Faculty of Engineering, Computing and Science

COS20030 Malware Analysis

Assignment 1 (20 marks)

Reminder:
All malware analysis tasks must be performed in a VM.
Do not extract the malware sample zip file directly on the host machine.
Bring it into the VM then extract it.

General Instructions

1. This is an individual assignment.
2. This assignment allows you to demonstrate essential skills for malware analysis.
3. If you refer to any online websites and articles, you must include proper citation and referencing. See Swinburne Library guide to APA style referencing.
4. Before submitting the report, please ensure that you have checked Canvas for any announcements and updates related to the assignment.
5. You can submit in Canvas several times before the deadline; each new submission will overwrite your previous submission.
6. Please check the submission deadline in Canvas. Any submissions after the deadline will be penalised based on policy outlined in the unit outline unless permission has been granted.
7. If you are unable to meet the submission deadline due to exceptional circumstances, please contact the unit convenor before the deadline. Extension may be granted for special cases after careful consideration by the unit convenor.

   Note: Reasons such as "busy with assignments from other units" or "many due dates happening at the same time" will not be considered.
8. **What to submit in Canvas:**
   i) Assignment report in PDF format
   ii) ZIP file containing modified executable (required by Question 2)

Plagiarism Warning

If plagiarism is detected in your submitted work, you will be subjected to serious penalties.

Acts of plagiarism includes:

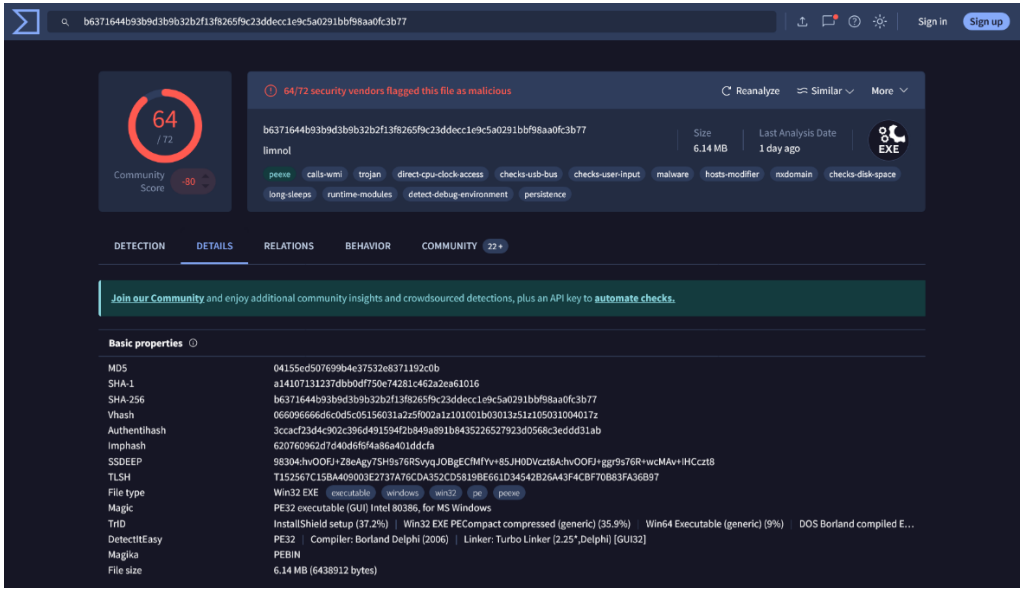
- sharing your original work with other students either on purpose or by accident – under no circumstances should you show or give your full/partial assignment to another student, nor should you ask to see another student's assignment;
- soliciting answers from online forums and tutoring sites (even if it does not involve any form of money payment);
- copying answers publicly posted online.

Please review the Declaration and Statement of Authorship on Canvas submission page. Electronic submission of your project report signifies that you agree with this declaration.

Question 1 (12 marks)

As a malware analyst, you are given a suspicious file named **sample_q1.exe** to investigate. Using what you have learned in the weekly labs, you are required to perform **basic static analysis** and **basic dynamic analysis** on the given file.

Document your findings in the table below. Any screenshots should be accompanied with appropriate caption or description.

Filename: sample_q1.exe		
Criteria	**Description/explanation**	**Tool used**
Indicators that the file is packed	Based on VirusTotal analysis, the sample_q1.exe appears to be **packed with the PECompact packer**, as indicated by the "PECompact compressed" file type description. Additionally, the .text, .rdata, and .data sections show high entropy values (around 6.5–6.6), suggesting compressed or encrypted content. Several antivirus detections label the file as a "packed" or "Simda" variant, confirming that the file is likely obfuscated to conceal its true behavior.  *Figure 1: VirusTotal analysis of sample_q1.exe showing a high number of antivirus detections and the file type identified as "PECompact compressed," indicating the file is packed.*	VirusTotal

DETECTION DETAILS RELATIONS BEHAVIOR COMMUNITY 22+

Join our Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Popular threat label trojan.simda/basine | Threat categories trojan dropper fakeav | Family labels simda basine rogue

Security vendors' analysis — Do you want to automate checks?

AhnLab-V3	Trojan/Win32.FakeAV.R76231	Alibaba	Backdoor:Win32/Simda.5f4f623b
AliCloud	Backdoor:Win/AgentIVK.LDG	ALYac	Gen:Variant.Delf.48
Antiy-AVL	Trojan[Backdoor]/Win32.Simda	Arcabit	Trojan.Delf.48
Arctic Wolf	Unsafe	Avast	Win32:Simda-EG [Trj]
AVG	Win32:Simda-EG [Trj]	Avira (no cloud)	TR/Simda.igtnk
BitDefender	Gen:Variant.Delf.48	Bkav Pro	W32.AIDetectMalware
ClamAV	Win.Trojan.7157609-1	CrowdStrike Falcon	Win/malicious_confidence_100% (W)
CTX	Exe.trojan.simda	Cynet	Malicious (score: 100)
DeepInstinct	MALICIOUS	DrWeb	Trojan.Hosts.35050
Elastic	Malicious (high Confidence)	Emsisoft	Gen:Variant.Delf.48 (B)
eScan	Gen:Variant.Delf.48	ESET-NOD32	A Variant Of Win32/RogueAV.I
Fortinet	W32/Small.HCZ!tr	GData	Gen:Variant.Delf.48
Google	Detected	GridinSoft (no cloud)	Trojan.Win32.Gen.bot!i
Huorong	Trojan/Delf.bi	Ikarus	Trojan.Win32.FakeAV
Jiangmin	TrojanDropper.Small.fkk	K7AntiVirus	Trojan (0001140e1)
K7GW	Trojan (0001140e1)	Kaspersky	Trojan-Dropper.Win32.Small.hcz
Kingsoft	Win32.Troj.Undef.a	Lionic	Trojan.Win32.Simda.G!c
Malwarebytes	Malware.AI.2645883817	MaxSecure	Trojan.Malware.2823018.susgen
McAfee Scanner	Tilb63716448B3B	Microsoft	Backdoor:Win32/Simda.AT
NANO-Antivirus	Trojan.Win64.Agent.kftri	Palo Alto Networks	Generic.ml
Panda	Trj/CI.A	QuickHeal	Trojan.Ghanarava.1732726989192c0b
Rising	Backdoor.Simda!8.2D9 (TFE:1:Y36MWhX:...	Sangfor Engine Zero	Backdoor.Win32.Simda.Vyw5
SecureAge	Malicious	SentinelOne (Static ML)	Static AI - Suspicious PE
Skyhigh (SWG)	BehavesLike.Win32.Dropper.vh	Sophos	Mal/Basine-C
SUPERAntiSpyware	Rogue.MySecurityShield	Symantec	ML.Attribute.HighConfidence
Tencent	Win32.Trojan-Dropper.Small.Eplw	Trellix ENS	Generic.FakeAV.cz
TrendMicro	TROJ_FRS.0NA103GD23	TrendMicro-HouseCall	TROJ_FRS.0NA103GD23
Varist	W32/Swisyn.C.gen!Eldorado	VBA32	BScope.TrojanRansom.CryFile
VIPRE	Gen:Variant.Delf.48	ViriT	Trojan.Win32.Genus.KUO
Webroot	W32.Rogue.Gen	WithSecure	Heuristic.HEUR/AGEN.1364260
Xcitium	Malware@#207ednybfcknw	Yandex	Trojan.GenAsa!Pwz4HpMoFuE
Zillya	Trojan.RogueAV.Win32.40	ZoneAlarm by Check Point	Mal/Basine-C
Acronis (Static ML)	Undetected	Baidu	Undetected
CMC	Undetected	TACHYON	Undetected
TEHTRIS	Undetected	Trapmine	Undetected
ViRobot	Undetected	Zoner	Undetected
Avast-Mobile	Unable to process file type	BitDefenderFalx	Unable to process file type
Symantec Mobile Insight	Unable to process file type	Trustlook	Unable to process file type

Figure 2: VirusTotal analysis report for sample_q1.exe, showing detection by 64/72 antivirus engines with popular threat labels including Trojan.Win32.FakeAV.R763231 and Backdoor.Win32.Simda.5476321b, alongside various family labels and threat categories, indicating potential packing and malicious behavior.

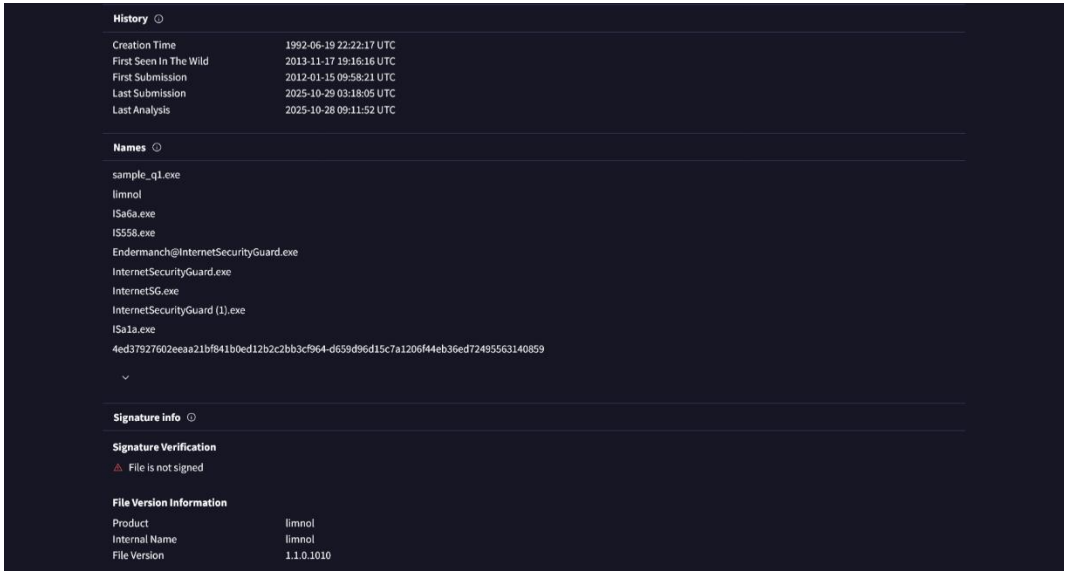


Figure 3: VirusTotal analysis report for sample_q1.exe, displaying the history section with creation time (1992-06-19 22:21:17 UTC), first seen in the wild (2013-11-17 19:08:16 UTC), and last submission (2025-10-28 09:18:52 UTC), along with file names (e.g., sample_q1.exe, ISsa6a.exe) and signature information indicating the file is not signed, with a product version of 1.1.10.10.

Portable Executable Info

Header

Target Machine	Intel 386 or later processors and compatible processors
Compilation Timestamp	1992-06-19 22:22:17 UTC
Entry Point	2519156
Contained Sections	9

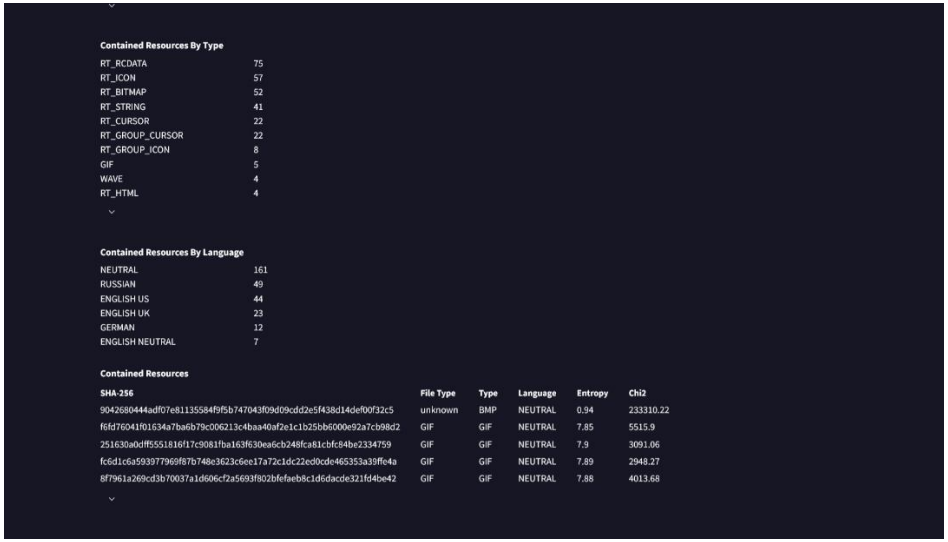
Sections

Name	Virtual Address	Virtual Size	Raw Size	Entropy	MD5	Chi2
.text	4096	2504832	2504832	6.54	5fdc7723fcd8603c386612cd0402b312	15516487
.itext	2510848	8732	8732	6.6	19ecc8c7625f37c6ca76cda6d09040eb	41931.42
.data	2523136	35828	35828	6.12	7bf9cf461b923184d4c4d3de95c67bc1	769088.69
.bss	2560000	24760	24760	0	ae340132cda20796c752b46953d9558e	6313744
.idata	2588672	20192	20192	5.72	7f5f51b2cc253821cbd6b34c936d2e01	287798.56

Imports

+ oleaut32.dll
+ advapi32.dll
+ user32.dll
+ kernel32.dll
+ gdi32.dll
+ version.dll
+ ole32.dll
+ comctl32.dll
+ URLMON.DLL
+ wininet.dll

Figure 4: Portable Executable (PE) information for sample_q1.exe, showing target machine (Intel 386 or later processors and compatible processors), compilation timestamp (1992-06-19 22:21:17 UTC), entry point (251856), and details of 9 contained sections, including virtual address, virtual size, raw size, entropy, and MD5 hashes, along with imported DLLs such as oleaut32.dll and kernel32.dll.

	 **Contained Resources By Type** RT_RCDATA 75 RT_ICON 57 RT_BITMAP 52 RT_STRING 41 RT_CURSOR 22 RT_GROUP_CURSOR 22 RT_GROUP_ICON 8 GIF 5 WAVE 4 RT_HTML 4 **Contained Resources By Language** NEUTRAL 161 RUSSIAN 49 ENGLISH US 44 ENGLISH UK 23 GERMAN 12 ENGLISH NEUTRAL 7 **Contained Resources** SHA-256 / File Type / Type / Language / Entropy / Chi2 9042680444adf07e81135584f9f5b747043f09d09cdd2e5f438d14def00f32c5 unknown BMP NEUTRAL 0.94 233310.22 f6fd76041f01634a7ba6b79c006213c4baa40af2e1c1b25bb6000e92a7cb98d2 GIF GIF NEUTRAL 7.85 5515.9 251630a0dff5551816f17c9081fba163f630ea6cb248fca81cbfc84be2334759 GIF GIF NEUTRAL 7.9 3091.06 fc6d1c6a593977969f87b748e3623c6ee17a72c1dc22ed0cde465353a39ffe4a GIF GIF NEUTRAL 7.89 2948.27 8f7961a269cd3b70037a1d606cf2a5693f802bfefaeb8c1d6dacde321fd4be42 GIF GIF NEUTRAL 7.88 4013.68 *Figure 5: VirusTotal analysis report for sample_q1.exe, detailing contained resources by type (e.g., RT_RCDATA: 75, RT_ICON: 57) and language (e.g., Neutral: 361, Russian: 49), with file type information including GIF and BMP formats, SHA-256 hash, file sizes, and entropy values, indicating the presence of embedded resource data.*	
Library and function imports	Sample_q1.exe DLL imports& functions: 1. OLEAUT32.dll: SysFreeString, SysReAllocStringLen, SysAllocStringLen, CreateErrorInfo, GetErrorInfo, SetErrorInfo, GetActiveObject, RegisterTypeLib, LoadTypeLibEx, SysFreeString, SafeArrayPtrOfIndex, SafeArrayGetUBound, SafeArrayGetLBound, SafeArrayCreate, VariantChangeType, VariantCopy, VariantClear, VariantInit	Dependecy Walker

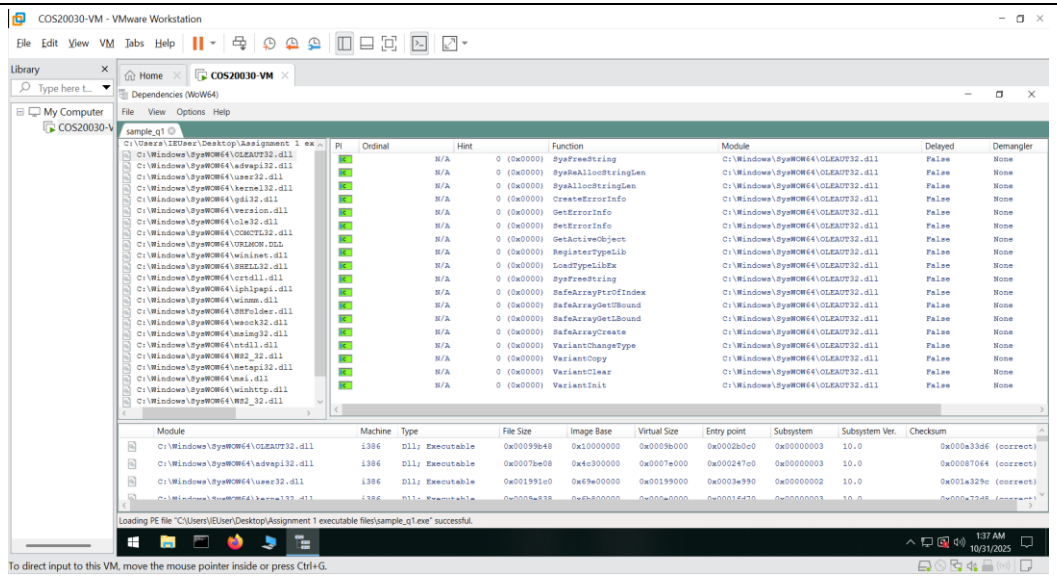


Figure 6: OLEAUT32.dll and its imported functions

2. advapi32.dll: RegQueryValueExA, RegOpenKeyExA, RegCloseKey, SetSecurityDescriptorDacl, SetFileSecurityA, RegSetValueExA, RegQueryValueExA, RegQueryInfoKeyA, RegOpenKeyExA, RegOpenKeyA, RegFlushKey, RegEnumValueA, RegEnumKeyExA, RegDeleteValueA, RegDeleteKeyA, RegCreateKeyExA, RegCloseKey, OpenThreadToken, OpenProcessToken, InitializeSecurityDescriptor, InitializeAcl, GetUserNameA, GetTokenInformation, FreeSid, EqualSid, AllocateAndInitializeSid, AddAccessAllowedAce, CryptDestroyHash, CryptHashData, CryptCreateHash, CryptGetHashParam, CryptReleaseContext, CryptAcquiredContextA, StartServiceA, QueryServiceStatus, QueryServiceConfigA, OpenServiceA, OpenSCManagerA, EnumServicesStatusA, ControlService, CloseServiceHandle, ChangeServiceConfigA, QueryServiceConfig2A, GetUserNameW

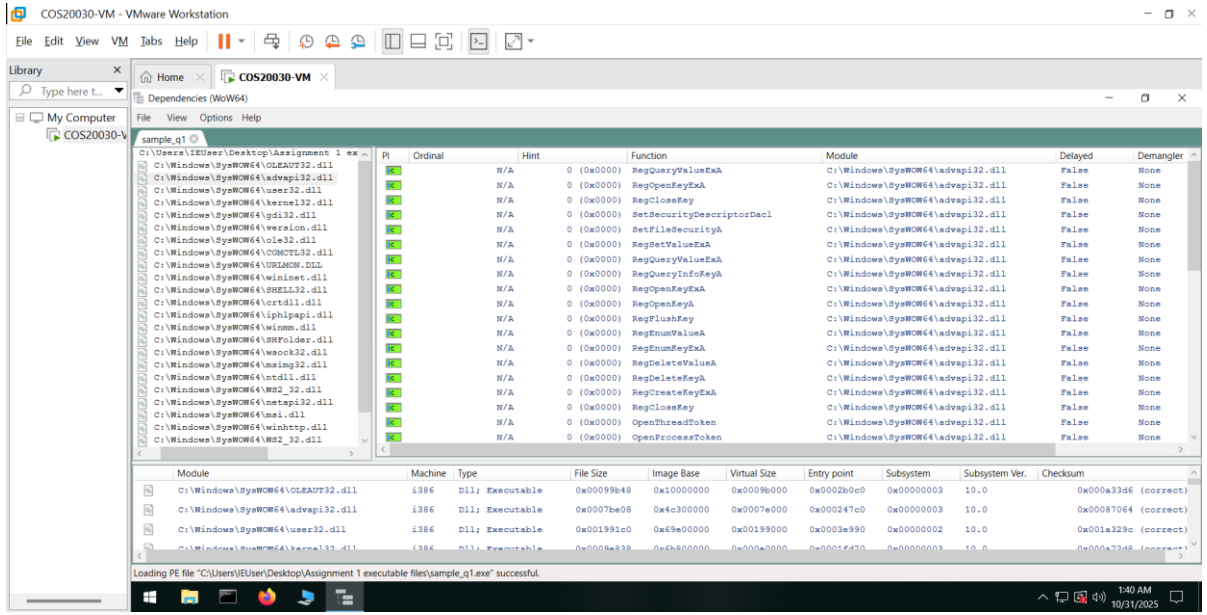


Figure 7: advapi32.dll and its imported functions part 1

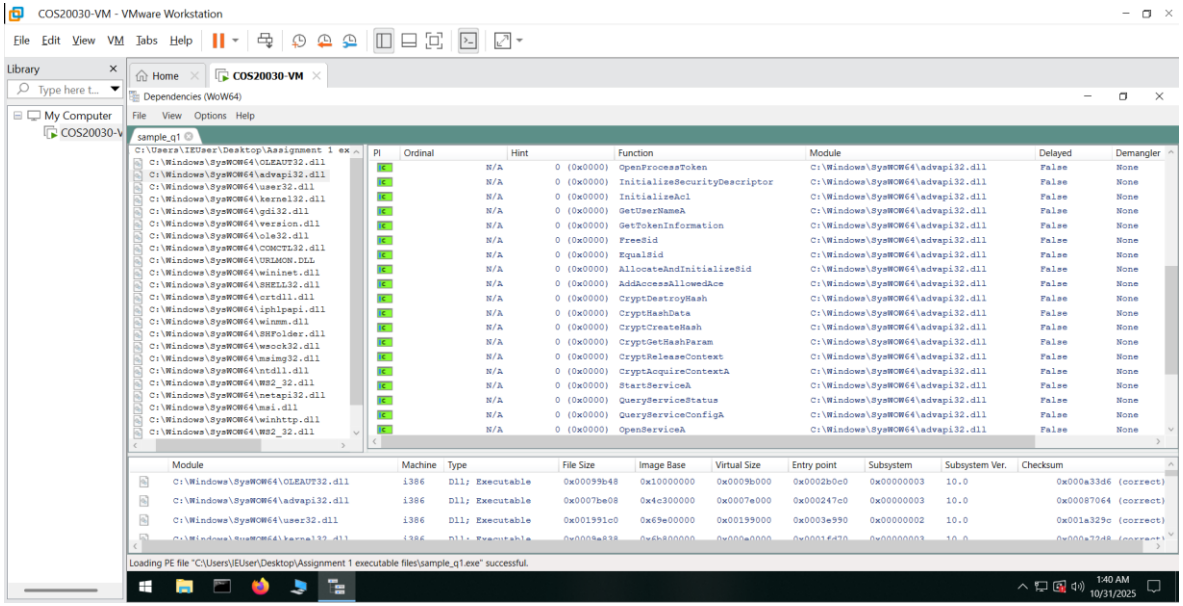


Figure 8: advapi32.dll and its imported functions part 2

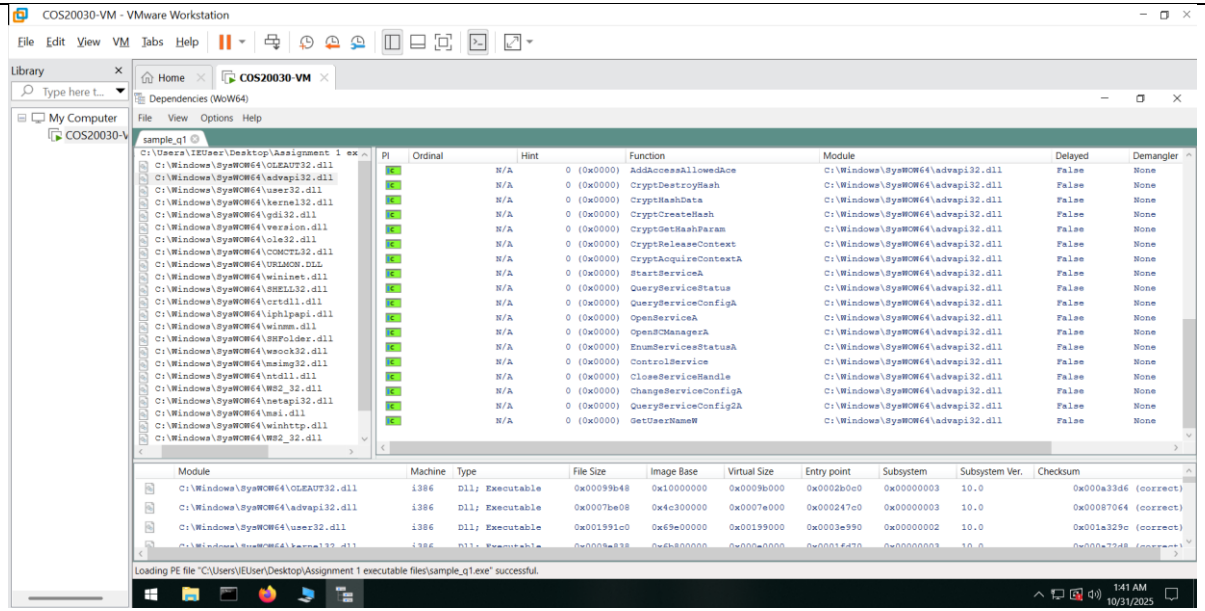


Figure 9: advapi32.dll and its imported functions part 3

3. user32.dll: GetKeyboardType, DestroyWindow, LoadStringA, MessageBoxA, CharNextA, CreateWindowExA, keybd_event, WindowFromPoint, WindowFromDC, WaitMessage, VkKeyScanA, ValidateRect, UpdateWindow, UnregisterClassA, UnionRect, UnhookWindowsHookEx, TranslateMessage, TranslateMDISysAccel, TrackPopupMenu, ToAscii, SystemParametersInfoW, SystemParametersInfoA, SubtractRect, ShowWindow, ShowScrollBar, ShowOwnedPopups, SetWindowRgn, SetWindowsHookExA, SetWindowTextA, SetWindowPos, SetWindowPlacement, SetWindowLongW, SetWindowLongA, SetTimer, SetSystemCursor, SetScrollRange, SetScrollPos, SetScrollInfo, SetRect, SetPropA, SetParent, SetMenuItemInfoA, SetMenu, SetForegroundWindow, SetFocus, SetCursor, SetClipboardData, SetClassLongA, SetCapture, SetActiveWindow, SendMessageW, SendMessageA, SendInput, ScrollWindow, ScrollDC, ScreenToClient, RemovePropA, RemoveMenu, ReleaseDC, ReleaseCapture, RegisterWindowMessageA, RegisterClipboardFormatA, RegisterClassA, RedrawWindow, PrInRect, PostThreadMessageA, PostQuitMessage, PostMessageA, PeekMessageW, PeekMessageA, OpenClipboard, OffsetRect, OemToCharA, MsgWaitForMultipleObjectsEx, MsgWaitForMultipleObjects, MoveWindow, MessageBoxA, MessageBeep, MapWindowPoints, MapVirtualKeyA, LockWindowUpdate, LoadStringA, LoadKeyboardLayoutA, LoadIconA, LoadCursorA, LoadBitmapA, KillTimer, IsZoomed, IsWindowVisible, IsWindowUnicode, IsWindowEnabled, IsWindow, IsRectEmpty, IsIconic, IsDialogMessageW, IsDialogMessageA, IsChild, InvalidateRect, IntersectRect, InsertMenuItemA,

