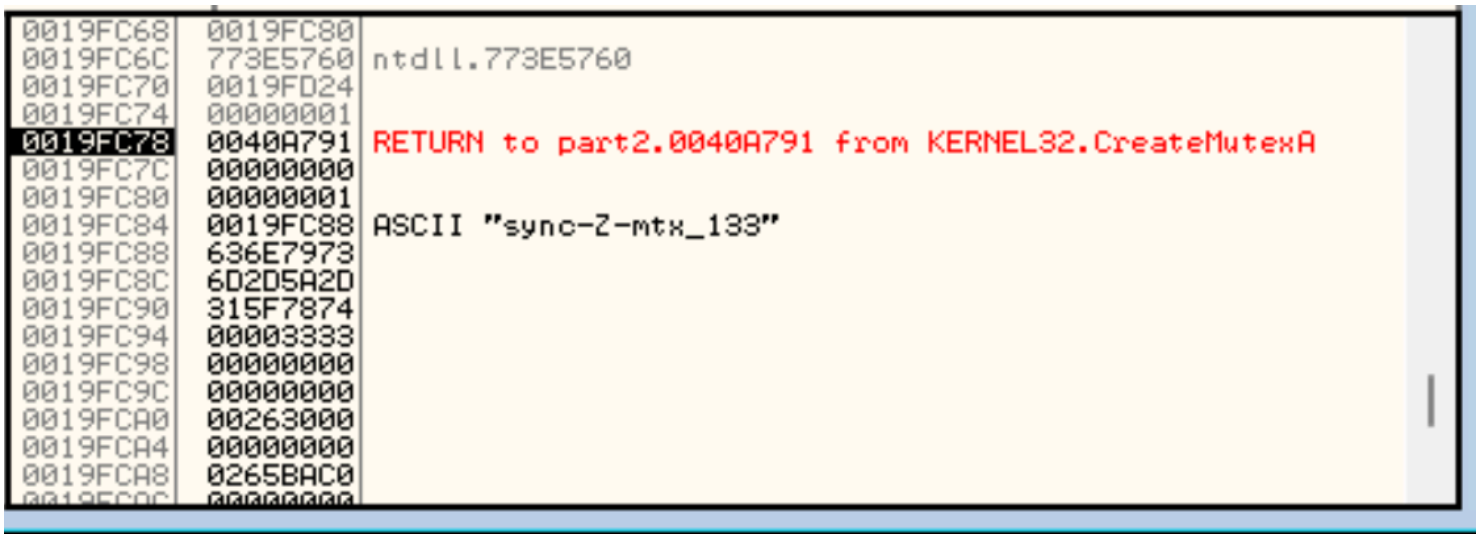
Advanced Dynamic Analysis

OllyDbg

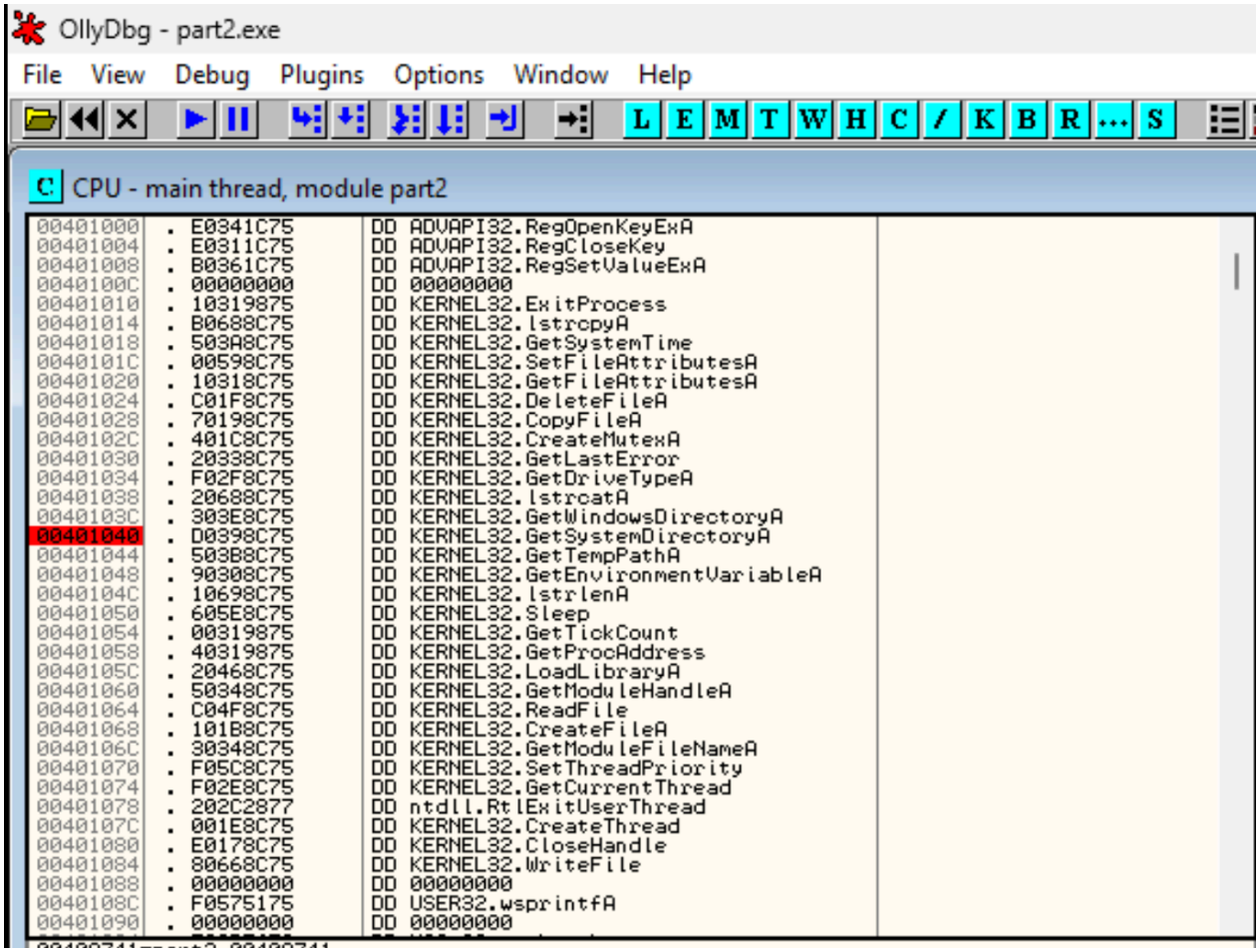
Step1: The first breakpoint to be hit was *CreateMutexA*



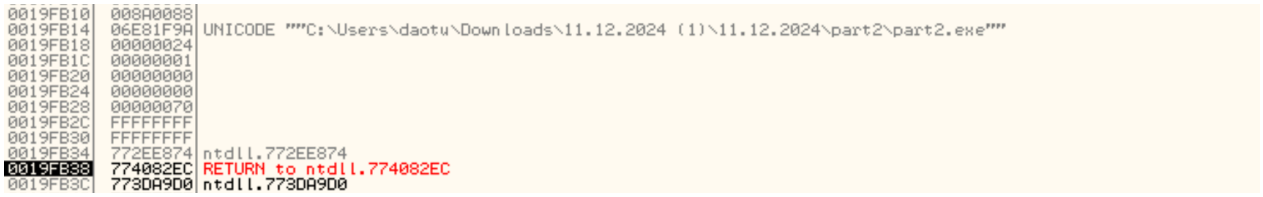
Step2: Run Debug and we can Looking at the output within ollydbg, we see the Mutex created with the name *sync-Z-mtx_133*.



Step3: Continuing to run the program, the next breakpoint was *GetSystemDirectoryA*.



Step4: Output returned: *C:\Users\daotu\Downloads\11.12.2024\part2\part2.exe*



Conclusion

After the analysis advanced dynamics, we can see The CreateMutexA function creates or opens a mutex to synchronize resource access between threads or processes. It ensures that only one thread accesses a shared resource at a time, preventing conflicts in multi-threaded environments. The GetSystemDirectoryA function retrieves the path of the Windows system directory, typically used to access system files.