	InsertMenuA, InflateRect, GetWindowThreadProcessId, GetWindowTextLengthW, GetWindowTextW, GetWindowTextA, GetWindowRect, GetWindowPlacement, GetWindowLongW, GetWindowLongA, GetWindowDC, GetUpdateRect, GetTopWindow, GetSystemMetrics, GetSystemMenu, GetSysColorBrush, GetSysColor, GetSubMenu, GetScrollRange, GetScrollPos, GetScrollInfo, GetScrollBarInfo, GetPropA, GetParent, GetWindow, GetMessageTime, GetMessagePos, GetMenuStringA, GetMenuState, GetMenuItemRect, GetMenuItemInfoA, GetMenuItemID, GetMenuItemCount, GetMenu, GetLastActivePopup, GetKeyboardState, GetKeyboardLayoutNameA, GetKeyboardLayoutList, GetKeyboardLayout, GetKeyState, GetKeyNameTextA, GetIconInfo, GetForegroundWindow, GetFocus, GetDlgCtrlID, GetDesktopWindow, GetDCEx, GetDC, GetCursorPos, GetCursor, GetClipboardFormatNameA, GetClipboardData, GetClientRect, GetClassNameA, GetClassLongA, GetClassInfoA, GetCapture, GetActiveWindow, FrameRect, FindWindowExA, FindWindowA, FillRect, EqualRect, EnumWindows, EnumThreadWindows, EnumClipboardFormats, EnumChildWindows, EndPaint, EndDeferWindowPos, EnableWindow, EnableScrollBar, EnableMenuItem, EmptyClipboard, DrawTextW, DrawTextA, DrawMenuBar, DrawIconEx, DrawIcon, DrawFrameControl, DrawFocusRect, DrawEdge, DispatchMessageW, DispatchMessageA, DestroyWindow, DestroyMenu, DestroyIcon, DestroyCursor, DeleteMenu, DeferWindowPos, DefWindowProcA, DefMDIChildProcA, DefFrameProcA, CreatePopupMenu, CreateMenu, CreateIcon, CopyRect, CopyImage, CopyIcon, CloseClipboard, ClientToScreen, ChildWindowFromPoint, CheckMenuItem, CharNextW, CallWindowProcA, CallNextHookEx, BringWindowToTop, BeginPaint, BeginDeferWindowPos, AttachThreadInput, CharNextA, CharLowerBuffA, CharLowerA, CharUpperBuffA, CharUpperA, CharToOemA, AdjustWindowRectEx, ActivateKeyboardLayout, DdeCmpStringHandles, DdeFreeStringHandle, DdeQueryStringA, DdeCreateStringHandleA, DdeGetLastError, DdeFreeDataHandle, DdeUnaccessData, DdeAccessData, DdeCreateDataHandle, DdeClientTransaction, DdeNameService, DdePostAdvise, DdeSetUserHandle, DdeQueryConvInfo, DdeDisconnect, DdeConnect, DdeUninitialize, DdeInitializeA	

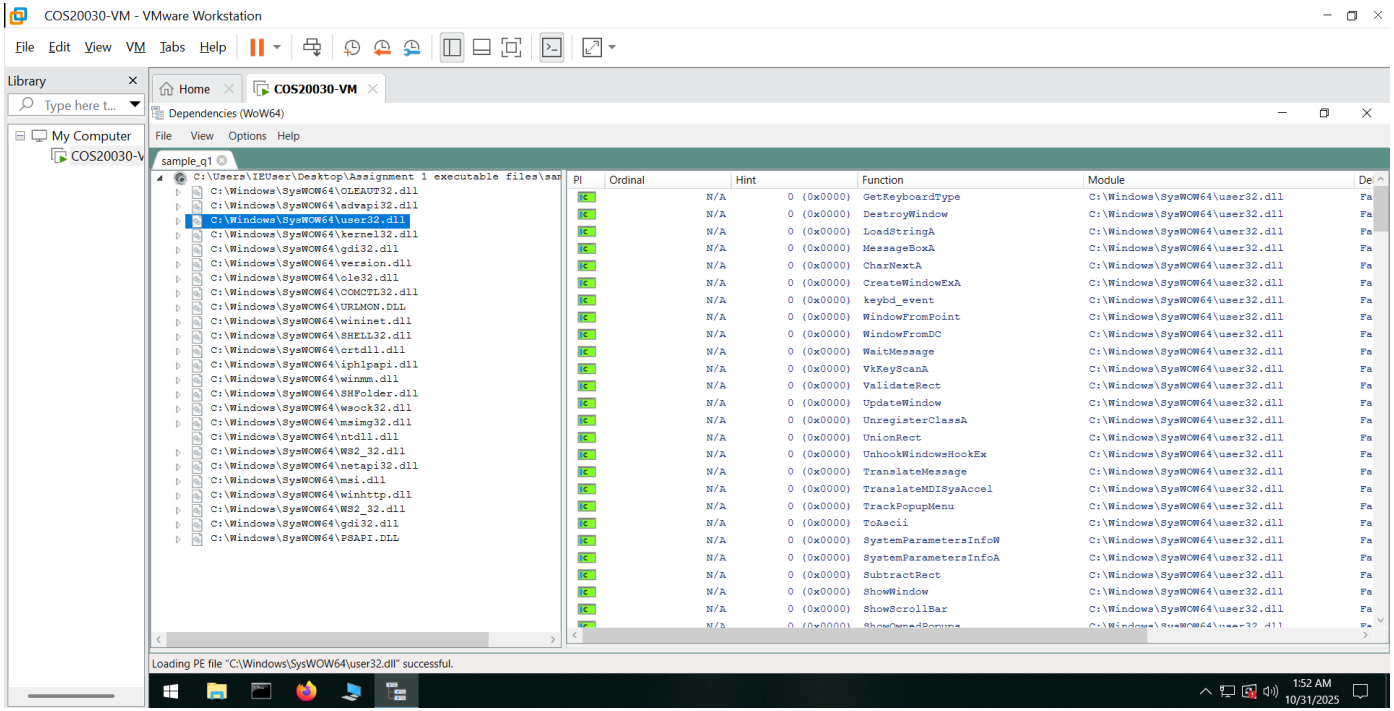


Figure 10: user32.dll and its imported functions part 1

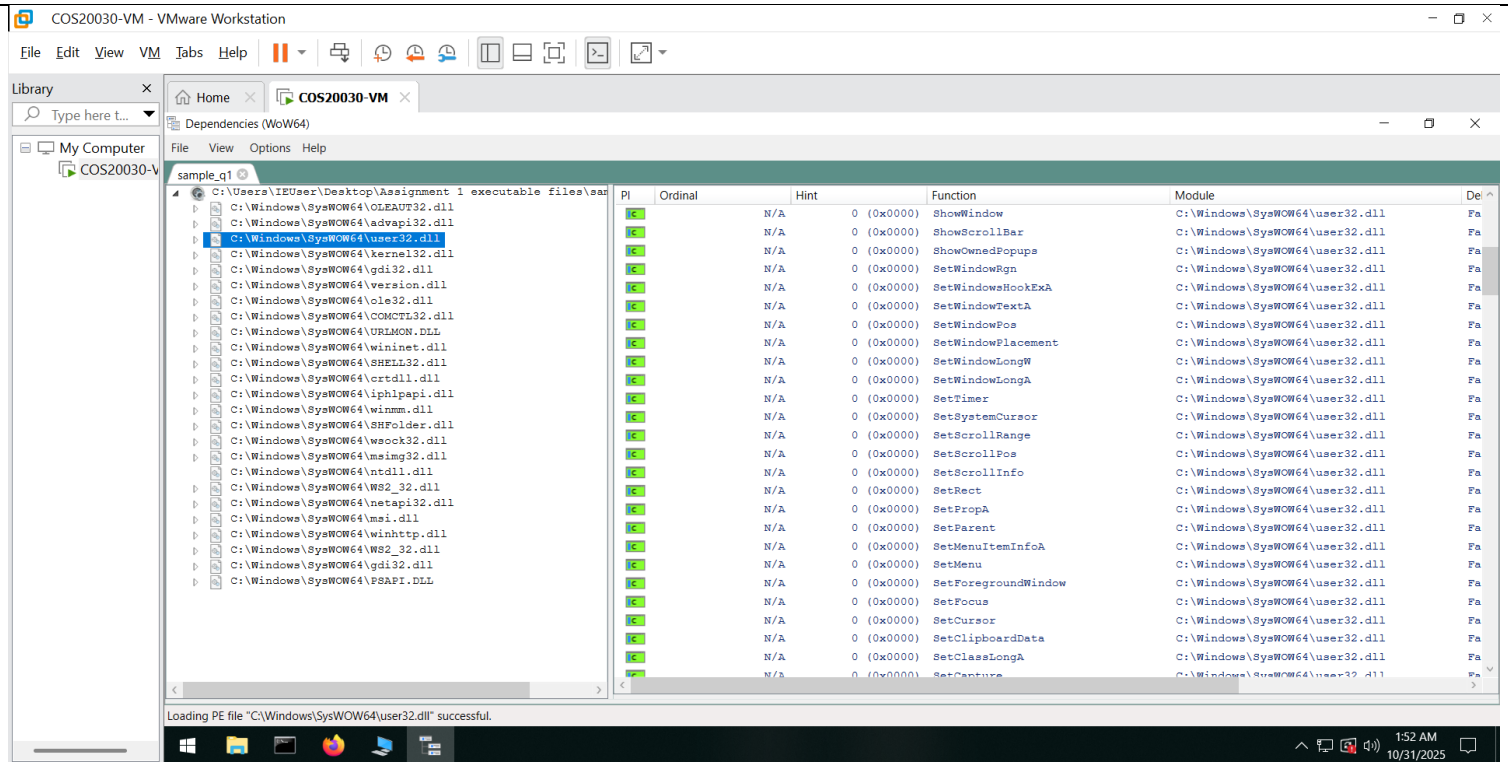


Figure 11: user32.dll and its imported functions part 2

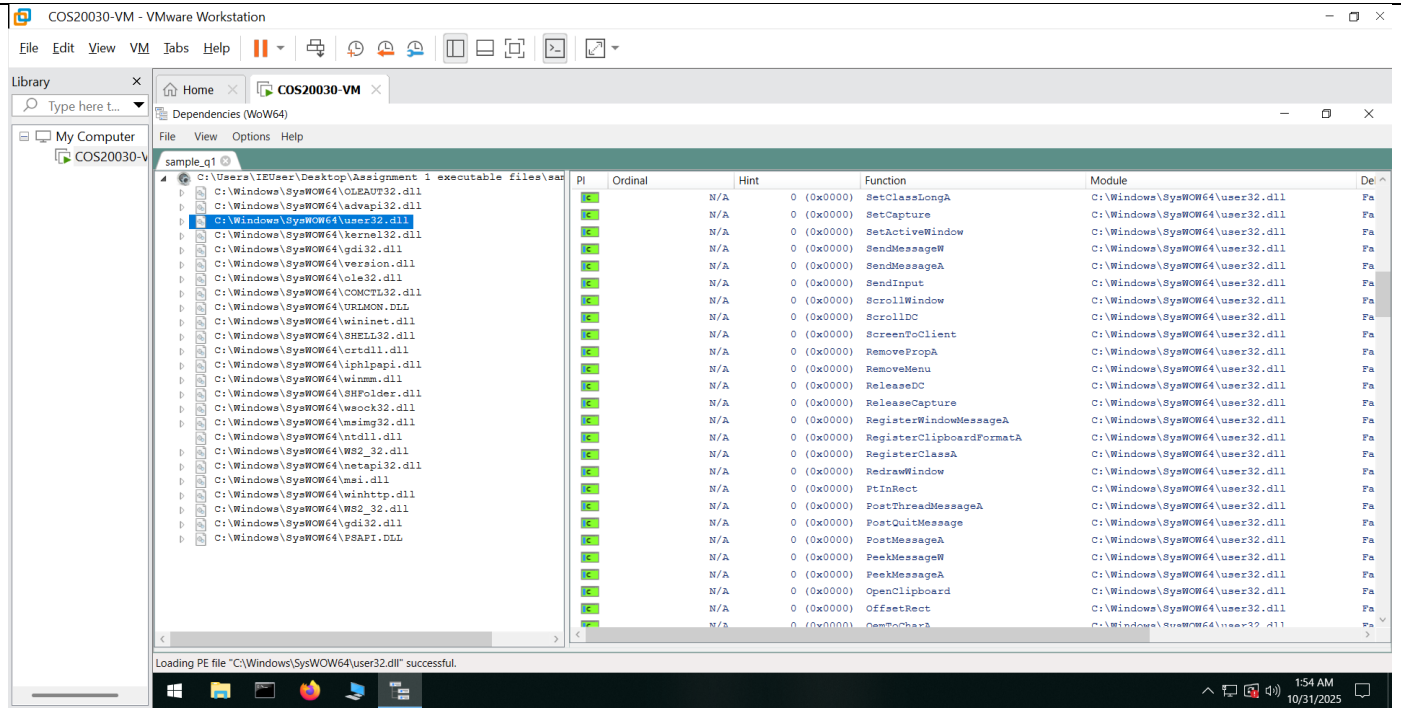


Figure 12: user32.dll and its imported functions part 3

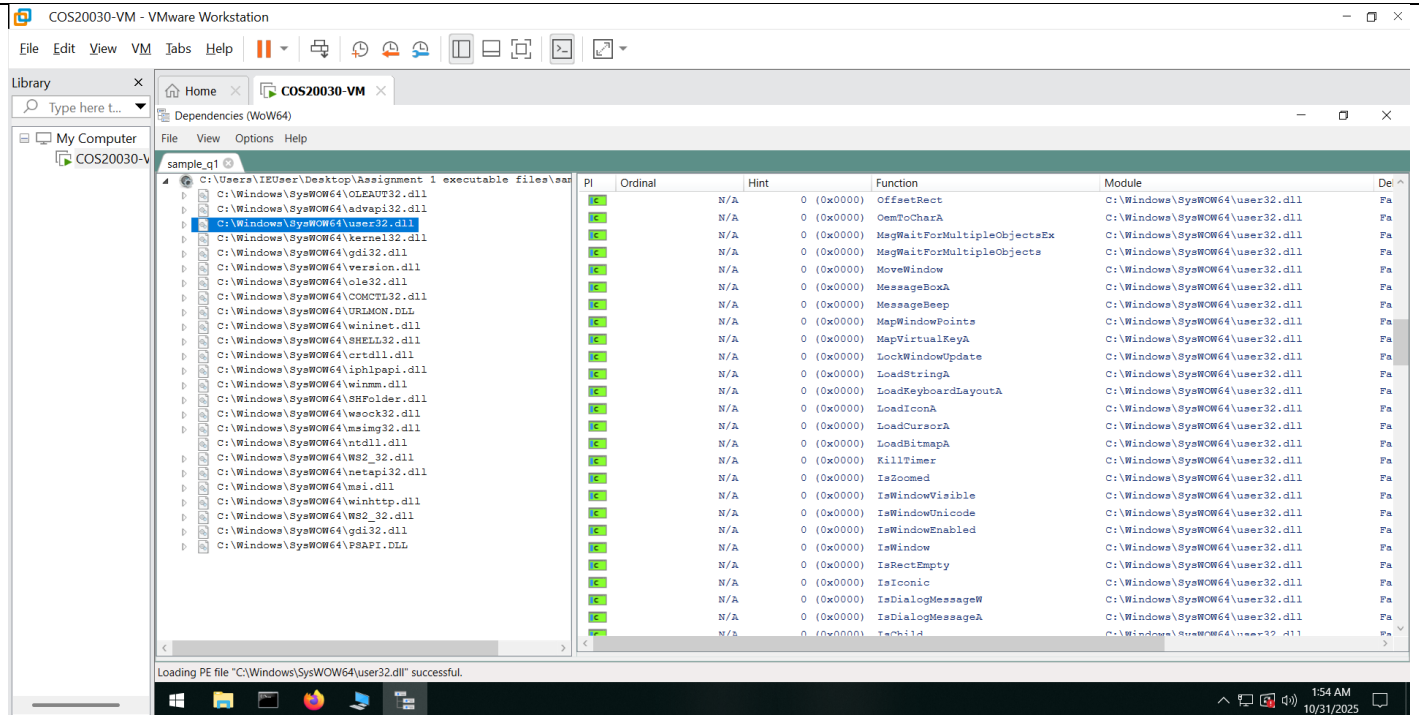


Figure 13: user32.dll and its imported functions part 4

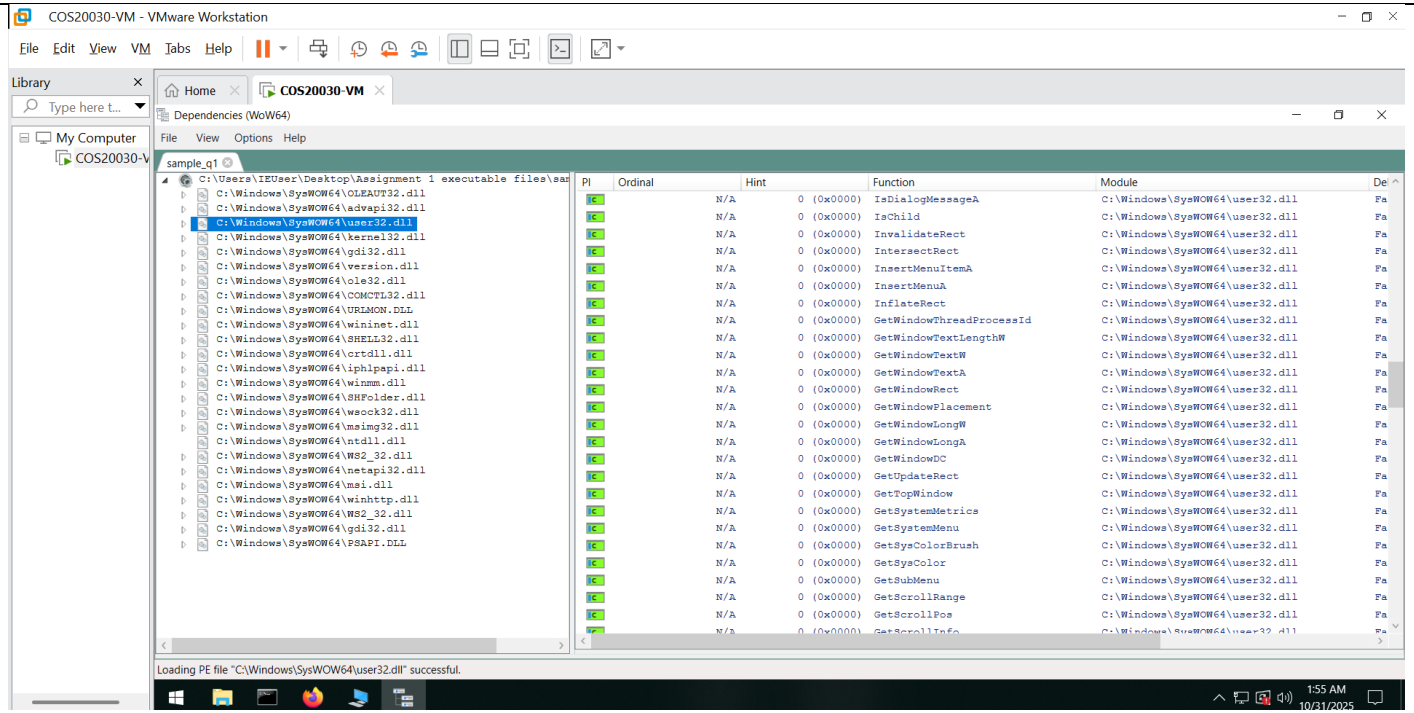


Figure 14: user32.dll and its imported functions part 5

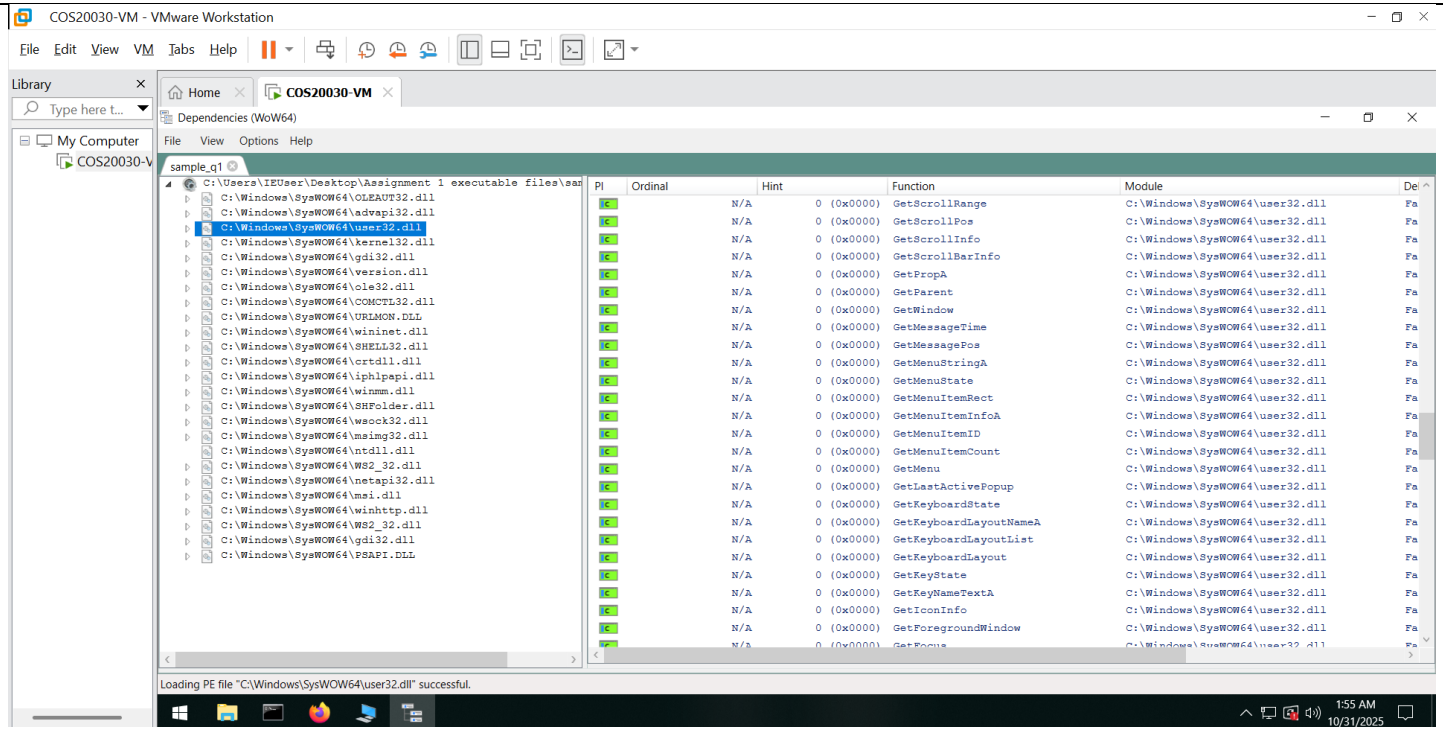


Figure 15: user32.dll and its imported functions part 6

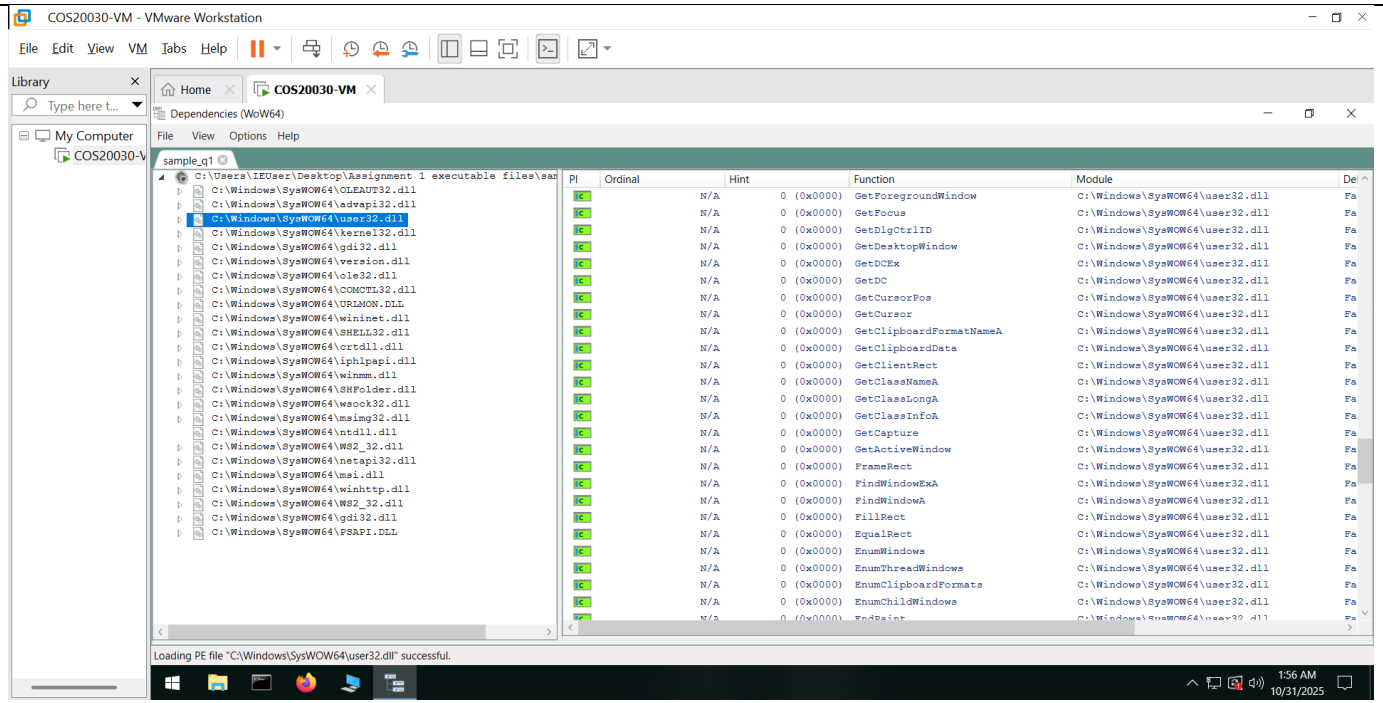


Figure 16: user32.dll and its imported functions part 7

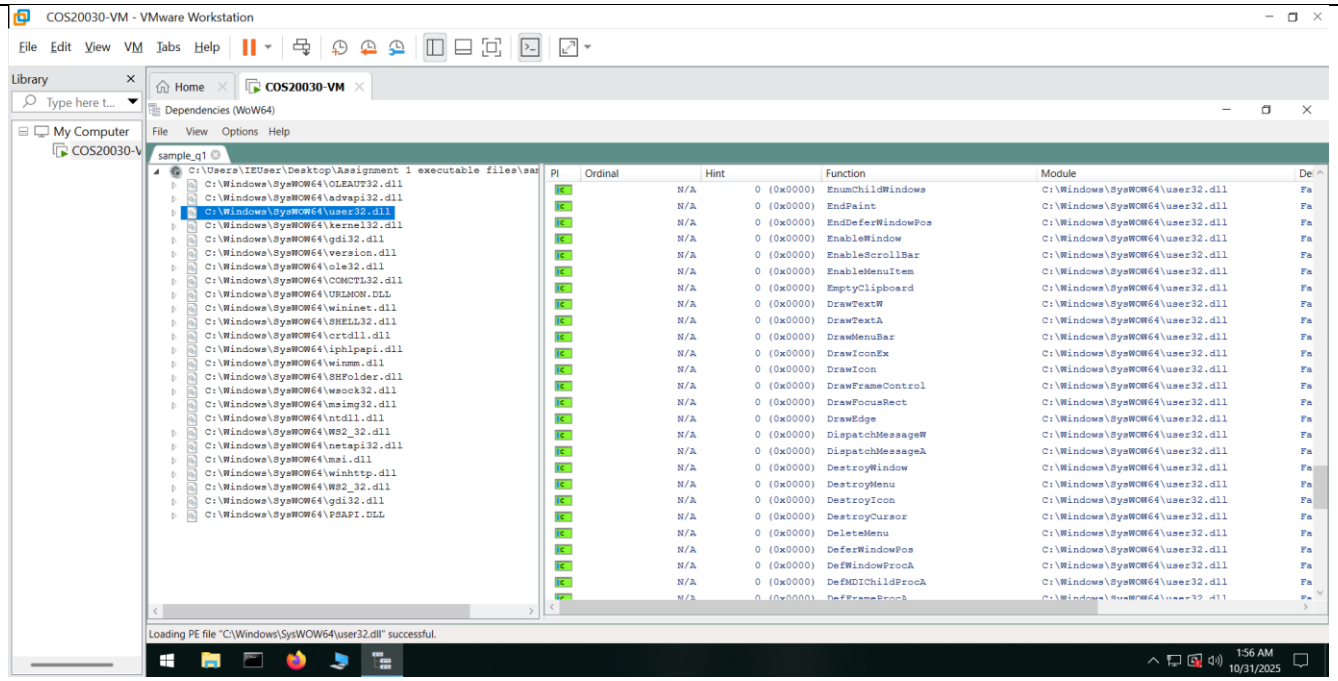


Figure 17: user32.dll and its imported functions part 8

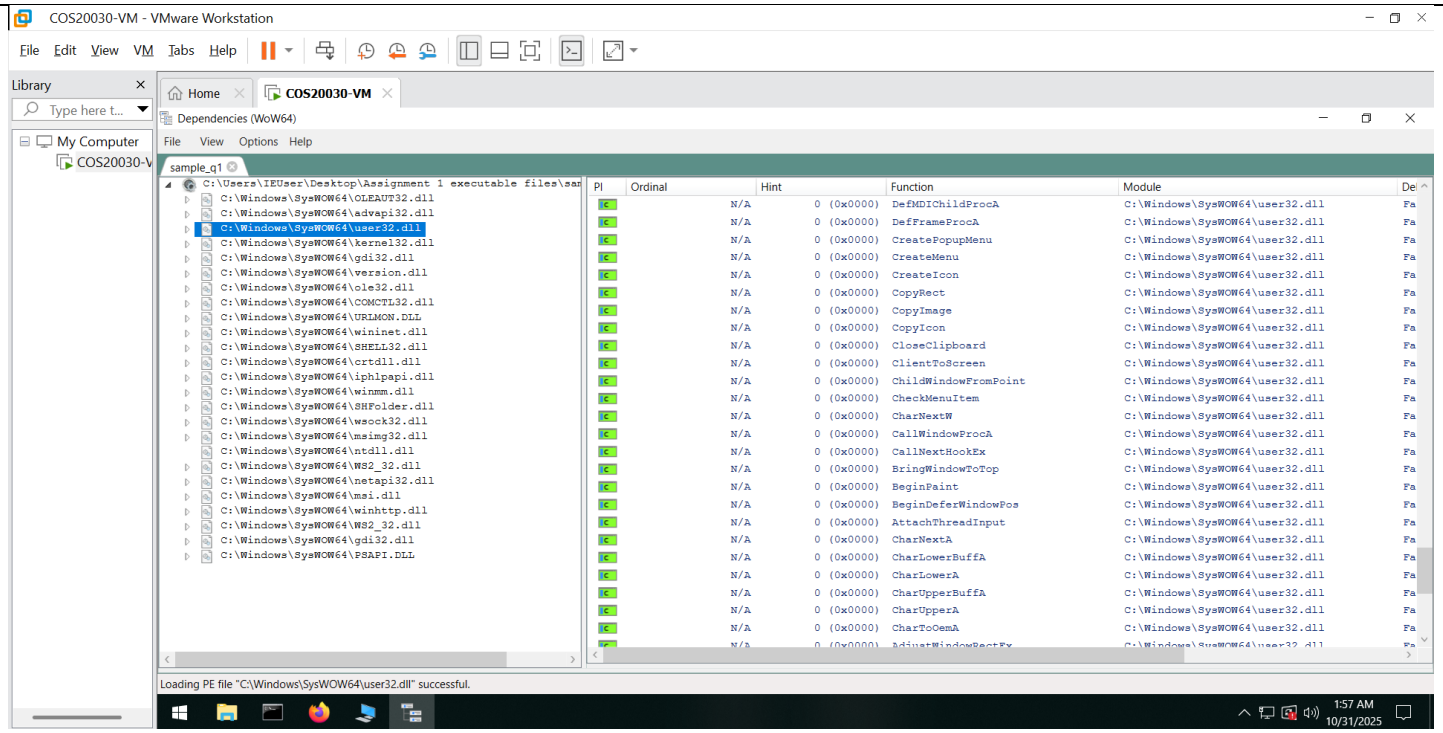


Figure 18: user32.dll and its imported functions part 9

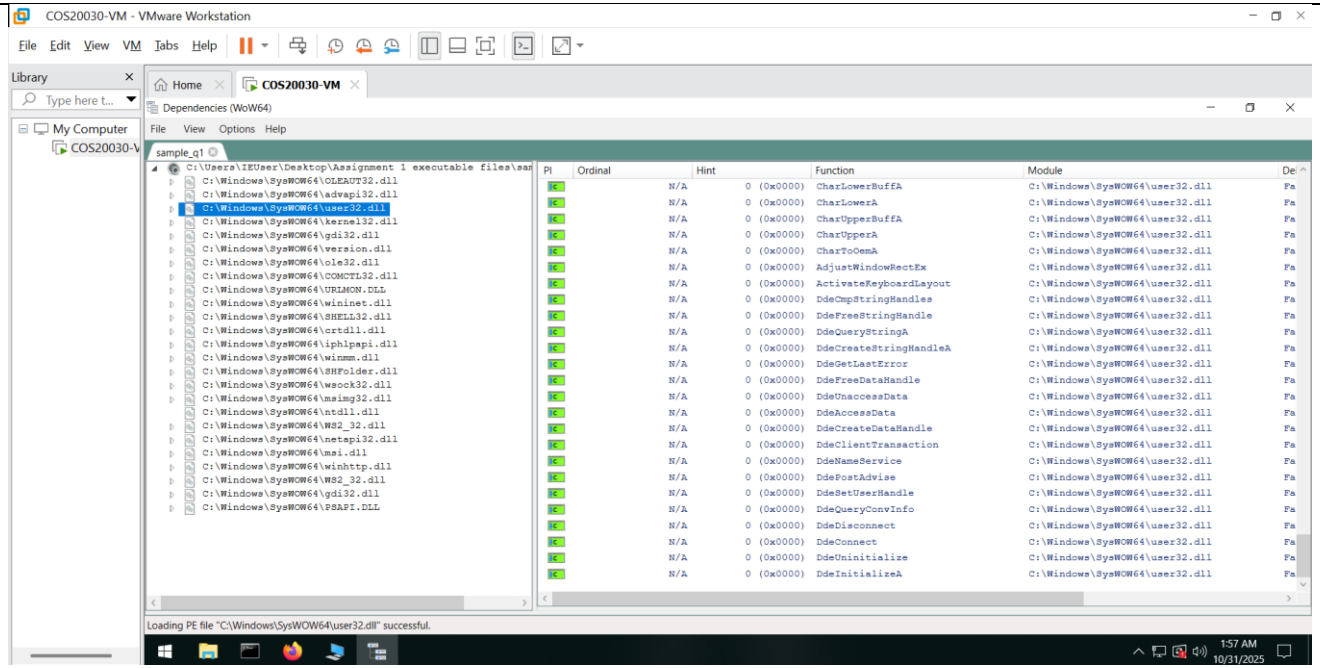


Figure 19: user32.dll and its imported functions part 10

4. kernel32.dll: GetACP, Sleep, VirtualFree, VirtualAlloc, GetTickCount, QueryPerformanceCounter, GetCurrentThreadId, InterlockedDecrement, InterlockedIncrement, VirtualQuery, WideCharToMultiByte, MultiByteToWideChar, IstrlenA, IstrcpynA, LoadLibraryExA, GetThreadLocale, GetStartupInfoA, GetProcAddress, GetModuleHandleA, GetModuleFileNameA, GetLocaleInfoA, GetLastError, GetCommandLineA, FreeLibrary, FindFirstFileA, FindClose, ExitProcess, ExitThread, CreateThread, CompareStringA,WriteFile, UnhandleExceptionFilter, SetFilePointer, SetEndOfFile, RtlUnwind, ReadFile, RaiseException, GetStdHandle, GetFileSize, GetFileType, CreateFileA, CloseHandle, TlsSetValue, TlsGetValue, LocalAlloc, GetModuleHandleA, IstrlenW, IstrcpyA, IstrcmpiA, IstrcmpA, WriteProcessMemory, WritePrivateProfilesStringA, WriteFile, WinExec, WideCharToMultiByte, WaitForSingleObject, WaitForMultipleObjectsEx, VirtualQuery, VirtualProtect, VirtualAllocEx, VirtualAlloc, VerLanguageNameA, TerminateProcess, SystemTimeToFileTime, SuspendThread, SleepEx, Sleep, SizeofResource, SetThreadPriority, SetThreadLocale, SetThreadAffinityMask, SetLastError, SetFilePointer, SetFileAttributesA, SetEvent, SetErrorMode, SetEndOfFile, ResumeThread, ResetEvent, RemoveDirectoryA, ReleaseMutex, ReadFile, RaiseException, QueryPerformanceFrequency, QueryPerformanceCounter, OpenProcess, MultiByteToWideChar, MulDiv, MoveFileA, LockResource, LoadResorce,

	LoadLibraryA, LeaveCriticalSection, InitializeCriticalSection, HeapFree, HeapDestroy, HeapCreate, HeapAlloc, GlobalUnlock, GlobalSize, GlobalHandle, GlobalLock, GlobalFree, GlobalFindAtomA, GlobalDeleteAtom, GlobalAlloc, GlobalAddAtomA, GetWindowsDirectoryA, GetVolumeInformationA, GetVersionExA, GetVersion, GetUserDefaultLCID, GetTimeZoneInformation, GetTickCount, GetThreadLocale, GetTempPathA, GetSystemTime, GetSystemInfo, GetSystemDirectoryA, GetSystemDefaultLangID, GetStdHandle, GetShortPathNameA, GetProcessHep, GetProcAddress, GetPrivateProfileStringA, GetModuleHandleA, GetModuleFileNameA, GetLocaleInfoW, GetLocaleInfoA, GetLocalTime, GetLastError, GetFullPathNameW, GetFullPathNameA, GetFileSize, GetFileAttributesA, GetExitCodeThread, GetExitCodeProcess, GetEnvironmentVariableA, GetDriveTypeA, GetDiskFreeSpaceA, GetDateFormatA, GetCurrentThreadId, GetCurrentThread, GetCurrentProcessId, GetCurrentProcess, GetComputerNameA, GetCPInfo, GetACP, FreeResource, InterlockedIncrement, InterlockedExchange, InterlockedDecrement, FreeLibrary, FormatMessageW, FormatMessageA, FindResourceW, FindResourceA, FindNextFileW, FindNextFileA, FindFirstFileW, FindFirstFileA, FindClose, FileTimeToSystemTime, FileTimeToLocalFileTime, FileTimeToDosDateTime, EnumResourceNamesA, EnumCalendarInfoA, EnterCriticalSection, DeviceIoControl, DeleteFileA, DeleteCriticalSection, CreateThread, CreateRemoteThread, CreateProcessA, CreatePipe, CreateMutexA, CreateFileW, CreateFileA, CreateEventA, CreateDirectoryA, CopyFileA, CompareStringW, CompareStringA, CloseHandle, Sleep, Process32Next, Process32First, CreateToolhelp32Snapshot, IsDebuggerPresent	

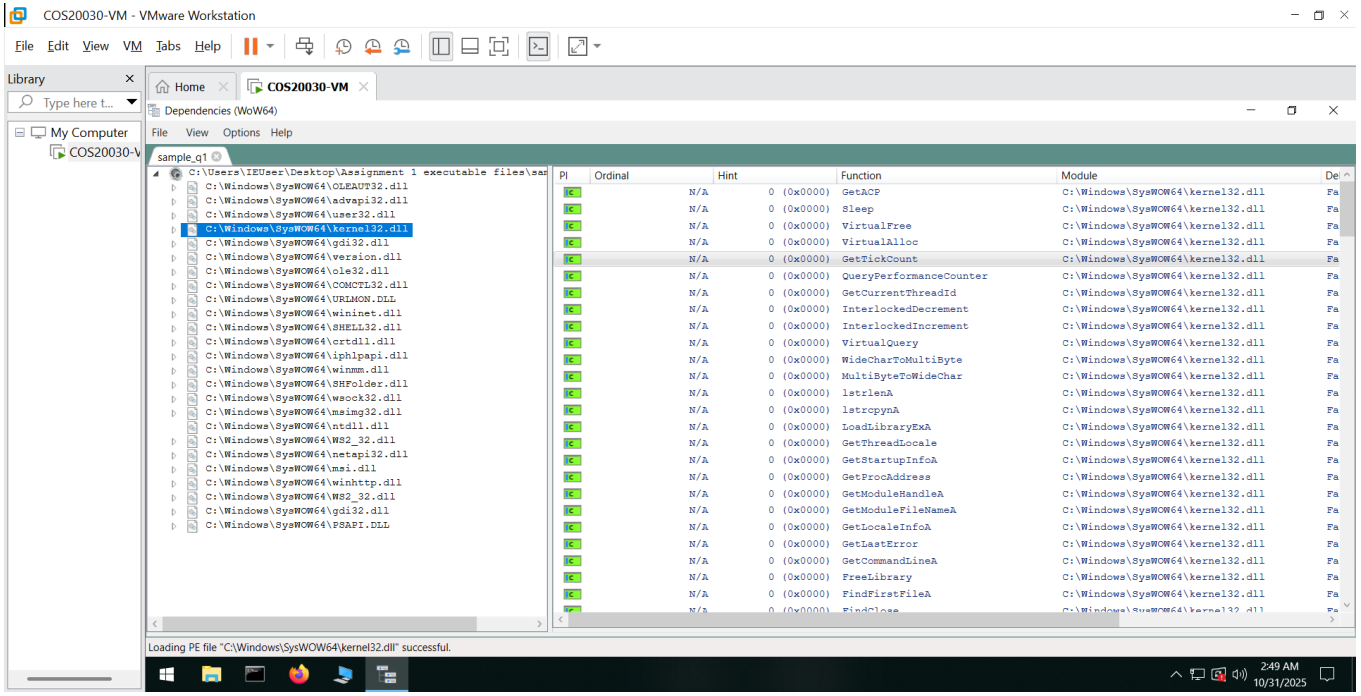


Figure 20: kernel32.dll and its imported functions part 1

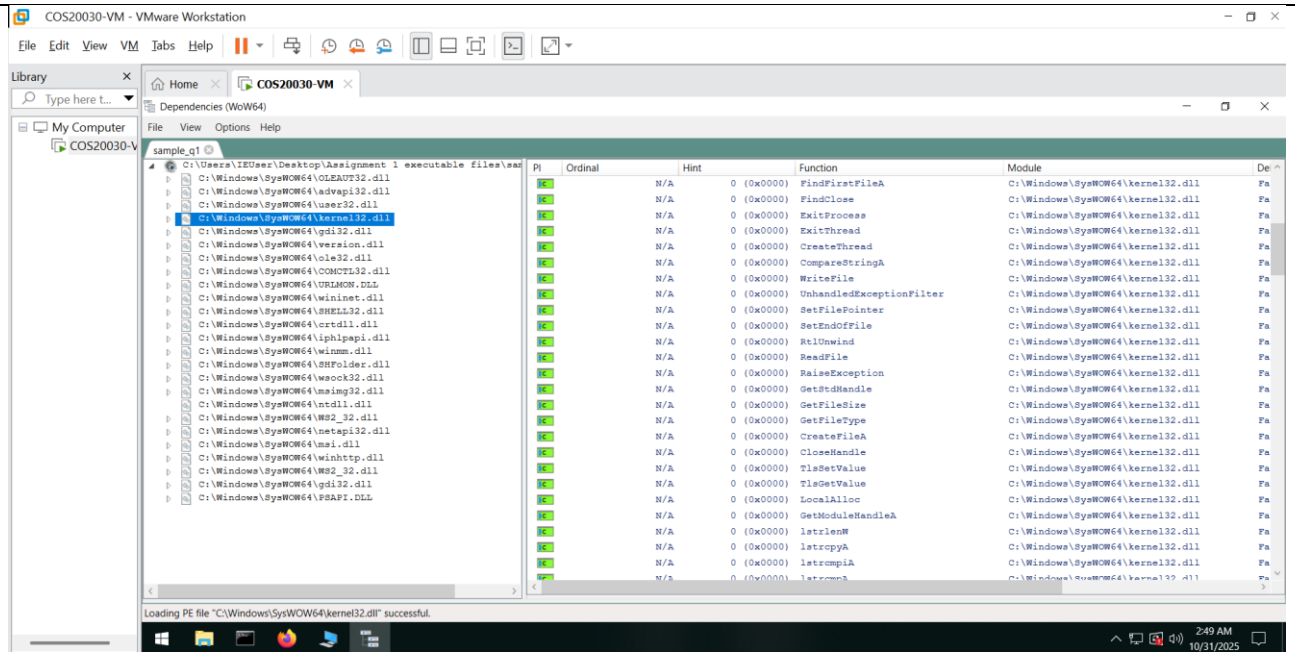


Figure 21: kernel32.dll and its imported functions part 2

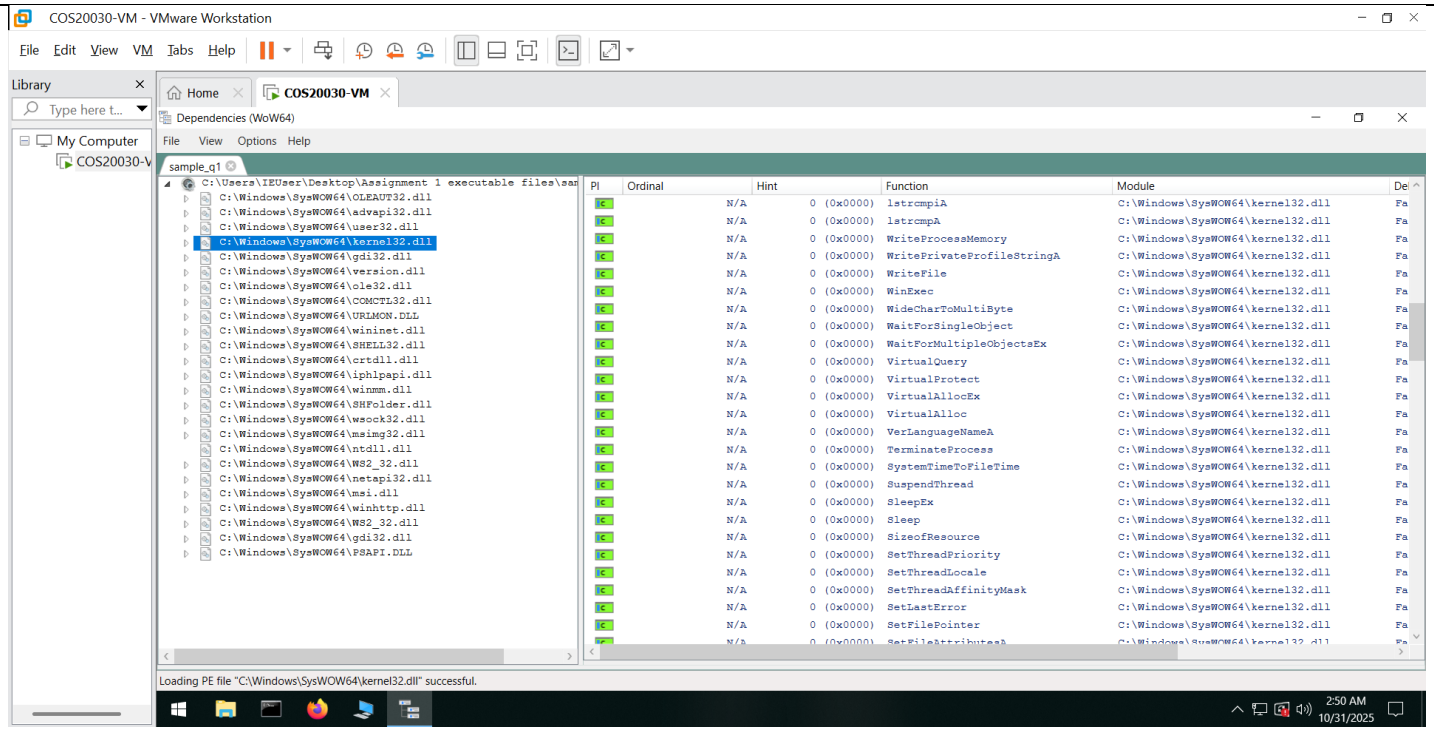


Figure 22: kernel32.dll and its imported functions part 3

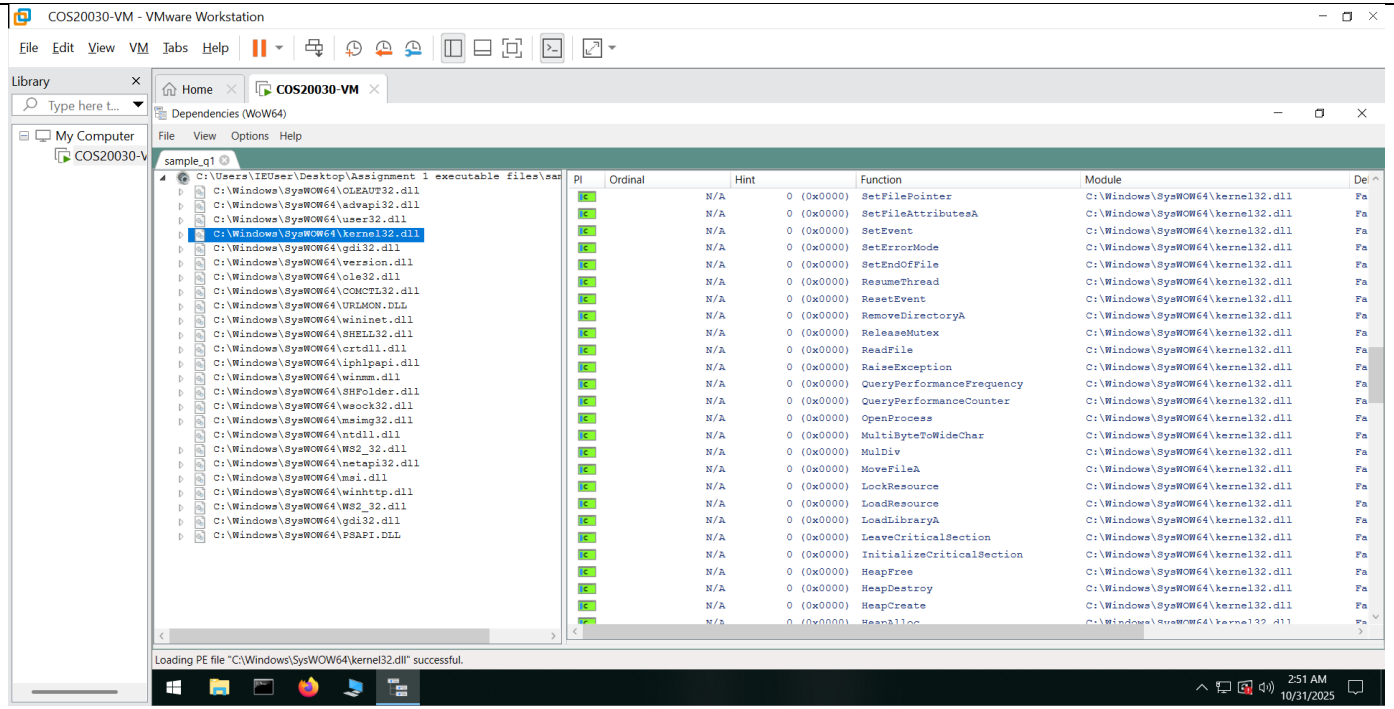


Figure 23: kernel32.dll and its imported functions part 4

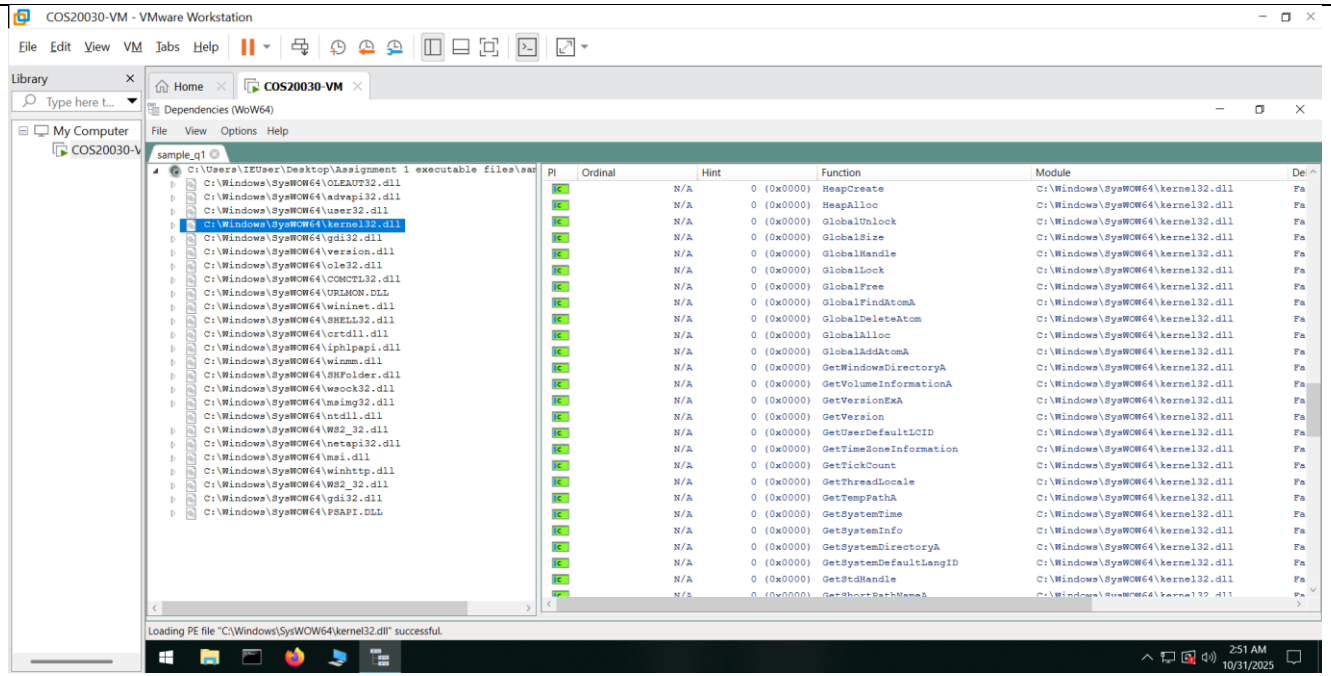


Figure 24: kernel32.dll and its imported functions part 5

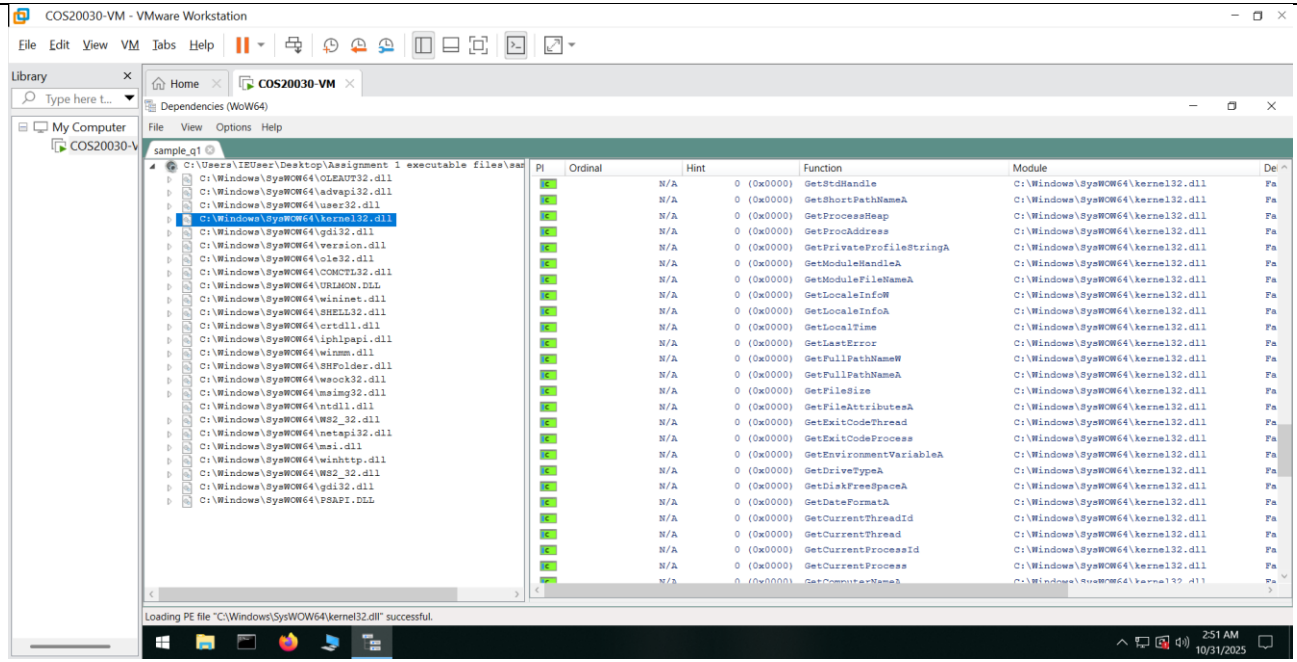


Figure 25: kernel32.dll and its imported functions part 6

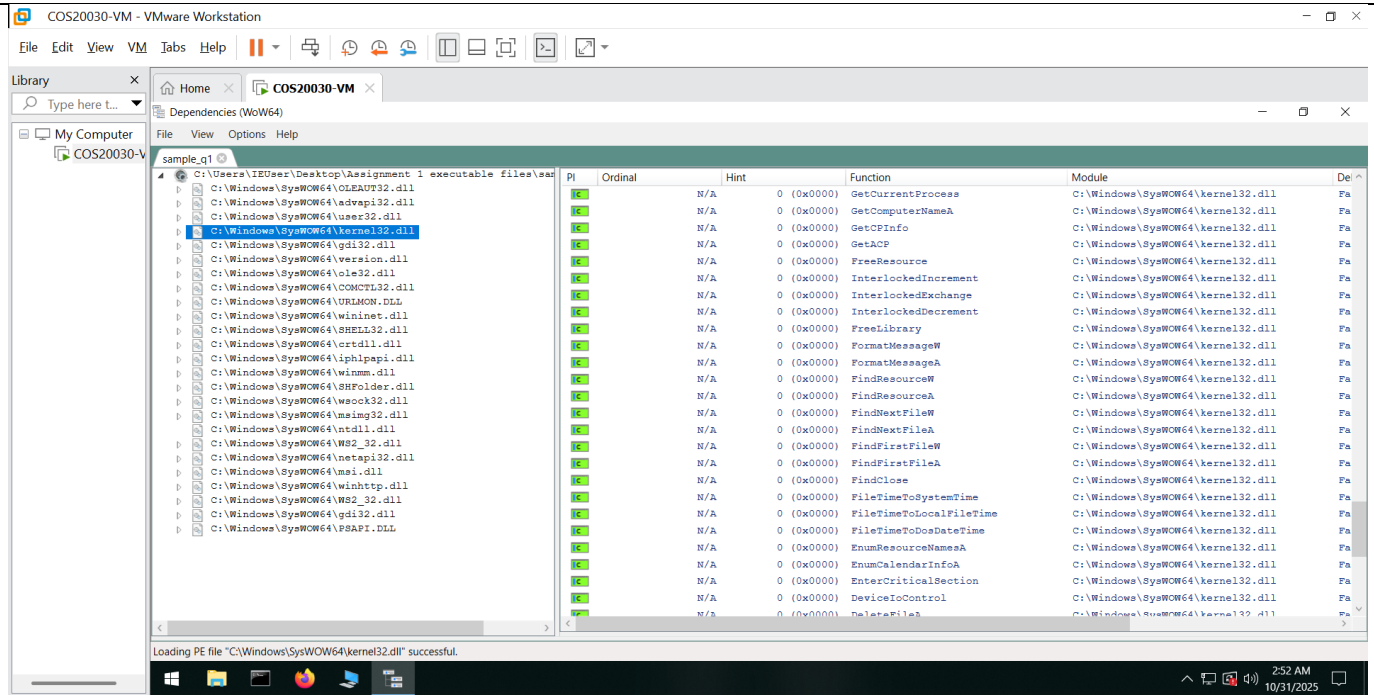


Figure 26: kernel32.dll and its imported functions part 7

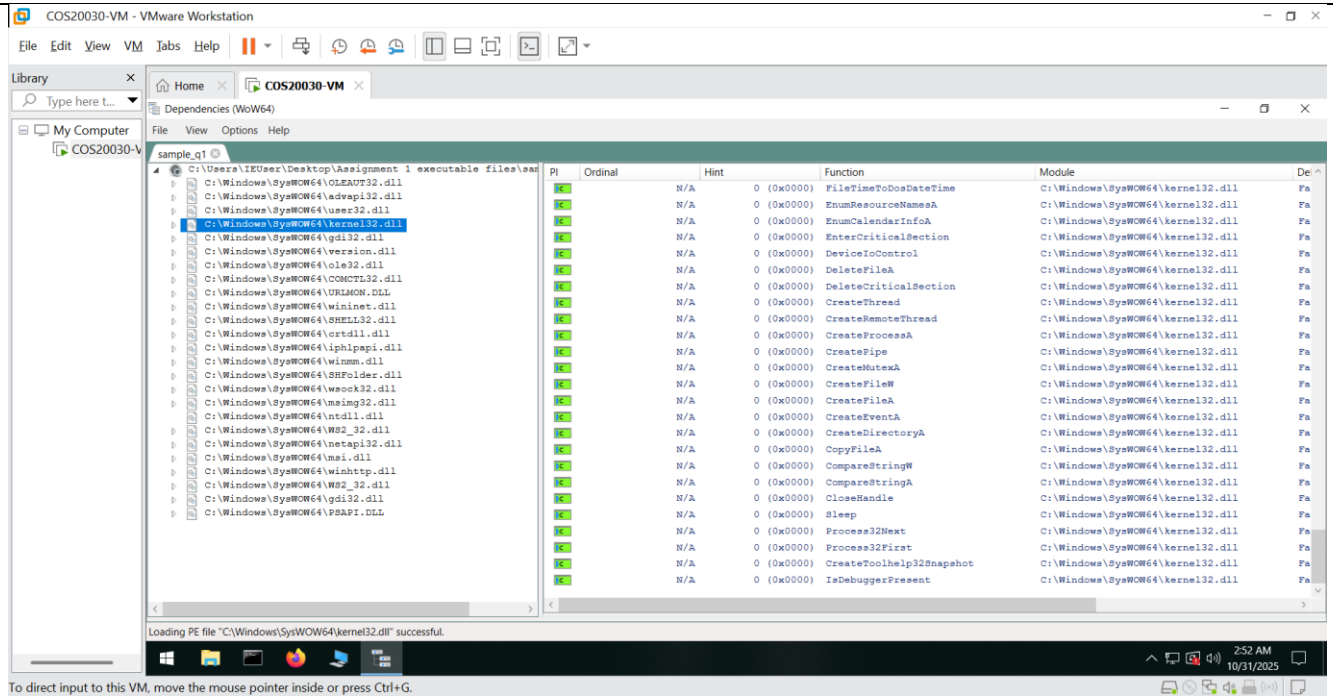


Figure 27: kernel32.dll and its imported functions part 8

5. gdi32.dll: UnrealizeObject, StretchDIBits, StretchBlt, SetWindowOrgEx, SetWinMetaFileBits, SetViewportOrgEx, SetTextColor, SetTextAlign, SetStretchBltMode, SetROP2, SetPixel, SetMapMode, SetEnhMetaFileBits, SetDIBColorTable, SetBrushOrgEx, SetBkMode, SetBkColor, SelectPalette, SelectObject, SelectClipRgn, SaveDC, RoundRect, RestoreDC, Rectangle, RectVisible, RealizePalette, Polyline, Polygon, PlayEnhMetaFile, PatBlt, OffsetRgn, MoveToEx, MaskBlt, LineTo, LPtoDP, IntersectClipRect, GetWindoOrgEx, GetWinMetaFileBits, GetTextMetricsA, GetTextExtentPointA, GetTextExtentPoint32W, GetTextExtentPoint32A, GetTextColor, GetTextAlign, GetSystemPaletteEntries, GetStockObject, GetRgnBox, GetPixel, GetPaletteEntries, GetObjectA, GetNearestColor, GetMapMode, GetEnhMetaFilePaletteEntries, GetEnhMetaFileHeader, GetEnhMetaFileDescriptionA, GetEnhMetFileBits, GetDeviceCaps, GetDIBits, GetDIBColorTable, GetDCOrgEx, GetCurrentPositionEx, GetCurrentObject, GetClipBox, GetBrushOrgEx, GetBkColor, GetBitmapDimensionEx, GetBitmapBits, GdiFlush, ExtTextOutW, ExtTextOutA, ExcludeClipRect, DeleteObject, DeleteEnhMetaFile, DeleteDC, CreateSolidBrush, CreateRectRgnIndirect, CreateRectRgn, CreatePenIndirect, CreatePen, CreatePatternBrush, CreatePalette, CreateHalftonePalette, CreateFontIndirectW, CreateFontIndirectA, CreateEnhMetaFileA, CreateDIBitmap,

CreateDIBSection, CreateCompatibleDC, CreateCompatibleBitmap, CreateBrushIndirect, CreateBitmap, CopyEnhMetaFileA, CombineRgn, CloseEnhMetaFile, BitBlt

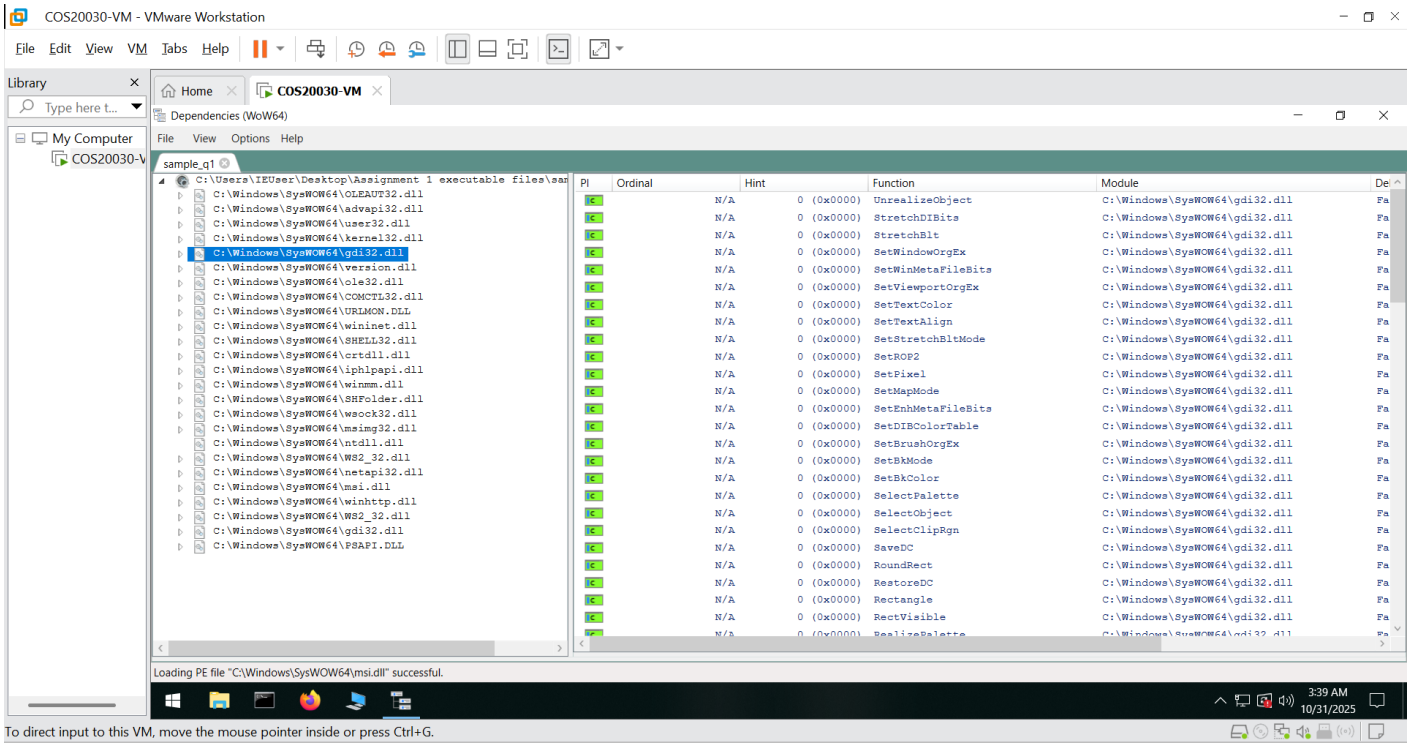


Figure 28: gdi32.dll and its imported functions part 1

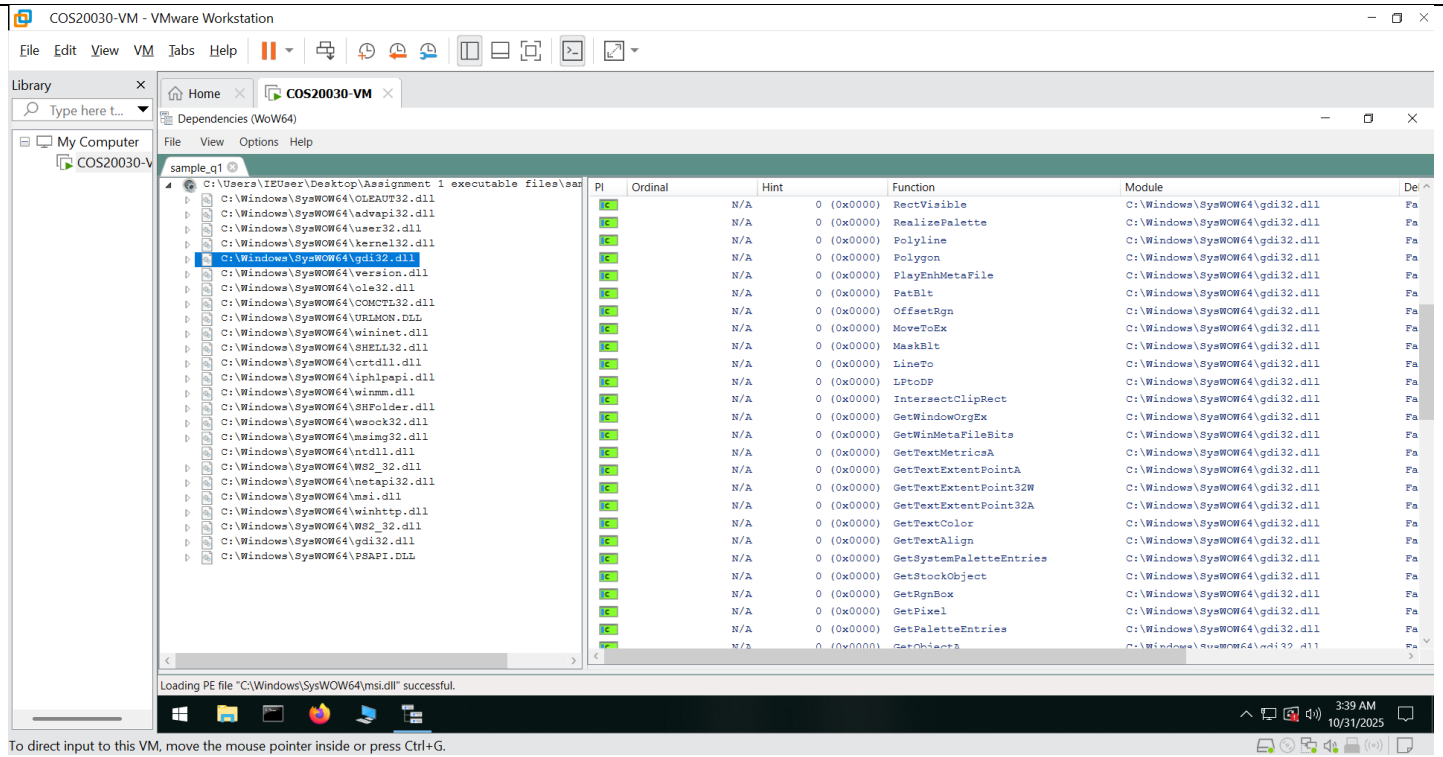


Figure 29: gdi32.dll and its imported functions part 2

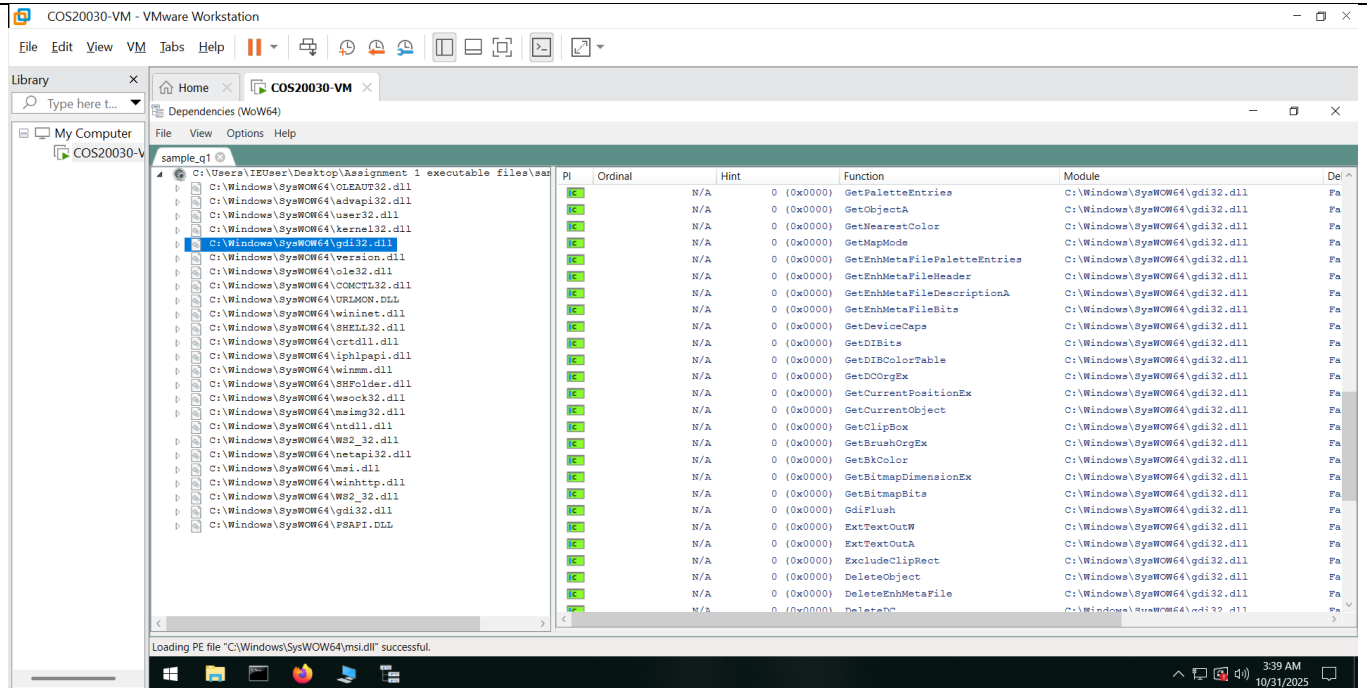


Figure 30: gdi32.dll and its imported functions part 3

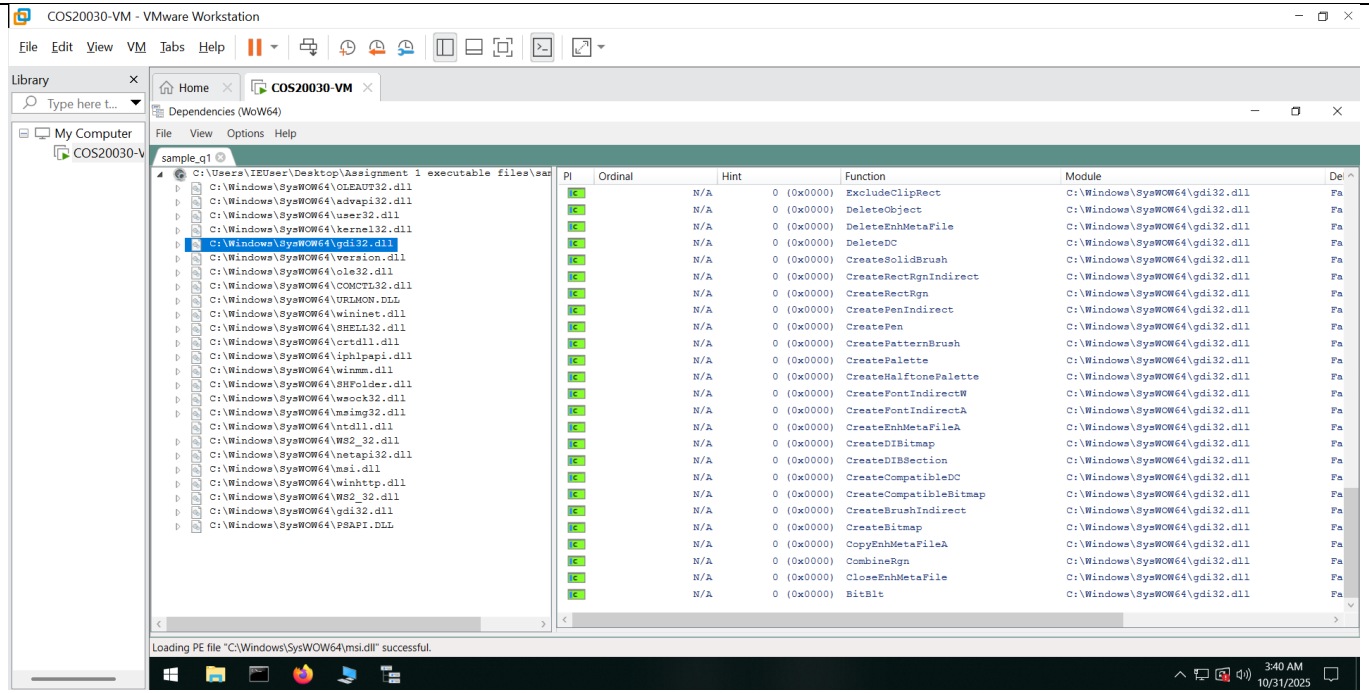


Figure 31: gdi32.dll and its imported functions part 4

6. version.dll: VerQueryValueA, GetFileVersionInfoSizeA, GetFileVersionInfoA

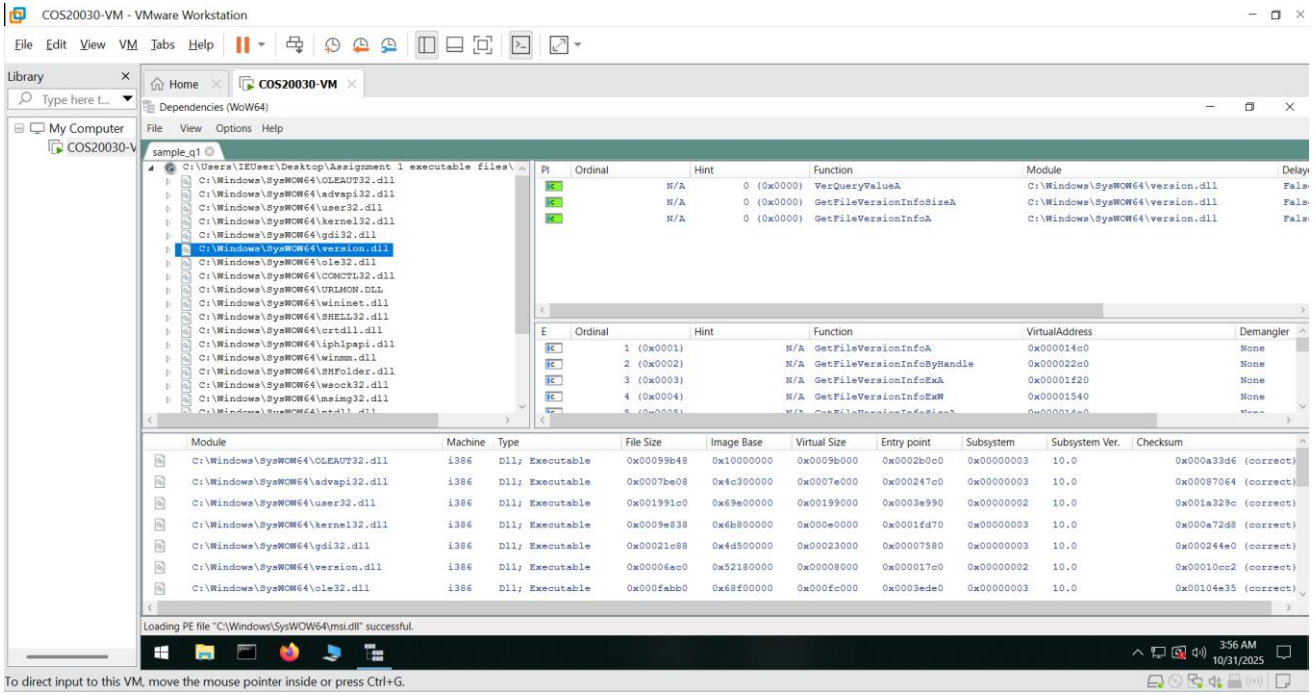


Figure 32: version.dll and its imported functions

7. ole32.dll: CreateStreamOnHGlobal, IsAccelerator, ReleaseStgMedium, OleDraw, OleSetMenuDescriptor, OletGetClipboard, OleSetClipboard, DoDragDrop, RevokeDragDrop, RegisterDragDrop, OleUninitialize, OleInitialize, CreateDataAdviseHolder, CoTaskMemFree, CoTaskMemAlloc, ProgIDFromCLSID, StringFromCLSID, CoCreateInstance, CoLockObjectExternal, CoDisconnectObject, CoRevokeClassObject, CoRegisterClassObject, CoGetClassObject, CoUninitialize, CoInitializeEx, CoInitialize, IsEqualGUID

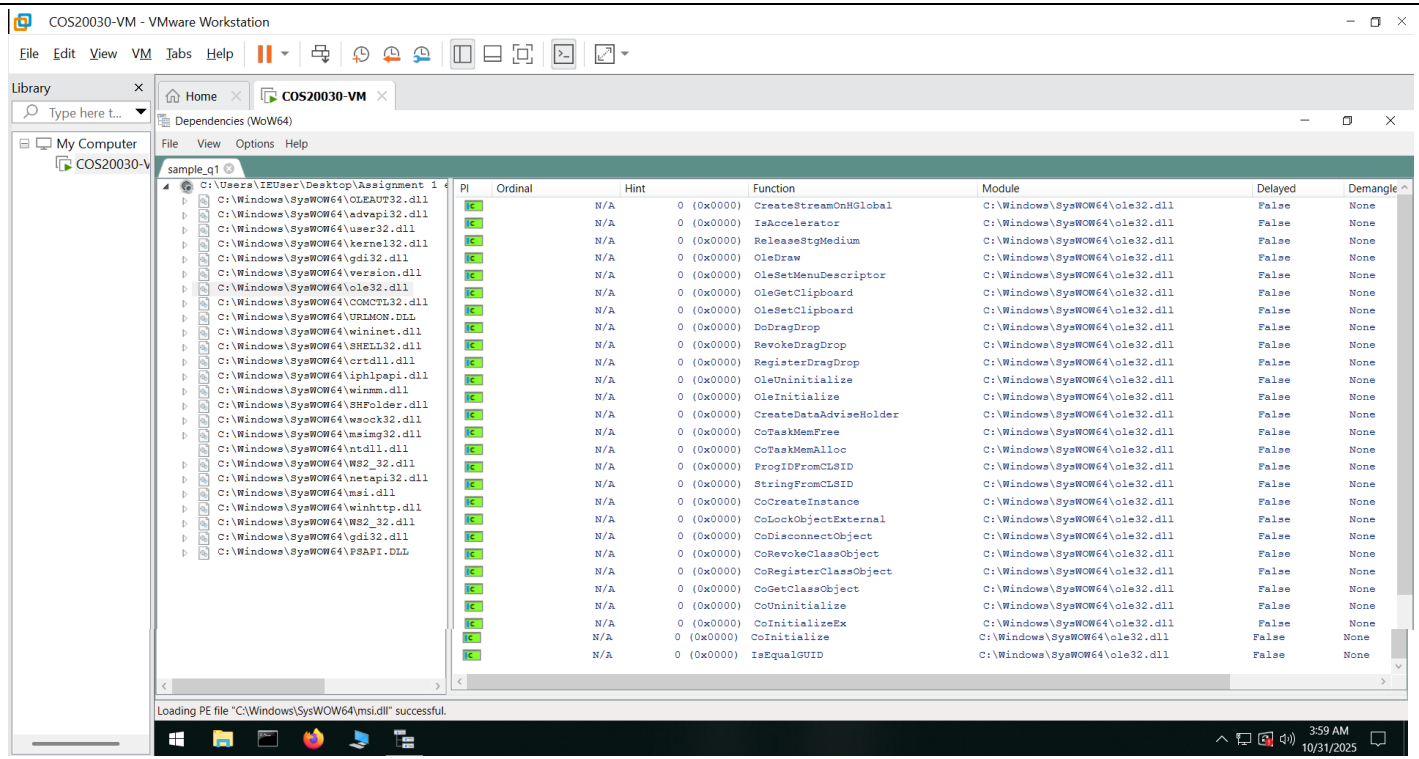


Figure 33: ole32.dll and its imported functions

8. COMCTL32.dll: UninitializeFlatSB, InitializeFlatSB, _TrackMouseEvent, ImageList_SetIconSize, ImageList_GetIconSize, ImageList_Write, ImageList_ Read, ImageList_GetDragImage, ImageList_DragShowNoLock, ImageList_DragMove, ImageList_DragLeave, ImageList_DragEnter, ImageList_EndDrag, ImageList_BeginDrag, ImageList_GetIcon, ImageList_Remove, ImageList_DrawEx, ImageList_Replace, ImageList_Draw, ImageList_GetBkColor, ImageList_SetBkColor, ImageList_Add, ImageList_GetImageCount, ImageList_Destroy, ImageList_Create, InitCommonControls

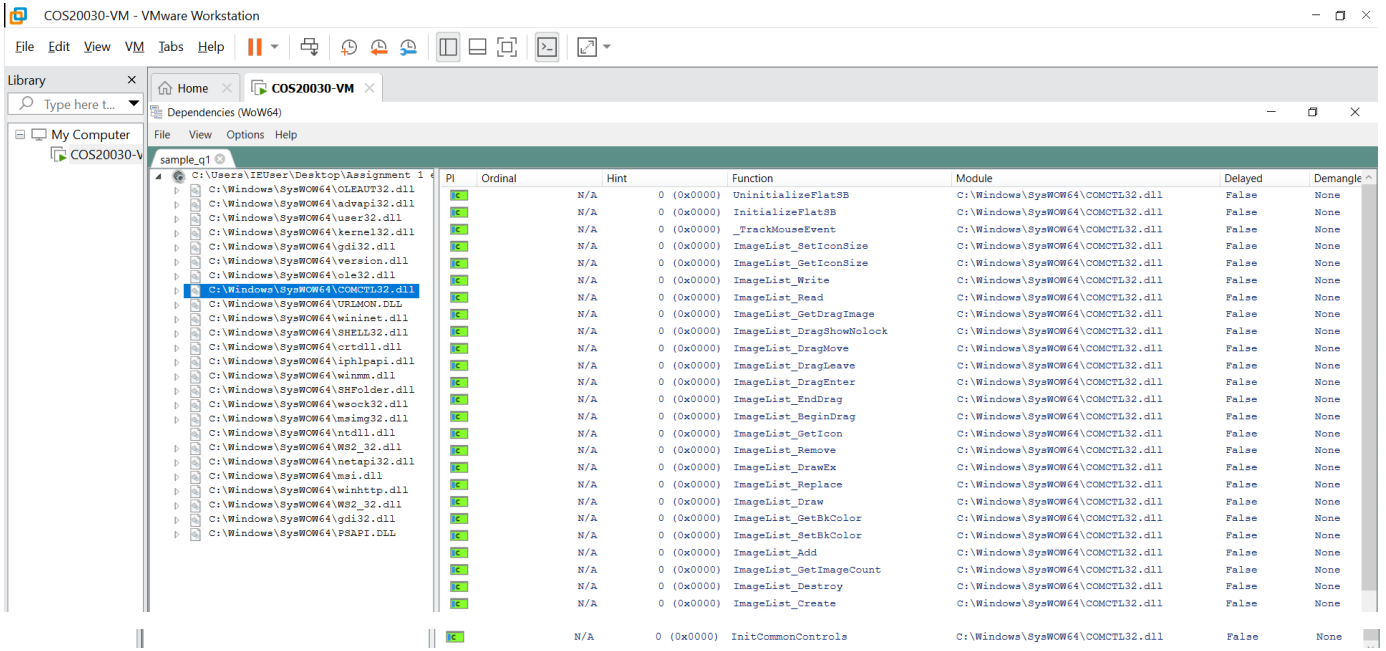


Figure 34 COMCTL32.dll and its imported functions

9. URLMON.DLL: UrlMkSetSessionOption

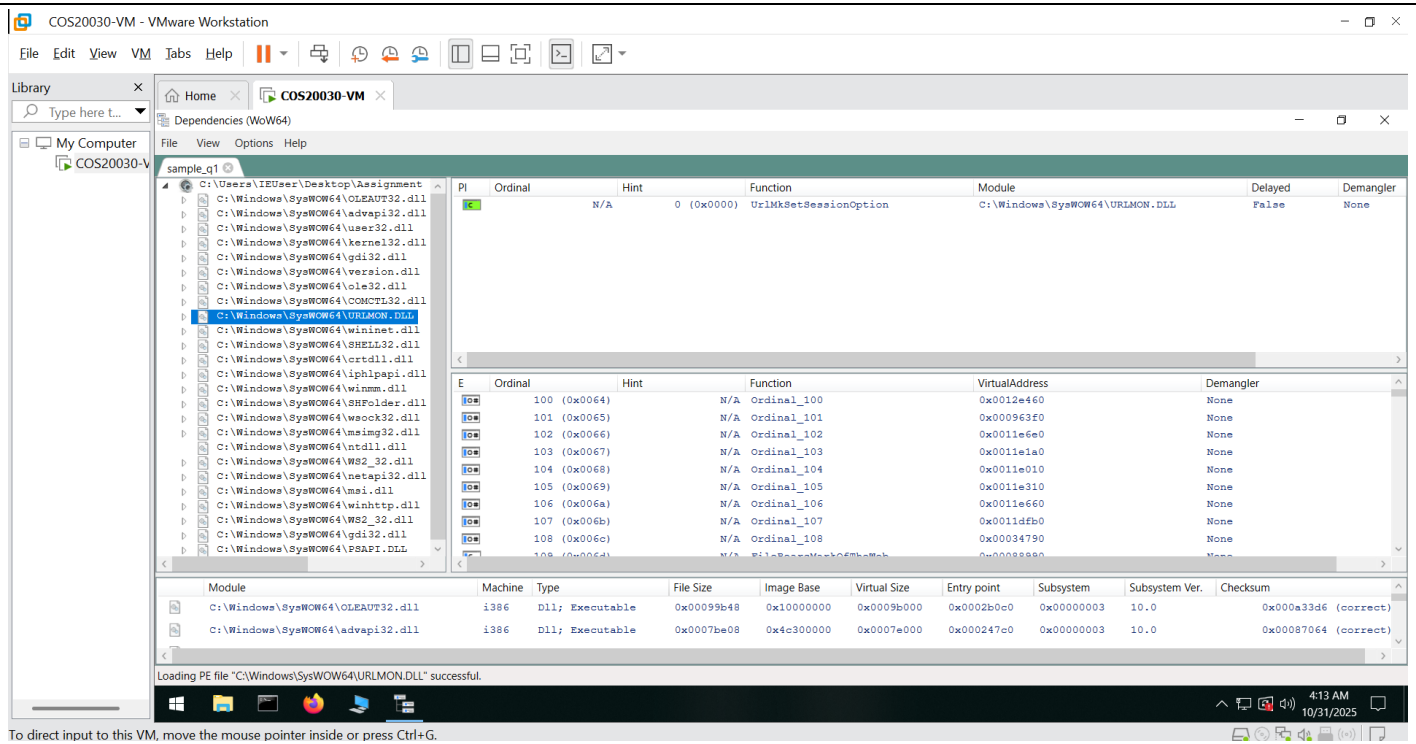


Figure 35: URLMON.DLL and its imported functions

10. wininet.dll: HttpSendRequestExA, InternetAttemptConnect, HttpEndRequestA, InternetWriteFile, InternetSetStatusCallback, InternetSetOptionA, InternetReadFile, InternetQueryDataAvailable, InternetOpenA, InternetCrackUrlA, InternetConnectA, InternetCloseHandle, HttpSendRequestA, HttpQueryInfoA, HttpOpenRequestA, HttpAddRequestHeadersA

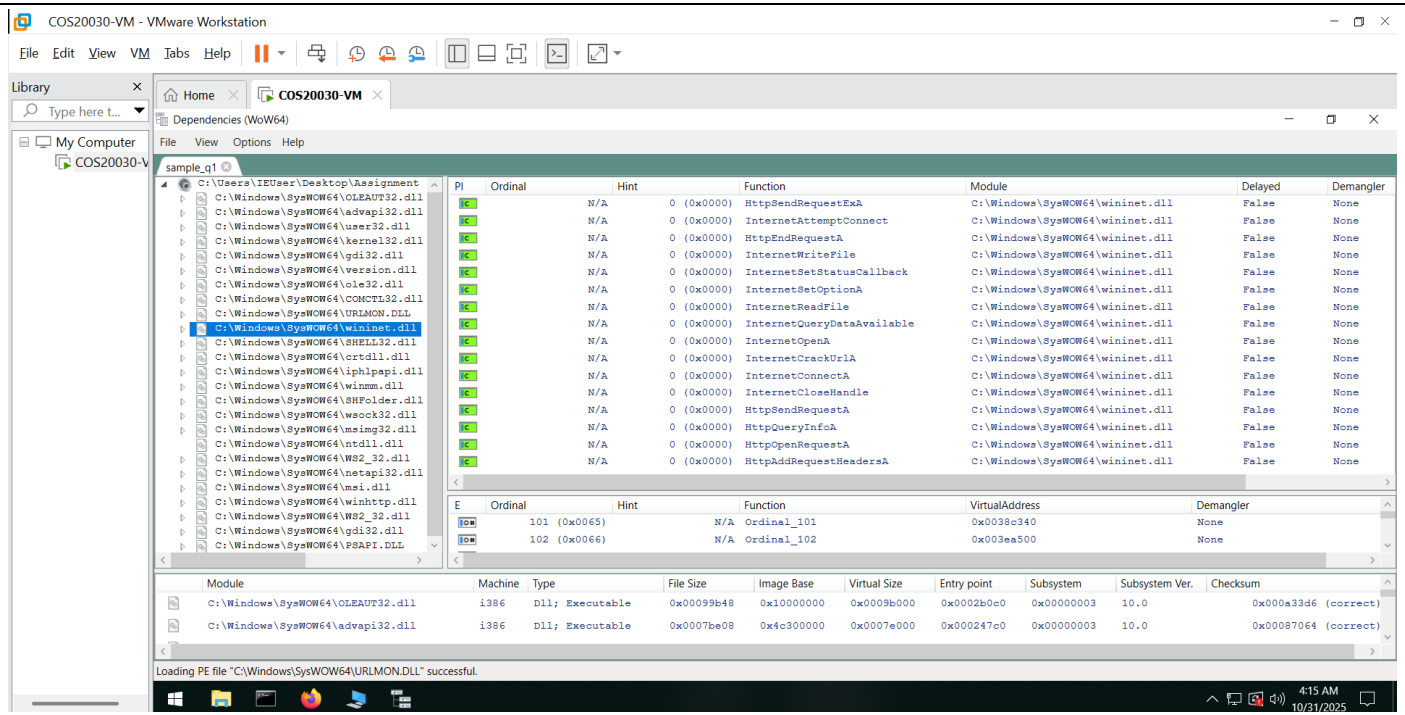


Figure 36: wininet.dll and its imported functions

11. SHELL32.dll: Shell_NotifyIconA, ShellExecuteExA, ShellExecuteA, SHGetFileInfoA, SHFileOperationA, SHGetSpecialFolderLocation, SHGetPathFromIDListA, SHGetMalloc, SHGetDesktopFolder, SHChangeNotify, SHBrowserForFolderA

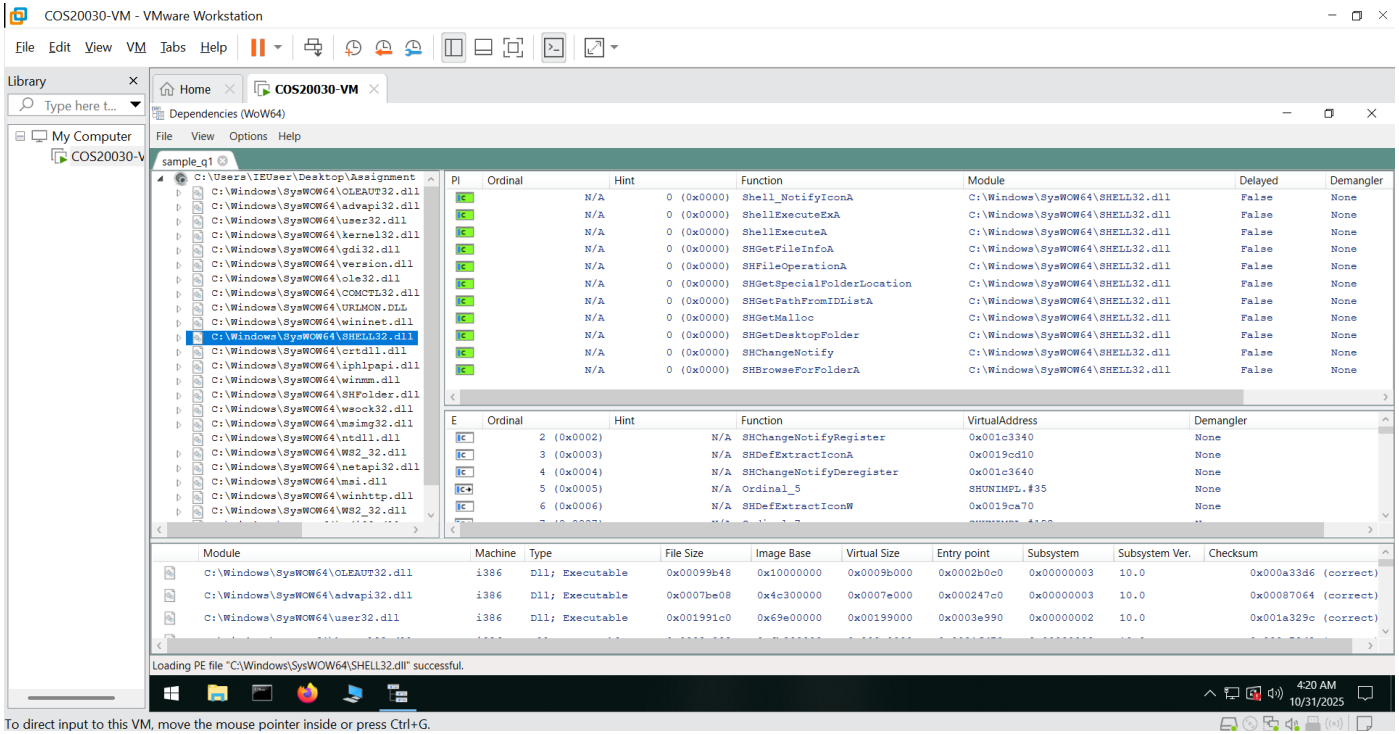


Figure 37: SHELL32.dll and its imported functions

12. crtdll.dll: isalnum, isprint, tolower, isspace

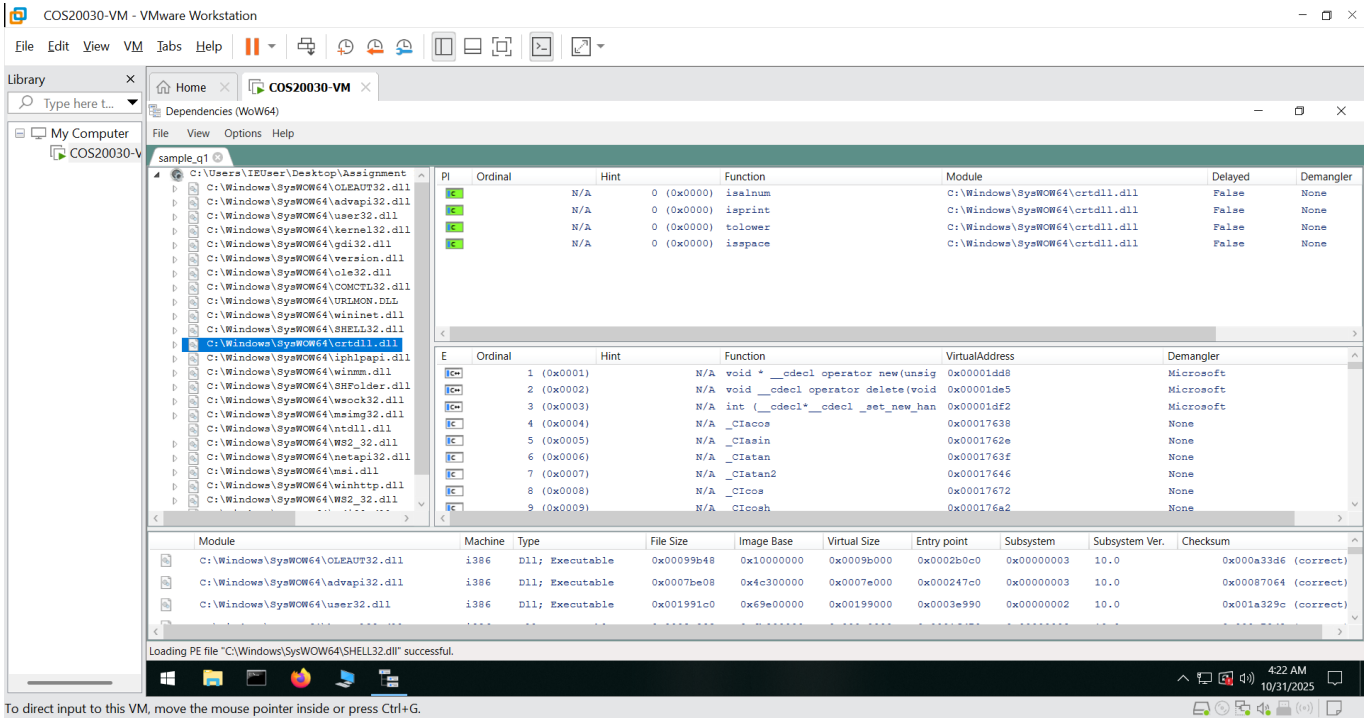


Figure 38: crtdll.dll and its imported functions

13. iphlpapi.dll: GetAdaptersInfo

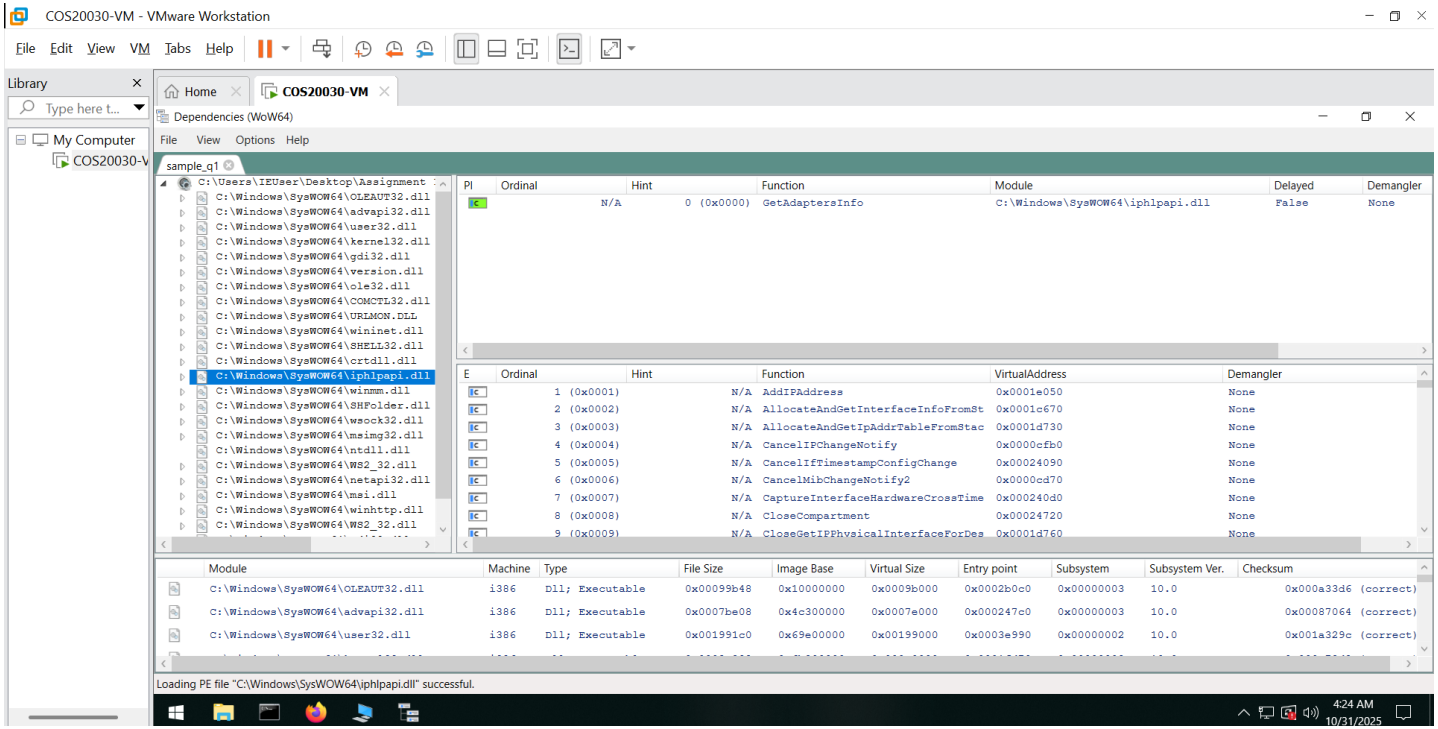


Figure 39: iphlpapi.dll and its imported functions

14. winmm.dll: timeGetTime, timeEndPeriod, timeBeginPeriod, PlaySoundA

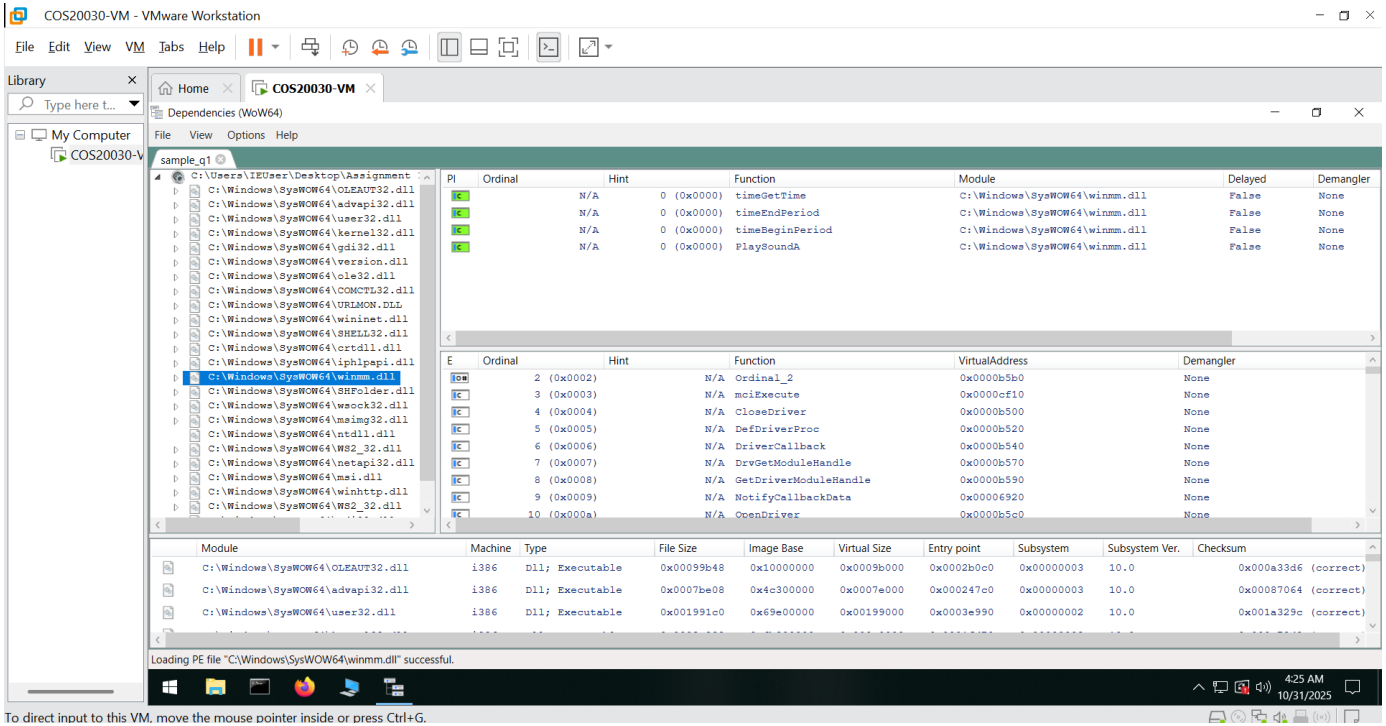


Figure 40: winmm.dll and its imported functions

15. SHFoler.dll: SHGetFolderPathA

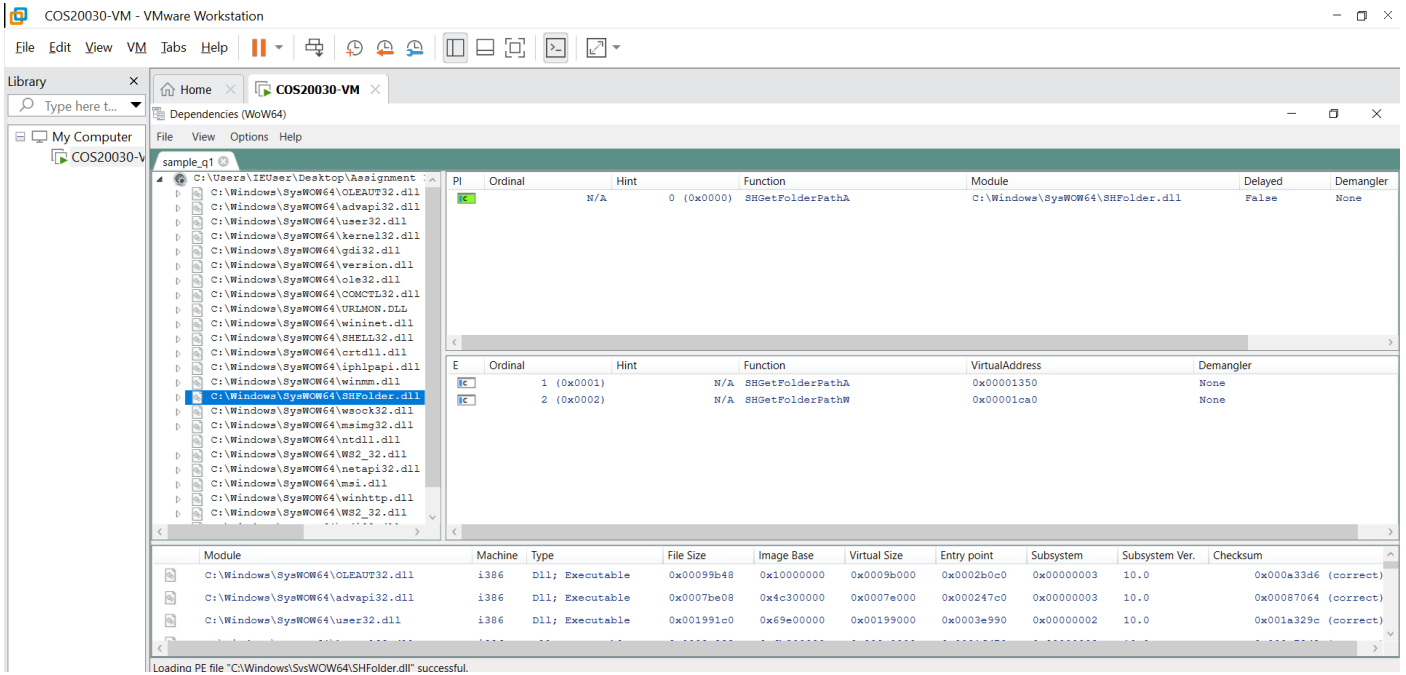


Figure 41: SHFoler.dll and its imported functions

16. wsock32.dll: WSACleanup, WSAStartup, WSAGetLastError, gethostname, gethostbyname, socket, sendto, inet_ntoa, inet_addr, htons, htonl, connect, closesocket

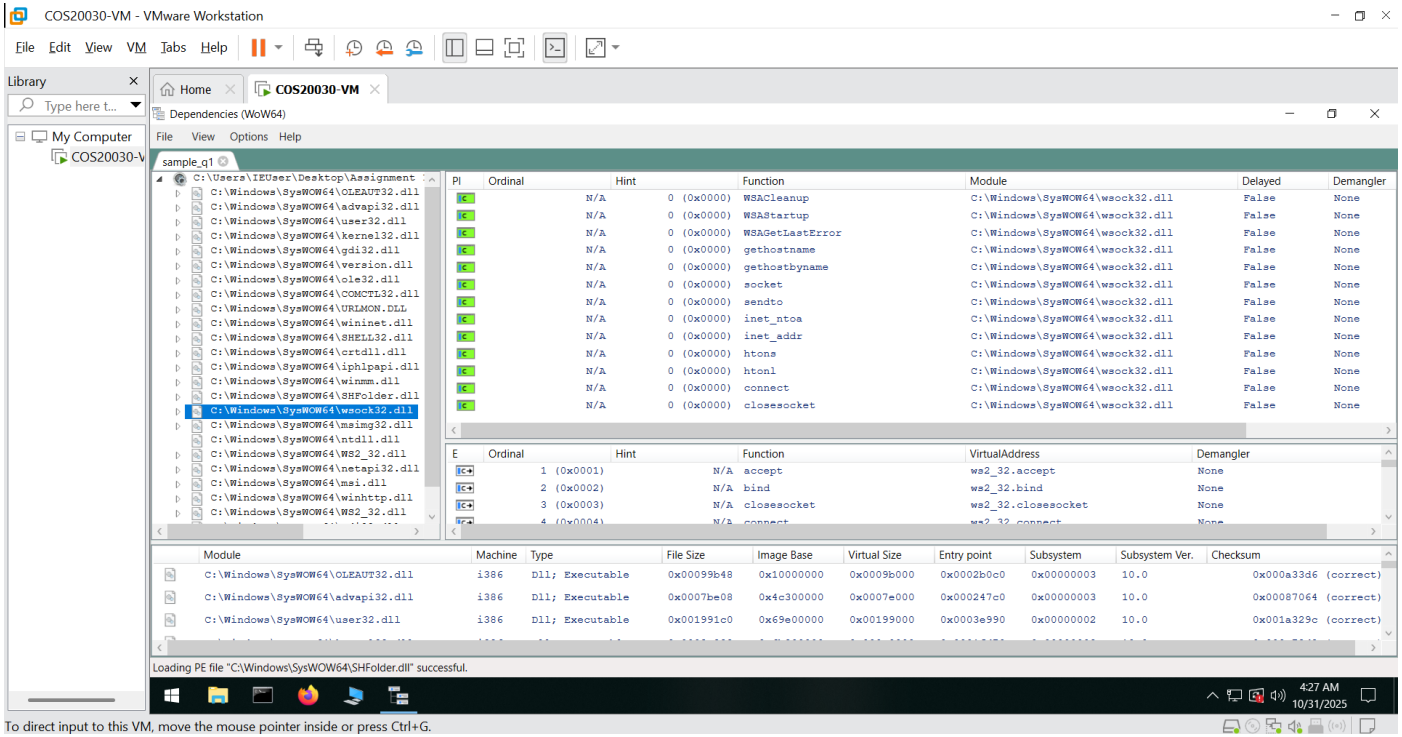


Figure 42: wsock32.dll and its imported functions

17. msimg32.dll: GradientFill

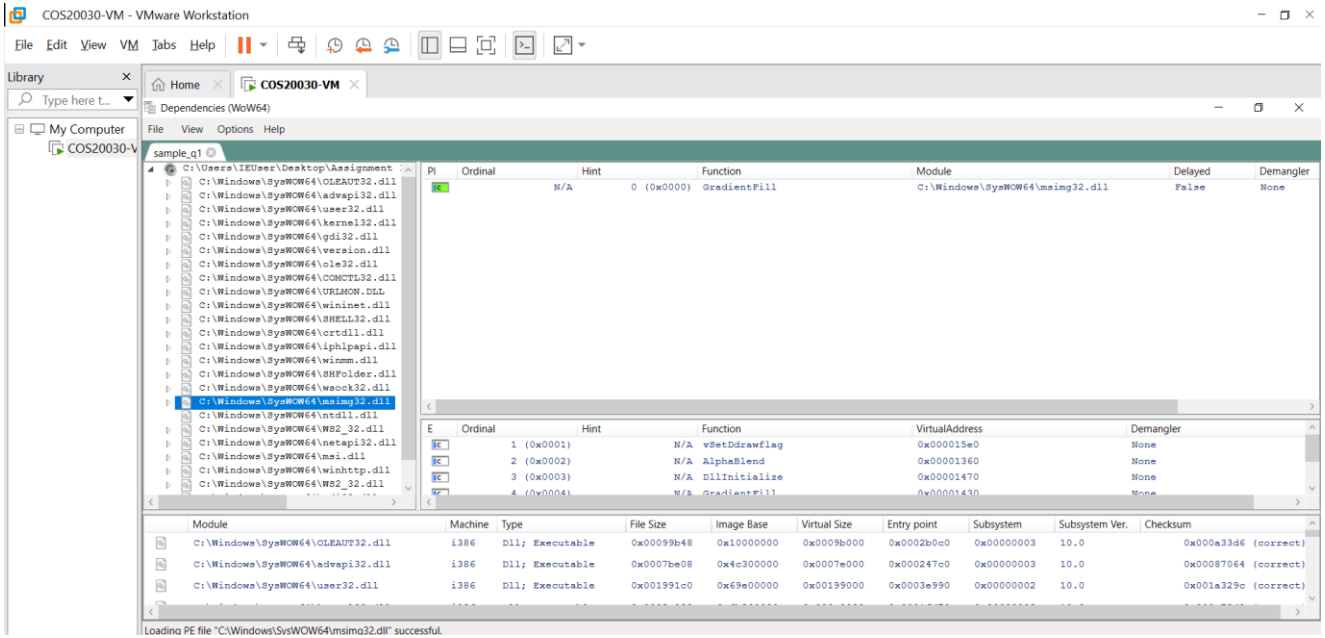


Figure 43: msimg32.dll and its imported functions

18. ntdll.dll: RtlZeroMemory

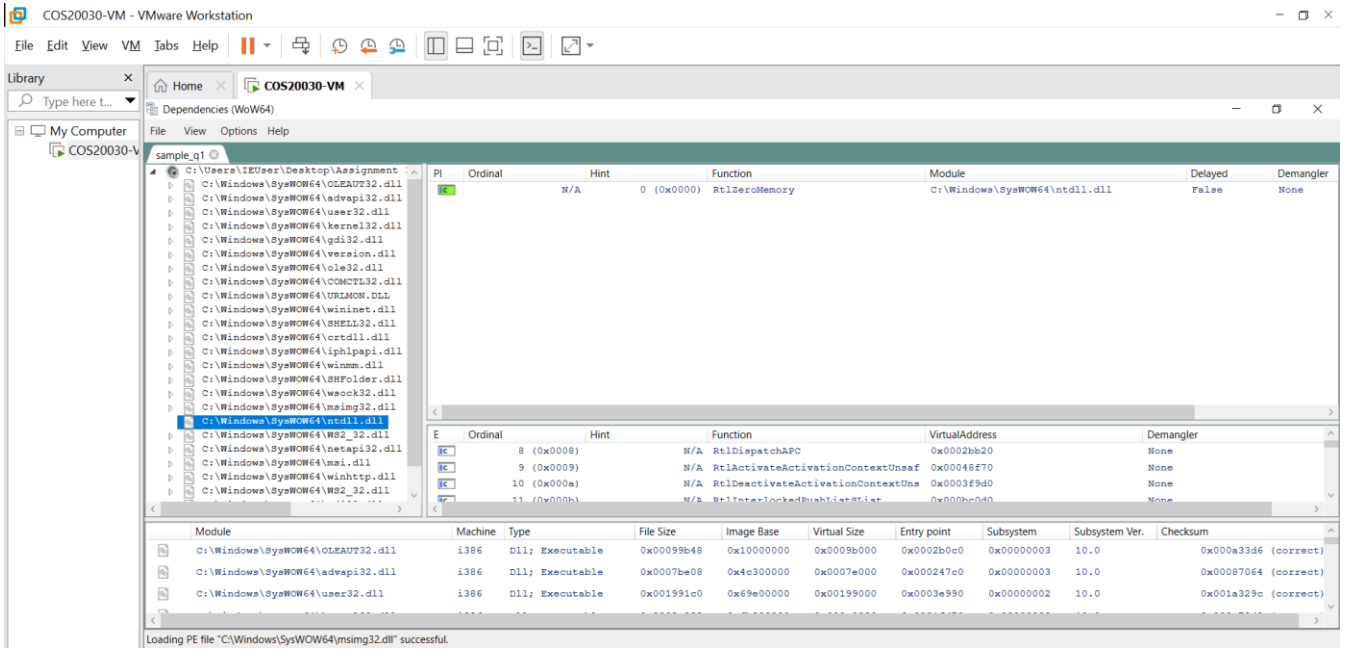


Figure 44: ntdll.dll and its imported functions

19. WS2_32.dll: closesocket, setsockopt

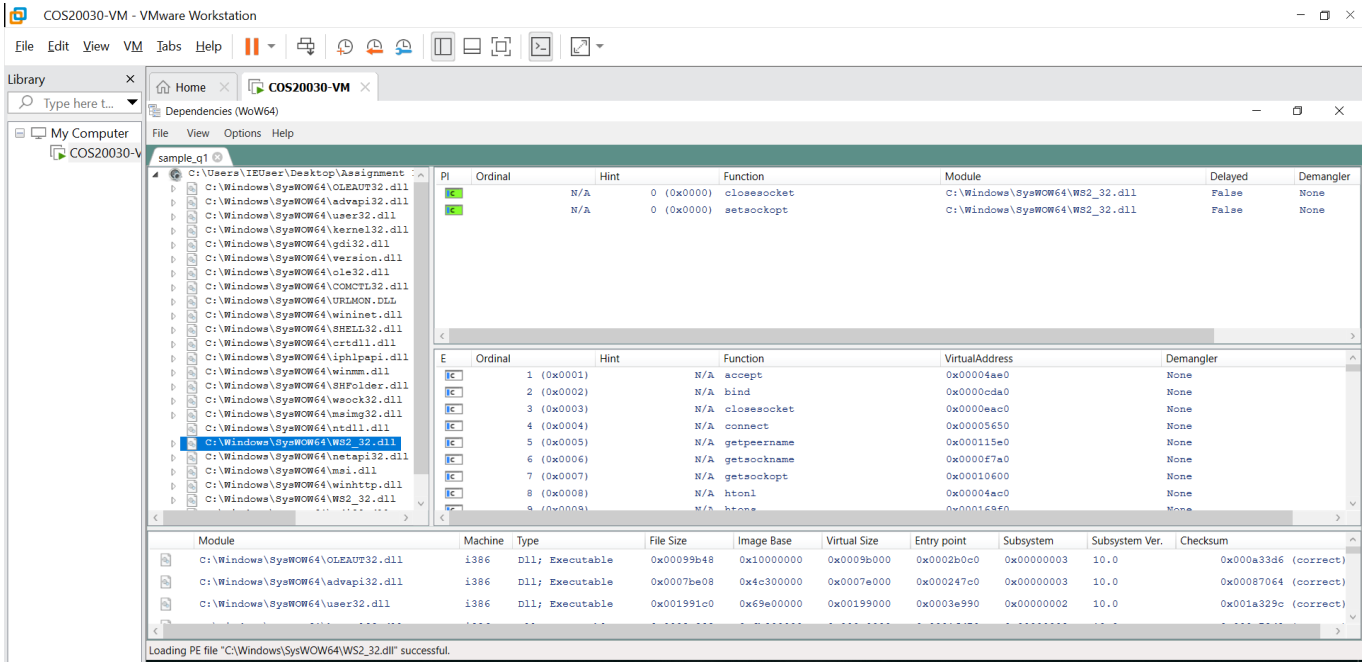


Figure 45: WS2_32.dll and its imported functions

20. netapi32.dll: NetUserGetInfo, NetApiBufferFree

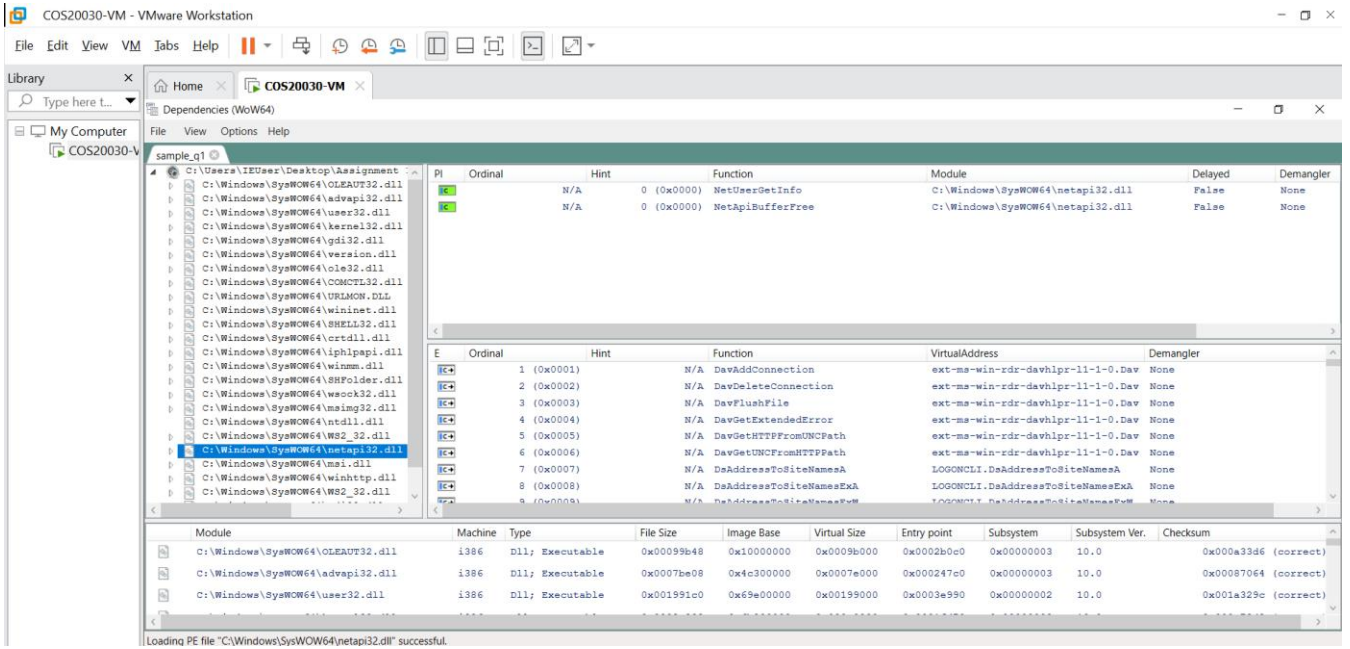


Figure 46: netapi32.dll and its imported functions

21. msi.dll: MsiEnumProductsA, MsiGetProductInfoA

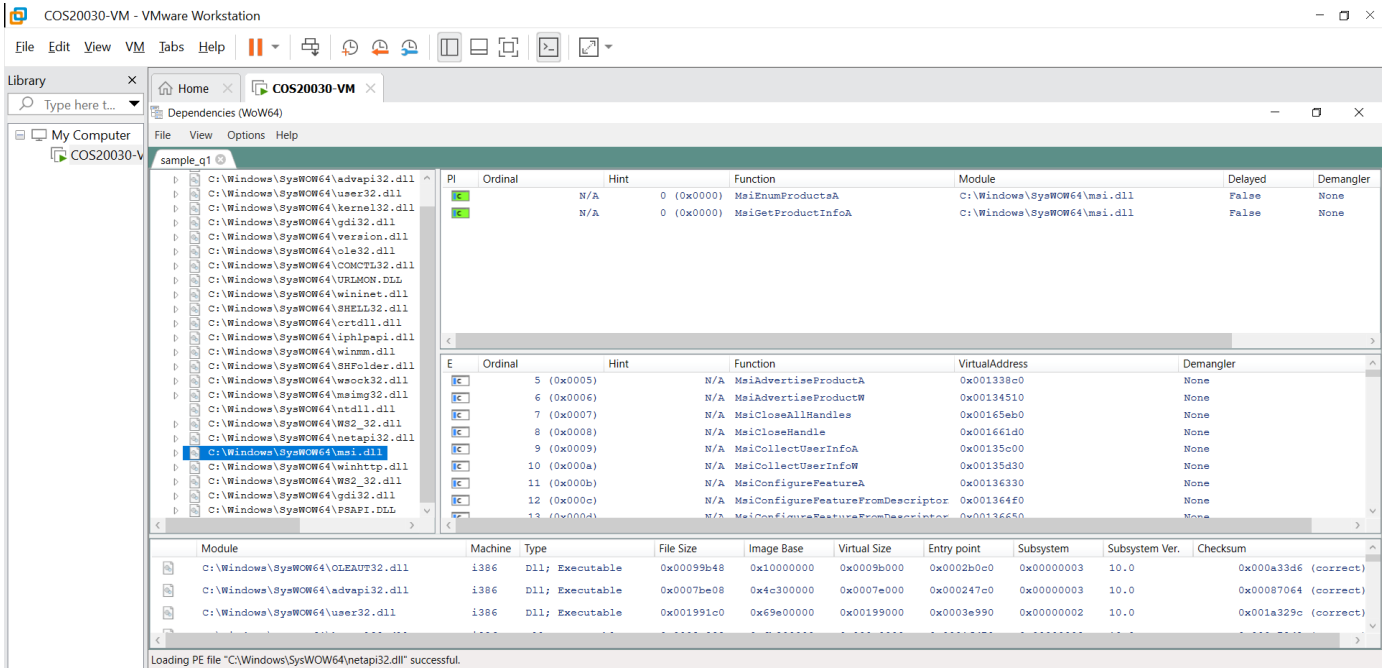


Figure 47: msi.dll and its imported functions

22. winhttp.dll: WinHttpSetStatusCallback, WinHttpCloseHandle, WinHttpReadData, WinHttpQueryDataAvailable, WinHttpQueryHeaders, WinHttpReceiveResponse, WinHttpWriteData, WinHttpSendRequest, WinHttpOpenRequest, WinHttpConnect, WinHttpOpen, WinHttpCrackUrl, WinHttpAddRequestHeaders, WinHttpSetOption

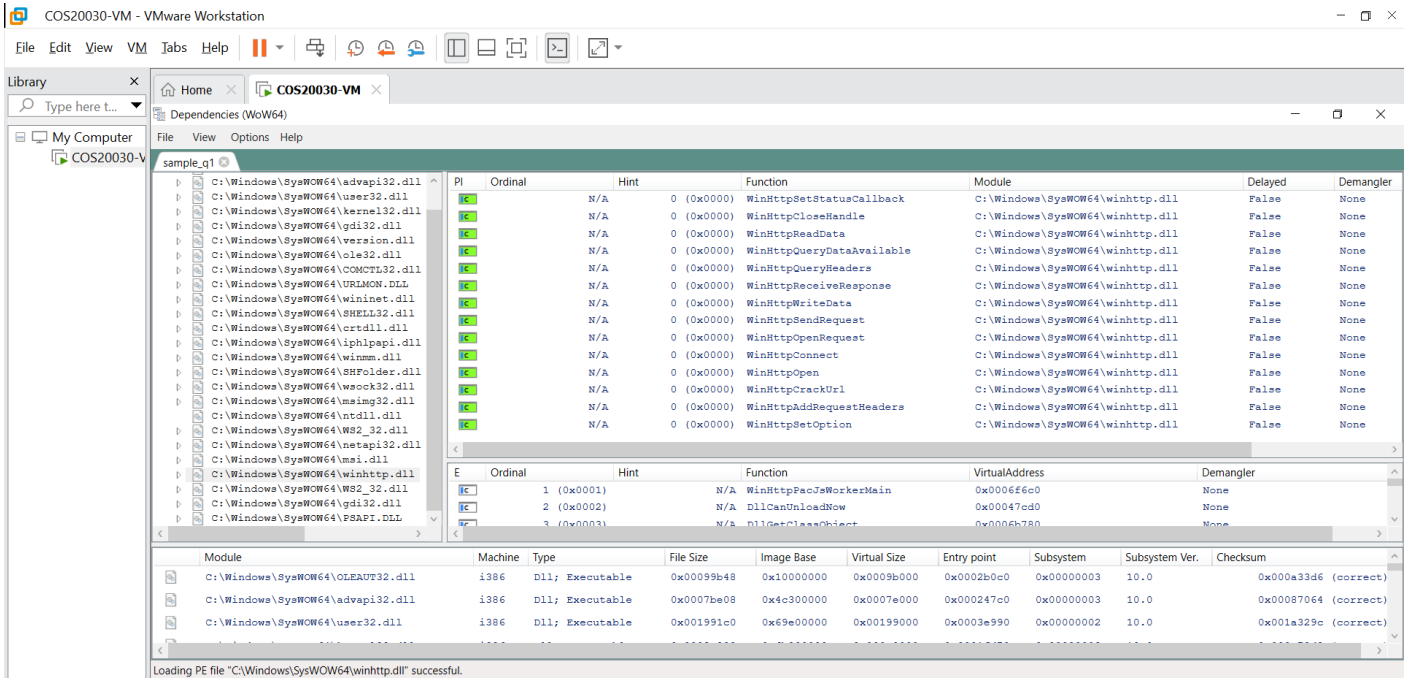


Figure 48: winhttp.dll and its imported functions

23. WS2_32.dll: WSACleanup, WSAStartup, htons, inet_addr, setsockopt, sendto, socket, closesocket

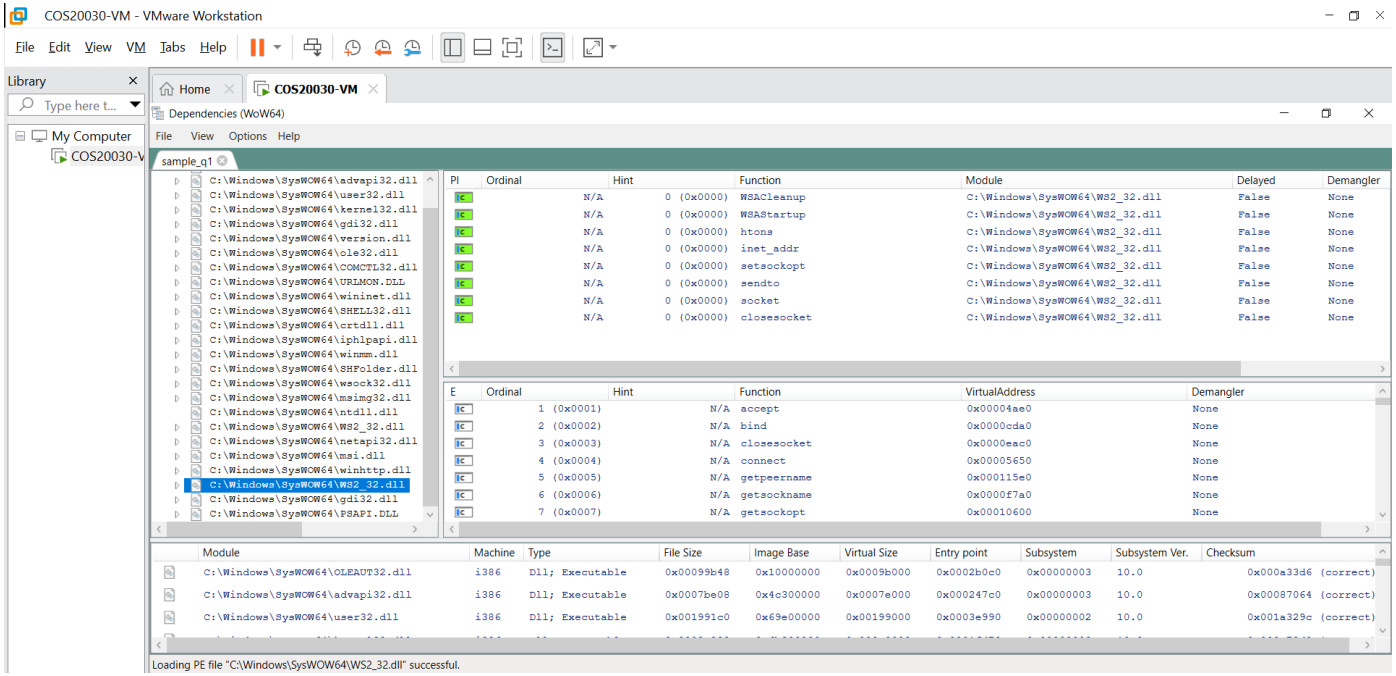


Figure 49: WS2_32.dll and its imported functions

24. gdi32.dll: GetRandomRgn

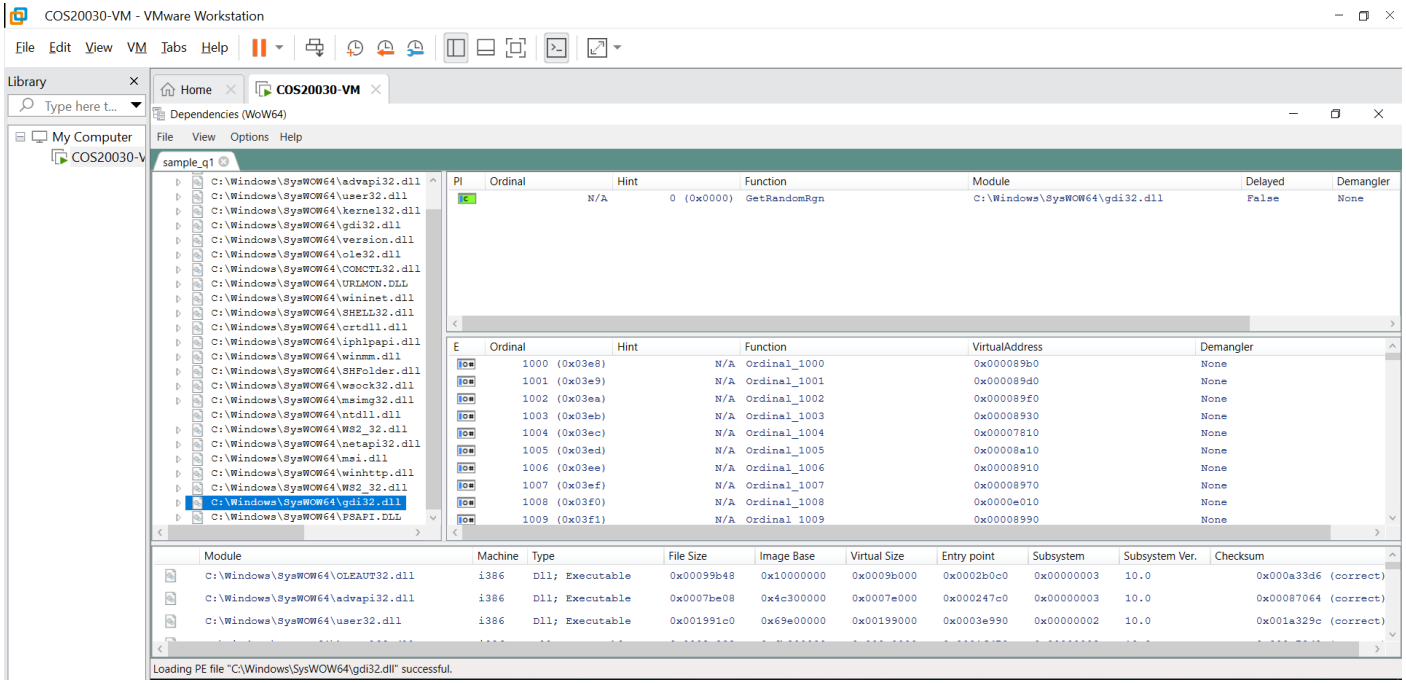
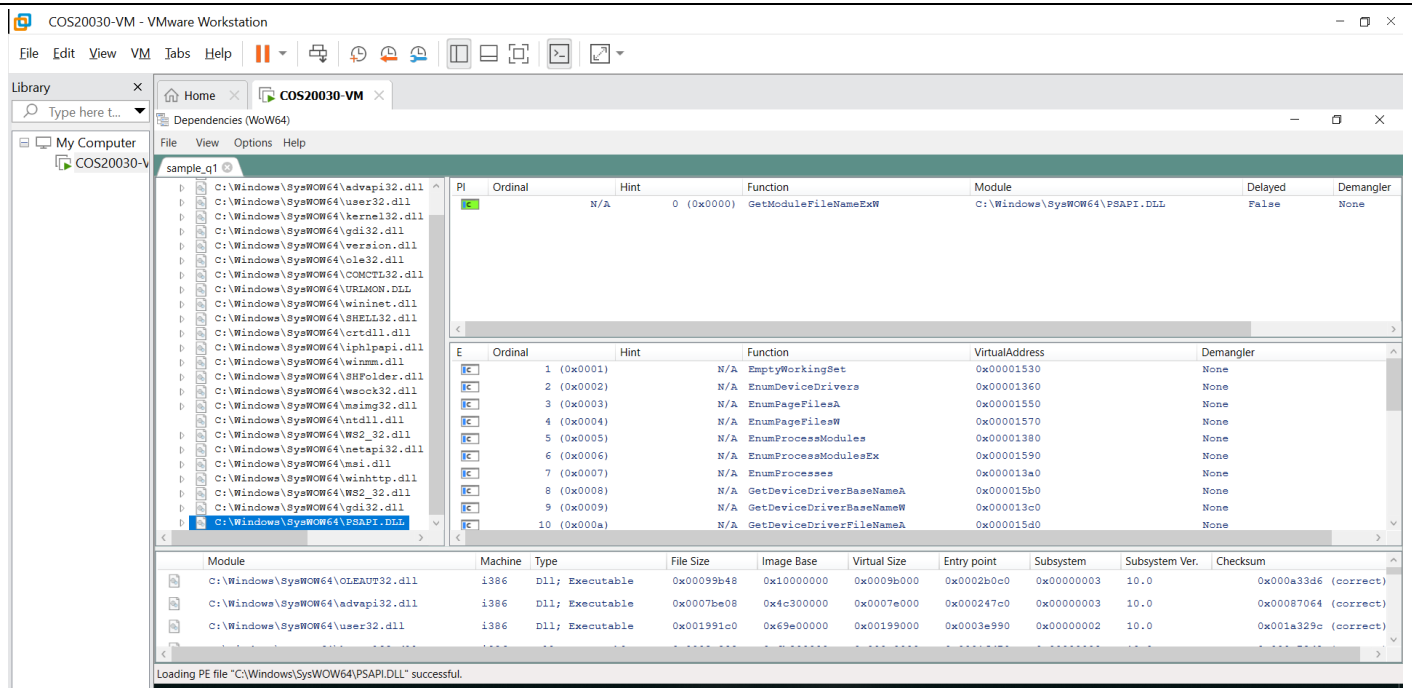


Figure 50: gdi32.dll and its imported functions

25. PSAPI.DLL: GetModuleFileNameExW

	 *Figure 51: PSAPI.DLL and its imported functions*	
Host-based indicators	Process Explorer shows sample_q1.exe running as a process under user IEUser, launched from C:\Users\IEUser\Desktop\Assignment 1 executable files\sample_q1.exe. The Properties window indicates the binary disables key mitigations (DEP, ASLR, Stack Protection), which is typical of malicious or packed executables. Parent process: explorer.exe.	Process Explorer, Process Monitor

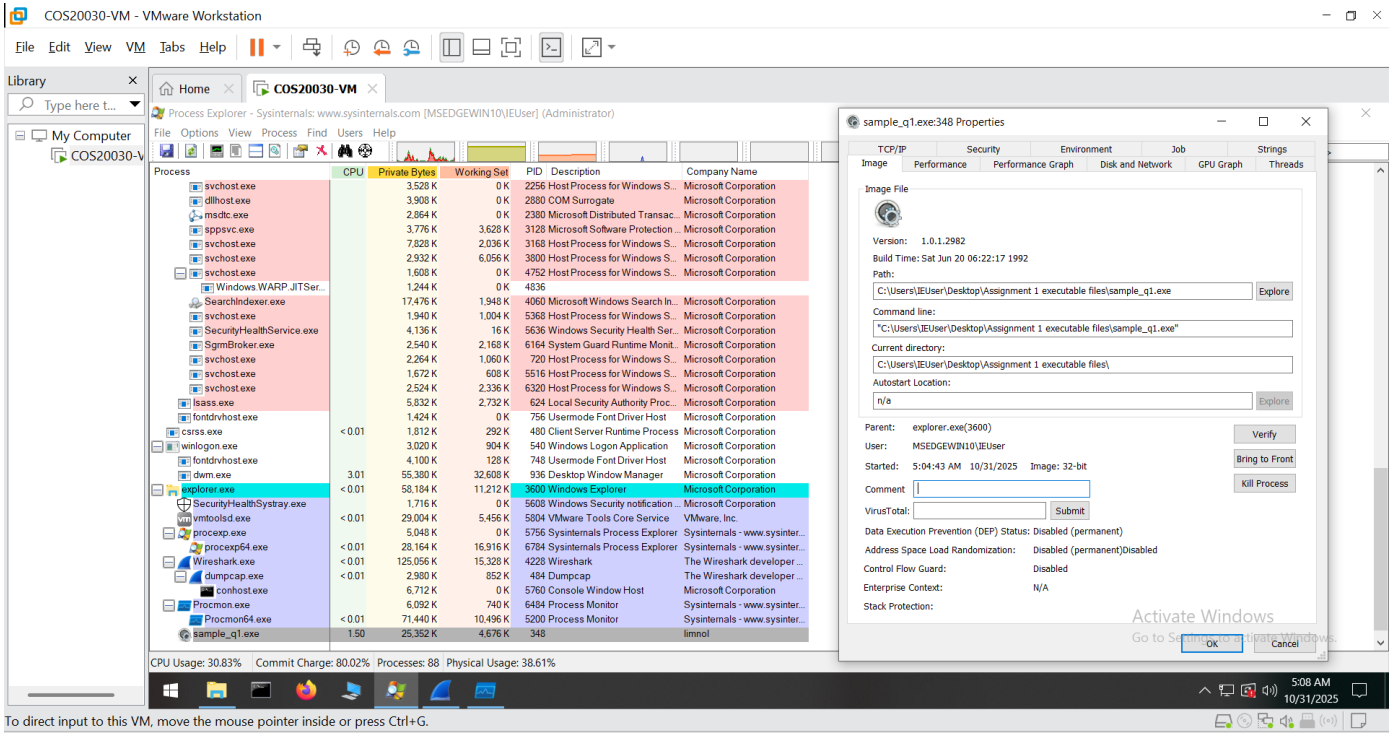


Figure 52: Process Explorer screenshot showing host-based indicators for sample_q1.exe in a VMware Workstation virtual machine, displaying CPU usage (e.g., 0.1%), memory consumption (e.g., 55,300 K), and running processes.

Process Monitor:

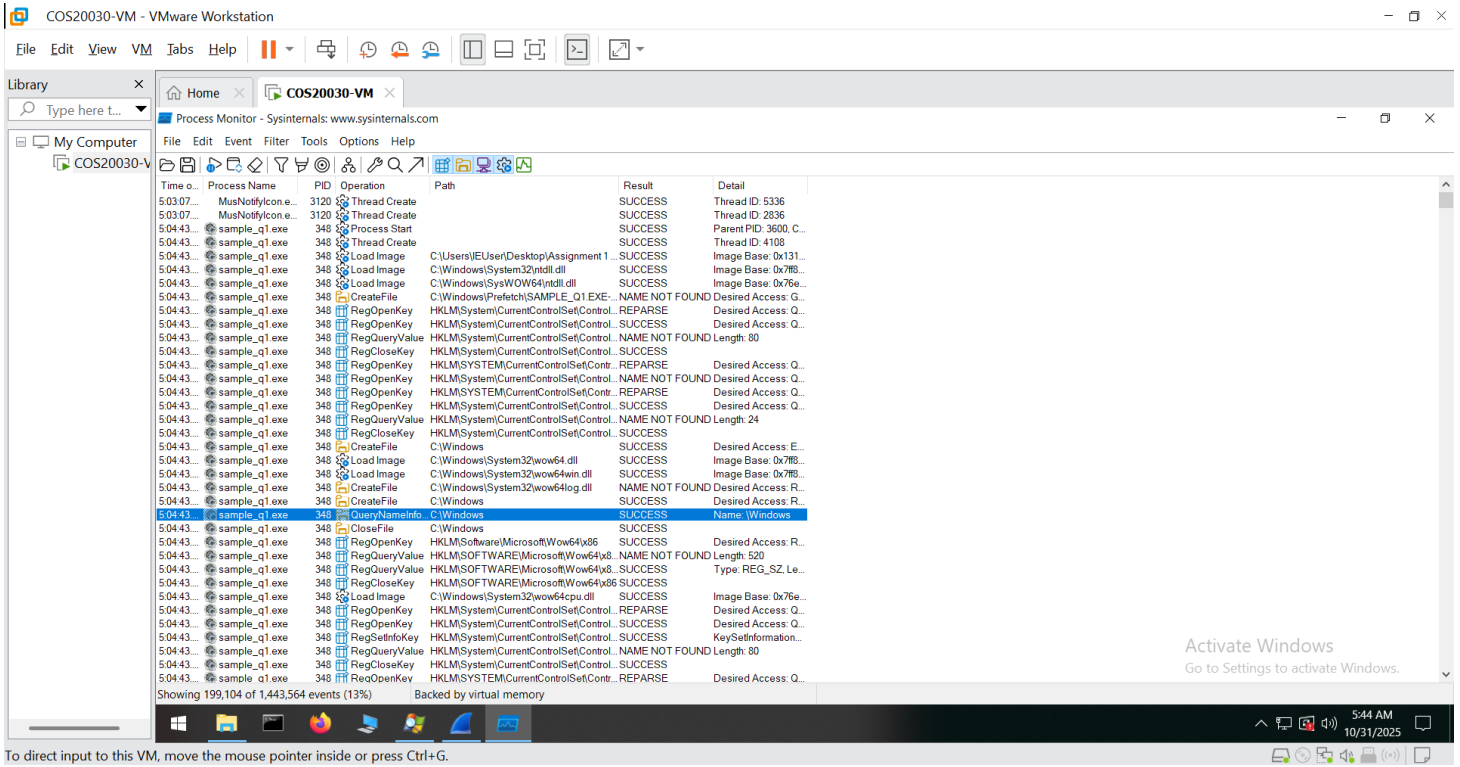


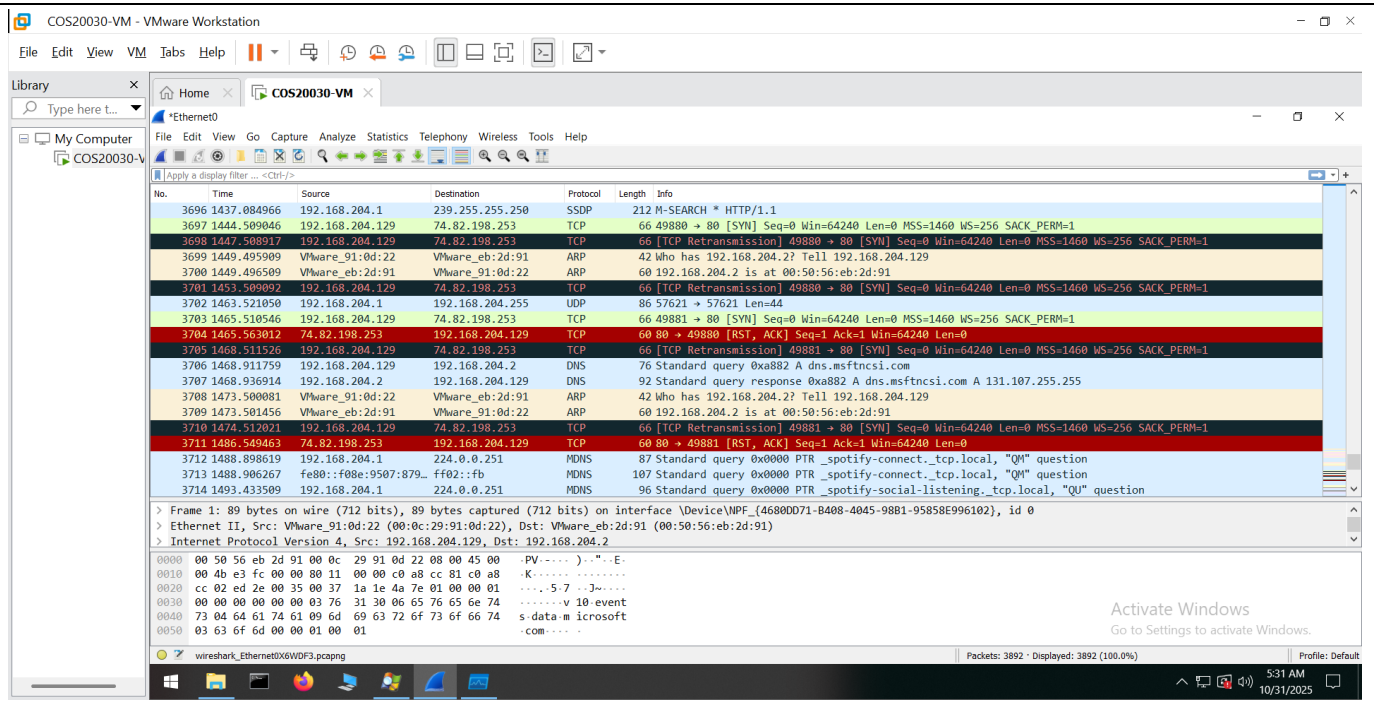
Figure 53: Dynamic analysis of sample_q1.exe using Process Monitor showing host-based activities such as file system access, registry modifications, and DLL loading within the Windows environment.

Observations:

- The process sample_q1.exe created multiple threads and initiated a process start event.
- It accessed files and directories under C:\Windows and C:\Users\...\Desktop.
- Several system DLLs were loaded, including ntdll.dll, wow64.dll, wow64win.dll, and wow64log.dll.
- Multiple registry operations were performed on keys under HKLM\Software\Microsoft\Wow6432Node and HKLM\System\CurrentControlSet\Control.
- Frequent RegOpenKey, RegQueryValue, and RegCloseKey actions were observed.

Explanations:

	• Thread creation and process start events suggest that the executable spawns background tasks or parallel processes, which can be used for persistence or stealthy execution. • Accessing system directories such as C:\Windows may indicate attempts to modify or drop files in protected areas. • Loading system DLLs shows that the executable interacts closely with the Windows API, possibly performing low-level operations. • Registry access activities imply system information gathering or modification of configuration data for persistence. • Overall, these behaviors are consistent with potentially malicious programs that attempt to manipulate system components or establish persistence on the host.	
Network-based indicators	During dynamic network analysis using Wireshark, the infected host machine (192.168.204.129) was observed attempting outbound TCP communication to an external public IP address (74.82.198.253) over port 80 (HTTP). The traffic shows repeated SYN and RST/ACK packets, indicating that the malware attempted to establish a connection but the remote server did not respond or actively refused the connection. This suggests the executable may be trying to contact a Command-and-Control (C2) server or download additional payloads. Additionally, DNS, MDNS, and NBNS traffic was observed, including queries for local services, which may indicate the malware is also performing network discovery to identify devices or services on the local network. Overall, these network behaviors are suspicious and support the possibility of malicious external communication attempts.	Wireshark

	 *Figure 54: Wireshark screenshot showing network-based indicators for sample_q1.exe in a VMware Workstation virtual machine, capturing TCP and UDP traffic with source and destination IP addresses (e.g., 192.168.1.1 to 203.0.113.1), ports (e.g., 2121, 80), and protocols.*	
Any other findings (if applicable)	**Wireshark:**	Wireshark

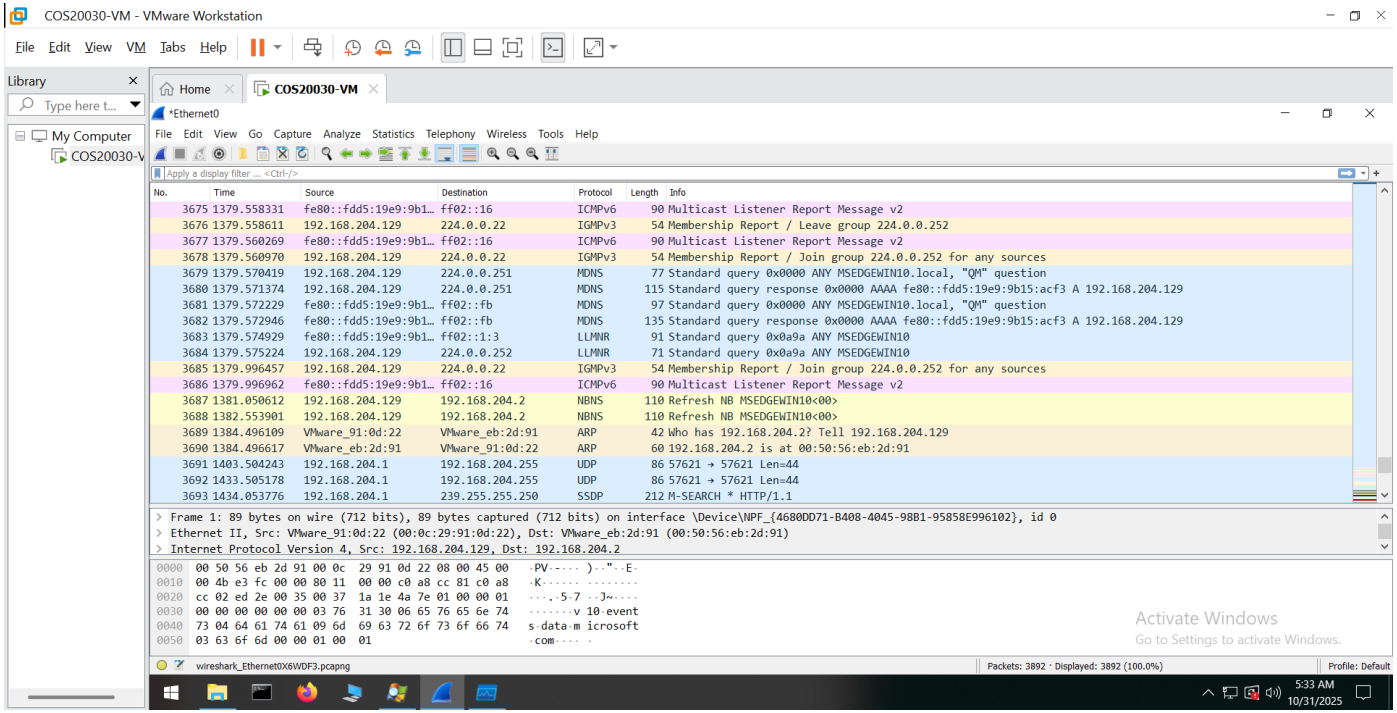


Figure 55: Wireshark capture showing local network broadcast and multicast packets (ICMPv6, NBNS, SSDP) possibly triggered by the execution of sample_q1.exe.

Observations:

- Network traffic captured in Wireshark shows multiple ICMPv6, IGMPv3, MDNS, NBNS, ARP, and SSDP packets.
- The source and destination IP addresses indicate local broadcast and multicast communication within the virtual network.
- Protocols such as NBNS (NetBIOS Name Service) and SSDP (Simple Service Discovery Protocol) were detected, which are typically used for device or service discovery on local networks.
- Some HTTP-related SSDP packets include strings such as "M-SEARCH" and "Microsoft.com", suggesting an attempt to discover or communicate with Windows services.

Explanations:

	• These findings indicate normal system background network activity within the VM environment, rather than direct malicious behavior. Windows and VMware often generate multicast and name resolution traffic automatically. • However, if this network activity was initiated immediately after running sample_q1.exe, it may suggest that the executable triggered service discovery requests or attempted to communicate over the local network — a possible sign of network scanning or reconnaissance behavior. • The presence of SSDP and NBNS queries could indicate that the file is probing for available devices or services, which is common for malware that attempts to spread laterally or report system details to other hosts.	
Potential purpose of these files	Based on the combined static and dynamic analysis, sample_q1.exe exhibits multiple indicators suggesting that it is a potentially malicious or Trojan-type executable designed for system compromise, persistence, and remote communication. • The presence of network-related DLL imports indicates that the executable is capable of performing HTTP and socket-based communication, likely for connecting to remote servers or Command-and-Control (C2) infrastructure. • The frequent registry modification activities suggest that the malware may attempt to modify system configurations or create persistence mechanisms by altering startup or service-related registry keys. • The thread creation and interaction with system directories under *C:\Windows* imply that it performs background operations or possibly drops additional payloads in protected areas of the operating system. • The disabled security mitigations (DEP, ASLR, Stack Protection) further reinforce that the executable is intentionally crafted to evade security features and may perform code injection or memory manipulation. • The observed outbound network connection attempts to external IPs and local discovery traffic (NBNS, SSDP) indicate an intent to communicate externally and possibly enumerate network devices, aligning with data exfiltration or reconnaissance behavior. Overall, the purpose of sample_q1.exe appears to be to establish control over the infected system, communicate with external servers, and potentially download, execute, or spread additional malicious components. Its characteristics are consistent with a backdoor or downloader-type malware designed to maintain remote access, evade detection, and gather information from the host environment.	

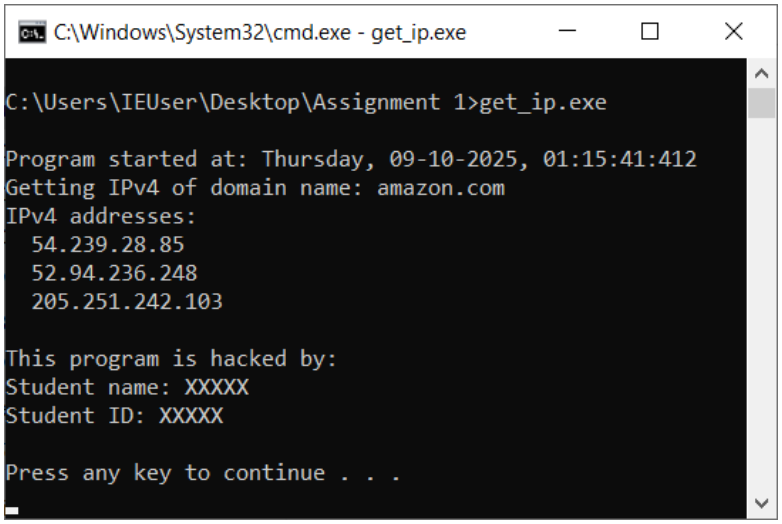
** Expand the table if necessary*

Question 2 (8 marks)

Note: *Please allow the VM to access the Internet.*

You are given an executable file named **get_ip.exe**. When executed in the command prompt, it queries a DNS server and returns the IPv4 address of Amazon (**amazon.com**).

Example of the program output is shown below.



a) Using the patching/hacking techniques covered in labs and lectures, modify the given executable to achieve the following:

   i) Replace the domain name **amazon.com** with **swinburne.edu.my**, so that the program returns the IPv4 address of Swinburne Sarawak.

   ii) Replace the text marked with **XXXXX** in the executable with your full name and student ID.

   Take screenshots of your patching steps and explain how you performed patching/hacking on the executable.

(6 marks)

1. Verify Original Behaviour:
   Run: get_ip.exe

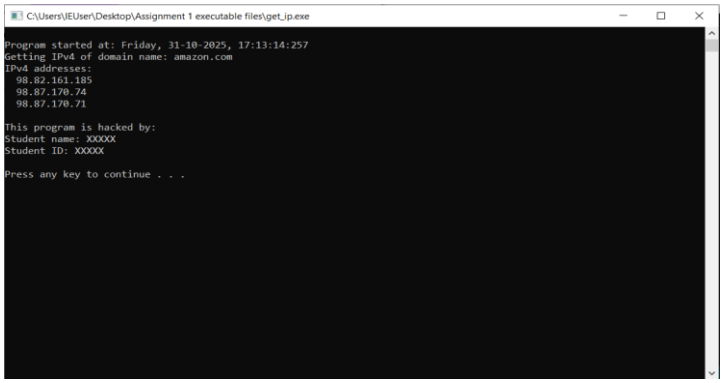


Figure 56: Original output – amazon.com

2. Load & Search for the Domain String

a. Open file get_ip.exe in OllyDbg.

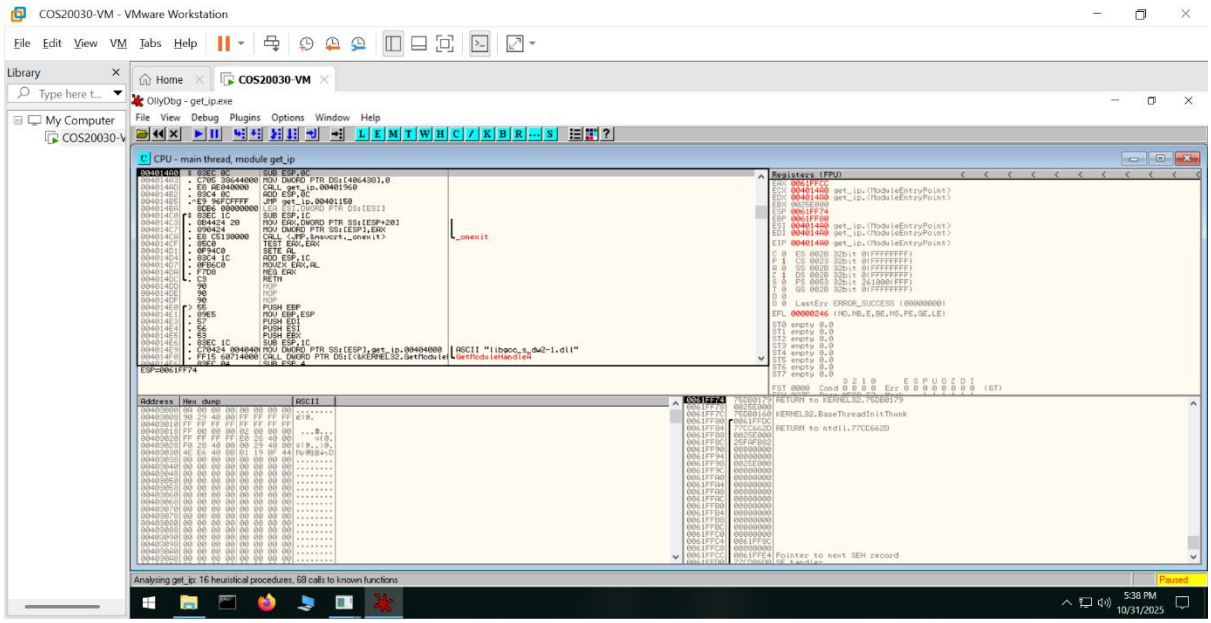


Figure 57: Loading get_ip.exe in OllyDbg

b. Right-click Assembly pane then click Search for then choose All referenced text strings. New Text strings window opens. Right-click inside it and click Search for text.

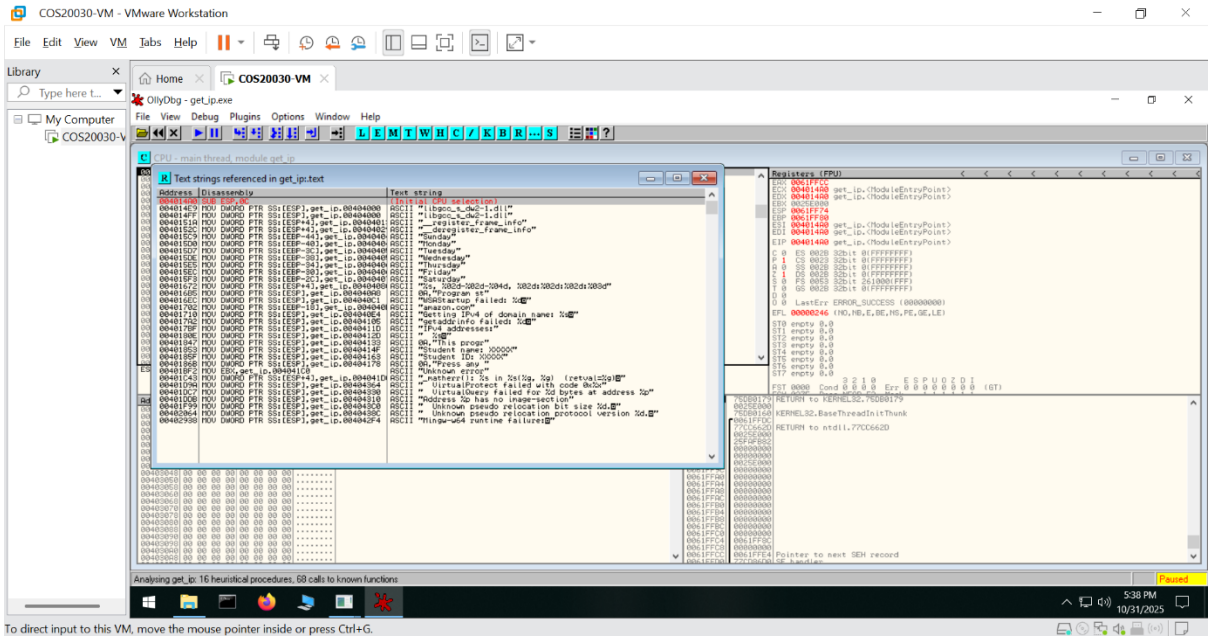


Figure 58: Searching All Referenced Text Strings

c. Type amazon.com and tick Case sensitive & Entire scope then click OK button.

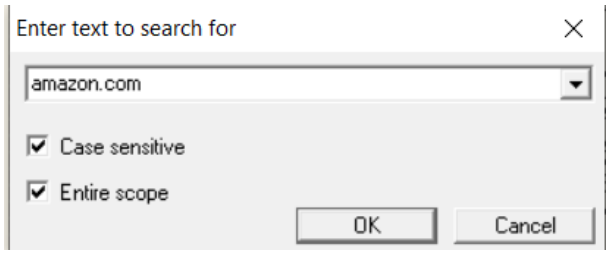


Figure 59: Searching for the Domain "amazon.com"

d. Now it land on the only reference. Double-click it to jump to CPU.

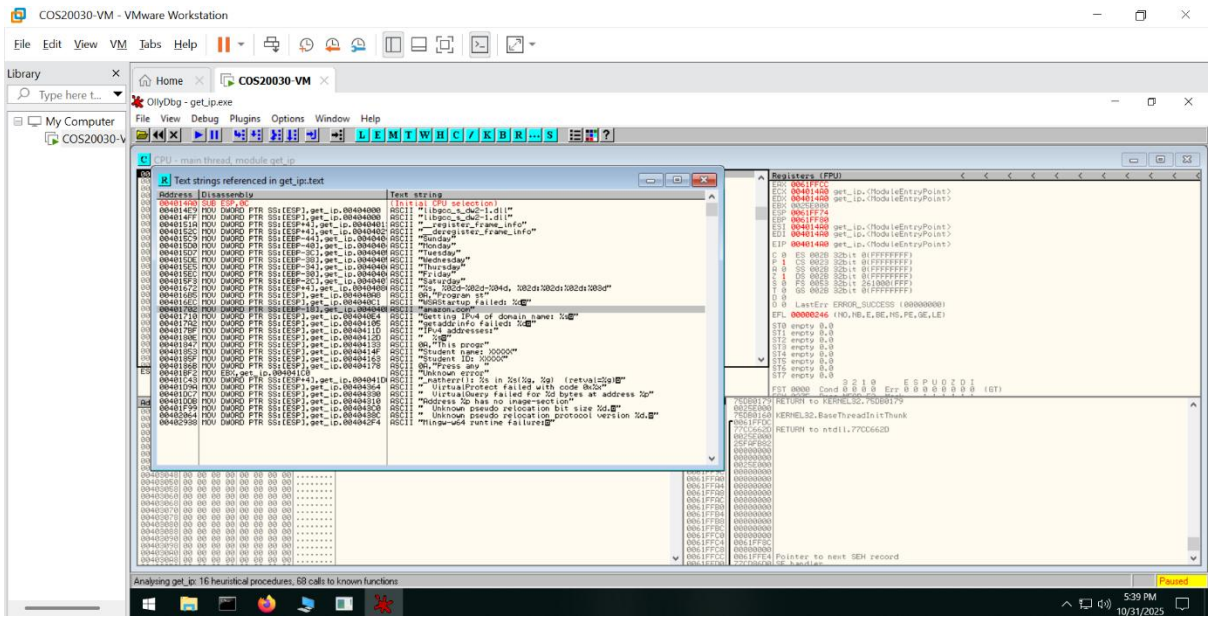


Figure 60: Locating the PUSH Instruction for "amazon.com"

e. OllyDbg CPU window showing the `PUSH` instruction that references the string `"amazon.com"` at memory address `004040D8`. This is the target for patching to `swinburne.edu.my`.

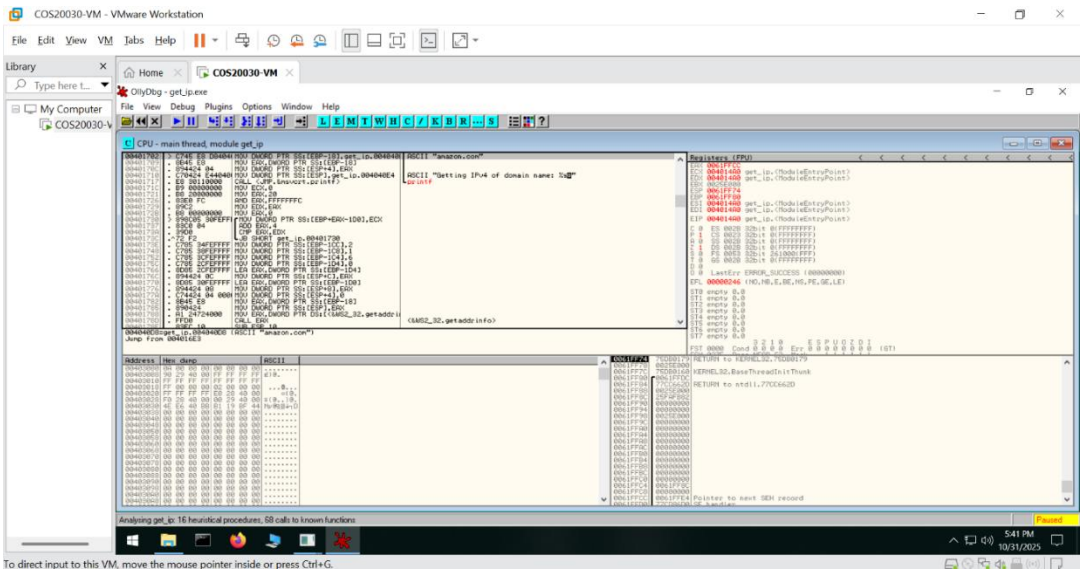


Figure 61: CPU pane with PUSH offset "amazon.com" highlighted

3. Right click on the CPU panel then click follow in dump suing selection, it will show the dump window, scroll down find empty binary. Then highlight 2 empty line then press ctrl + e to edit the library. Type the swinburne.edu.my in the ASCII. Then click OK.

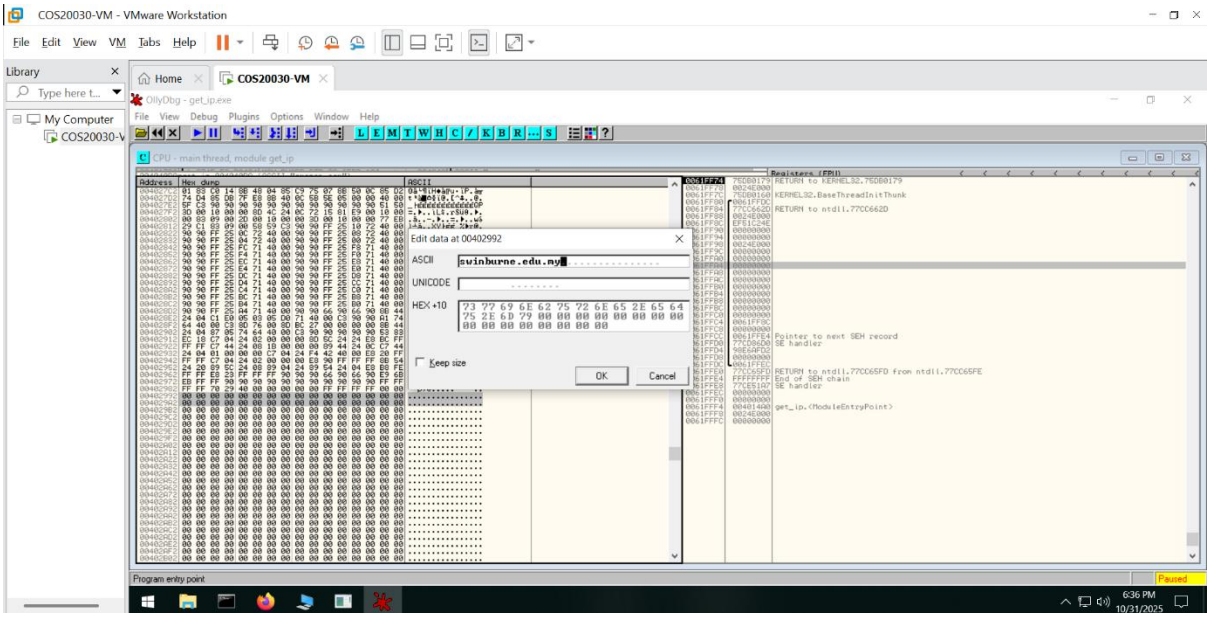


Figure 62: Binary editing the blank binary to "swinburne.edu.my"

4. Now it change to the swinburne.edu.my with the corresponding binary. Noted down the address for this line which is 00402992.

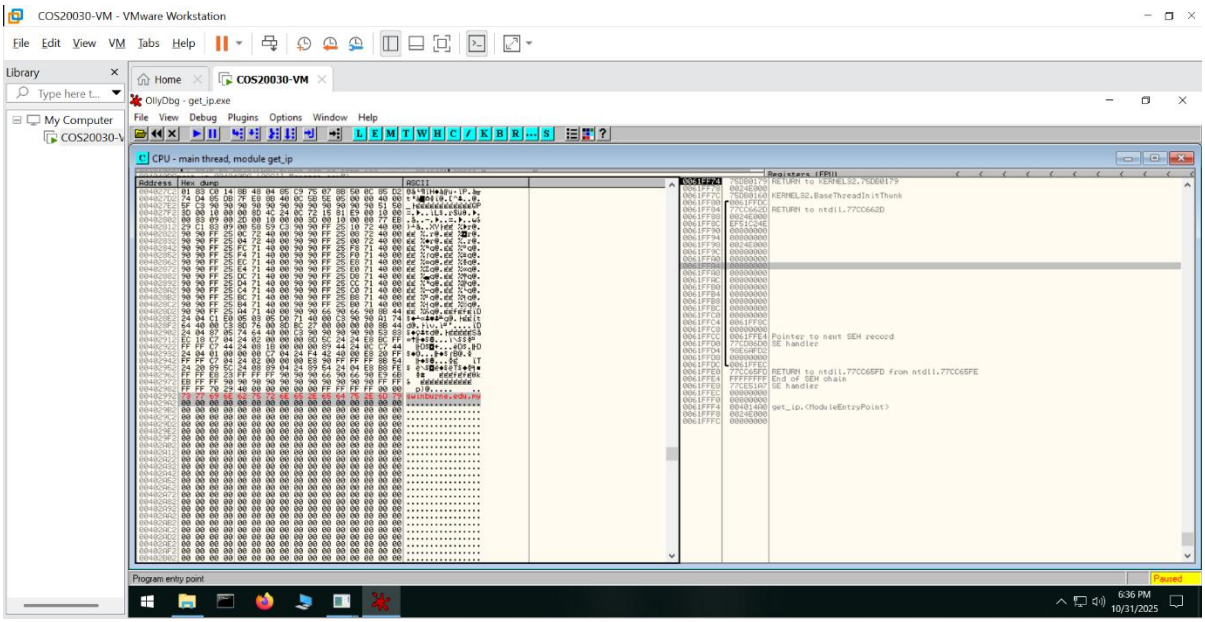


Figure 63: Dump after successfully patching the domain string

5. Leave one blank after the domain string after swinburne.edu.my because after the domain name it need at least 1 empty space for the assembly code to know it finished. After that, also highlight 2 empty line after one line blank then press ctrl + e. Type Student name: PHANG XIA HUI into the ASCII box.

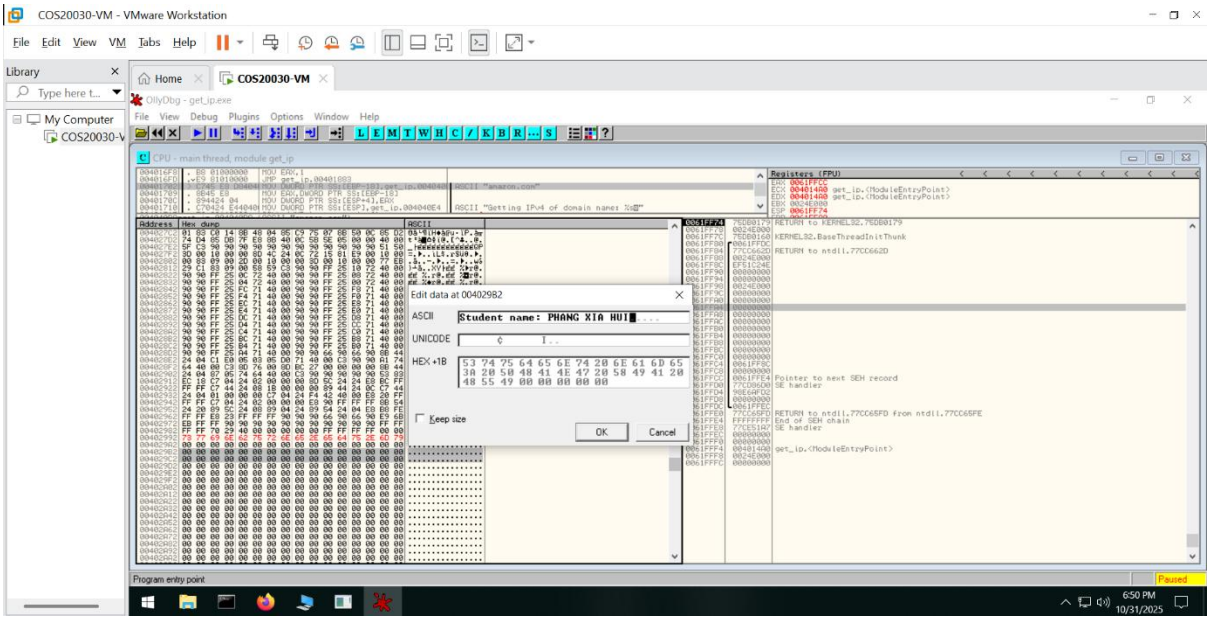


Figure 64: Binary editing the blank ASCII to "Student name: PHANG XIA HUI"

6. Now highlight another two line after this then press ctrl + e. Type Student ID: 102773508 into the ASCII box.

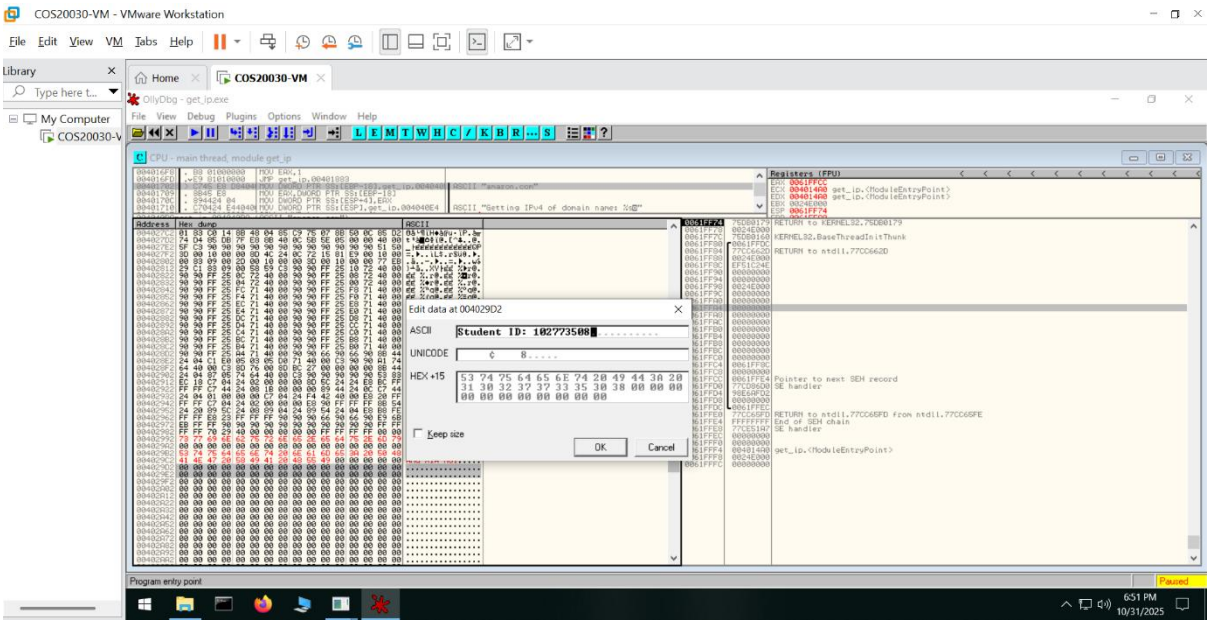
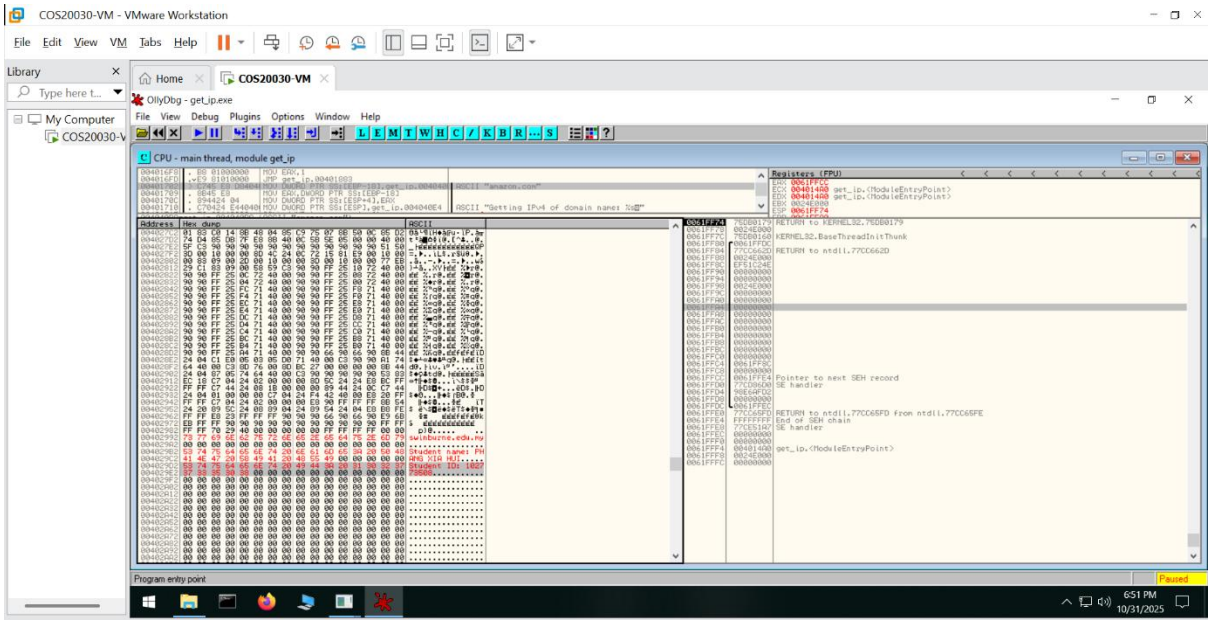


Figure 65: Binary editing the blank ASCII to "Student ID: 102773508"

7. Now, we can see those four lines change to the Student name: PHANG XIA HUI & Student ID: 102773508 with the corresponding binary. Noted down the address for this two lines which is 004029B2 and 004029D2.



9. Then go to the first address we go to amazon.com on the code section, click on it and then press spacebar, it will pop up the assemble pop up for us to change the code.

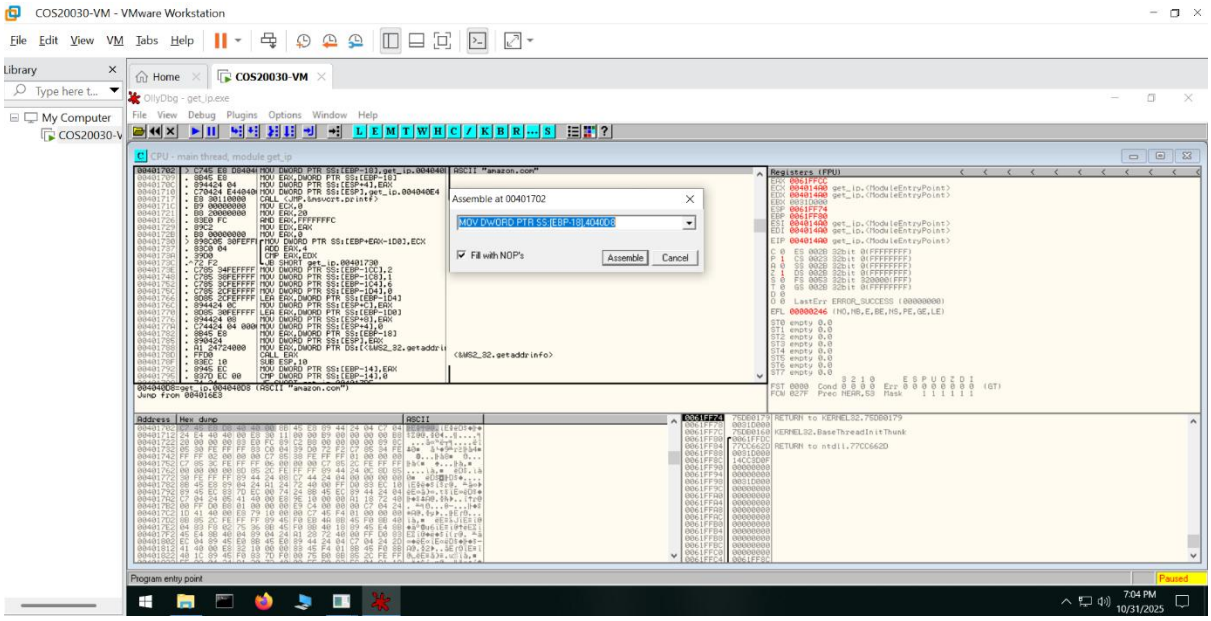


Figure 66: Assembly editing with NOP instructions

10. Change the original address 4040D8 to the address we noted for swinburne.edu.my which is 00402992.

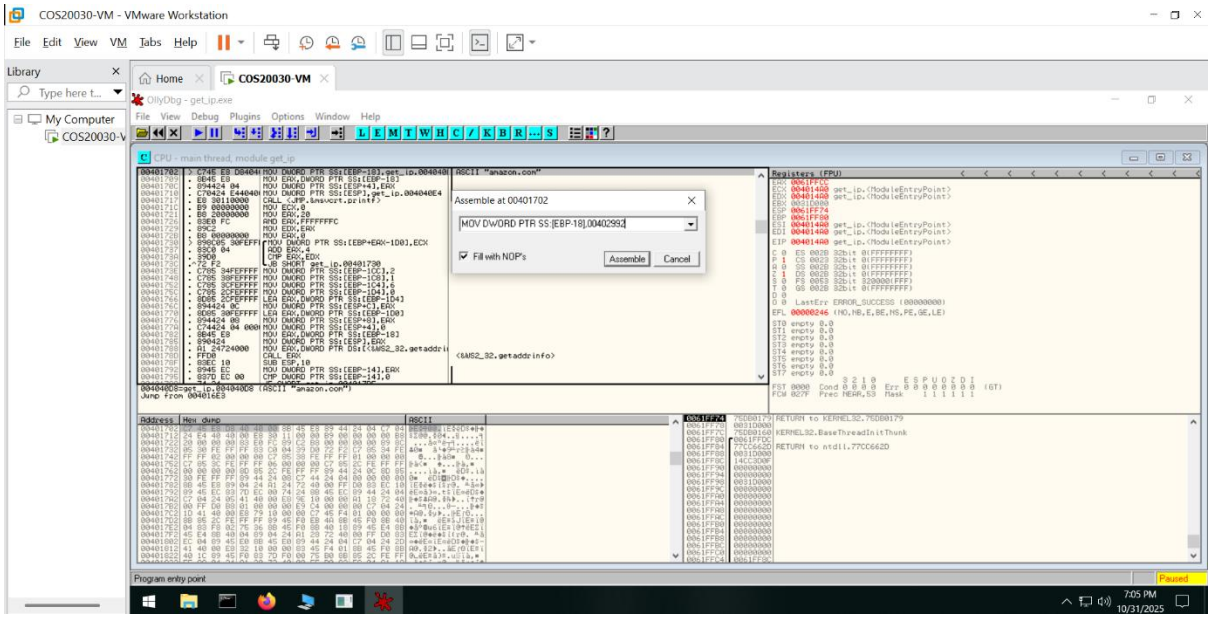


Figure 67: Address after edited

11. Click Assemble. Now, we can see the amazon.com on the code section change to swinburne.edu.my. It will replace the original domain string from amazon.com to swinburne.edu.my after edited the address.

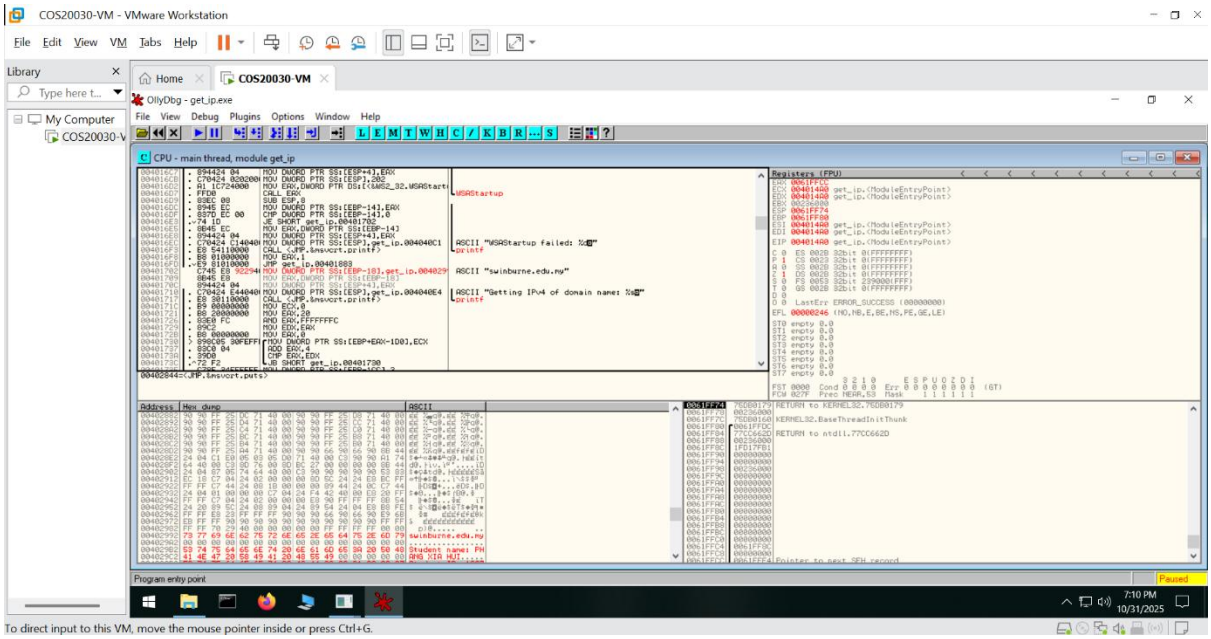


Figure 68: Domain change to Swinburne.edu.my

12. Repeat those step for the Student name and Student ID. First, find the part that for student id and student name. Then, click on it and then press spacebar, it will pop up the assemble pop up for us to change the code. Change the original address to the address we noted down as well which is 004029B2 and 004029D2.

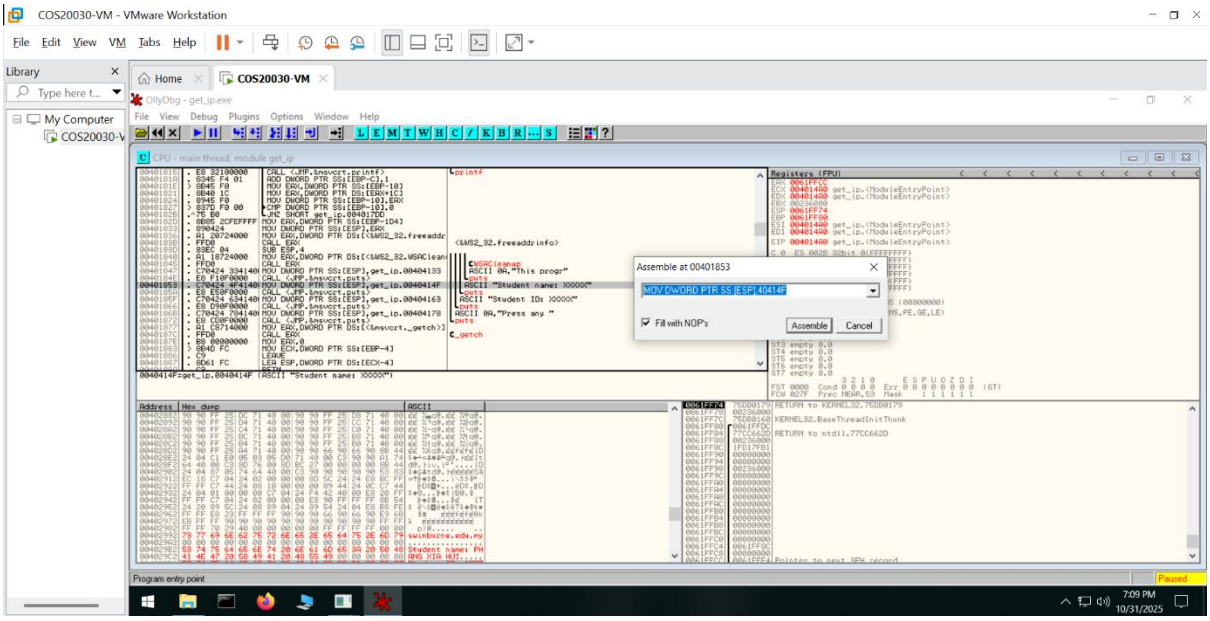
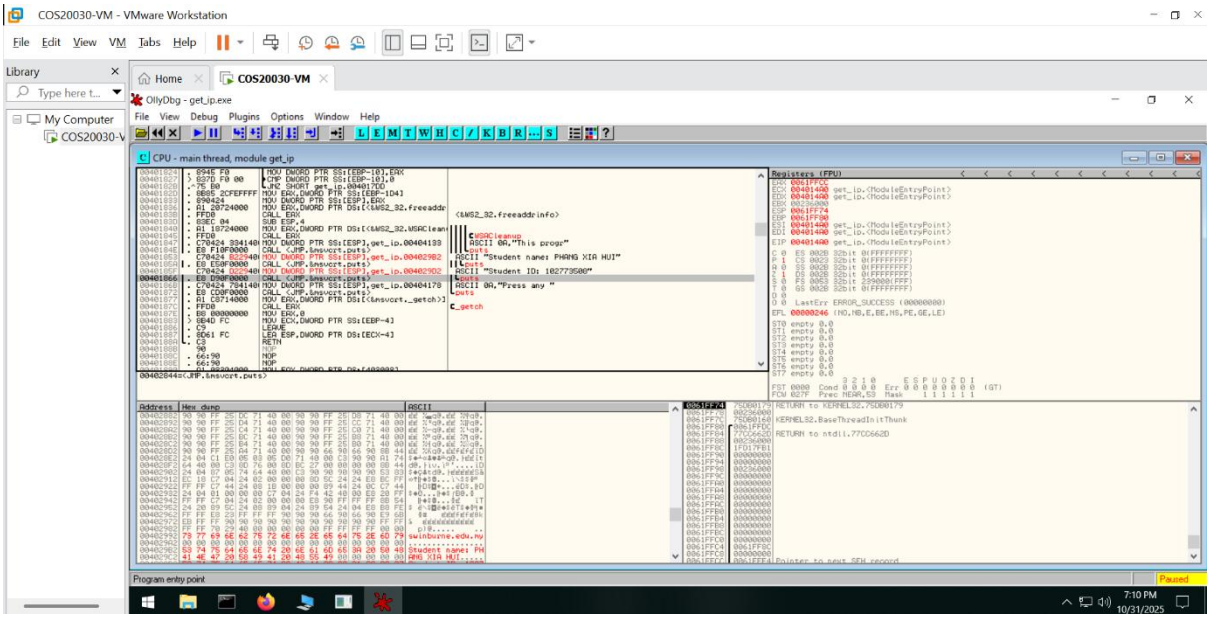


Figure 69: Assembly editing with NOP instructions

13. Then click Assemble, Now, we can see the original text change to Student name: PHANG XIA HUI and Student ID: 102773508.



14. Then right-click the code section then select “copy to executable” > “all modifications” then click on copy all on the pop up menu, it shows another pop up panel, right click that and select “save file”. Then save it as “get_ip_edited.exe”.

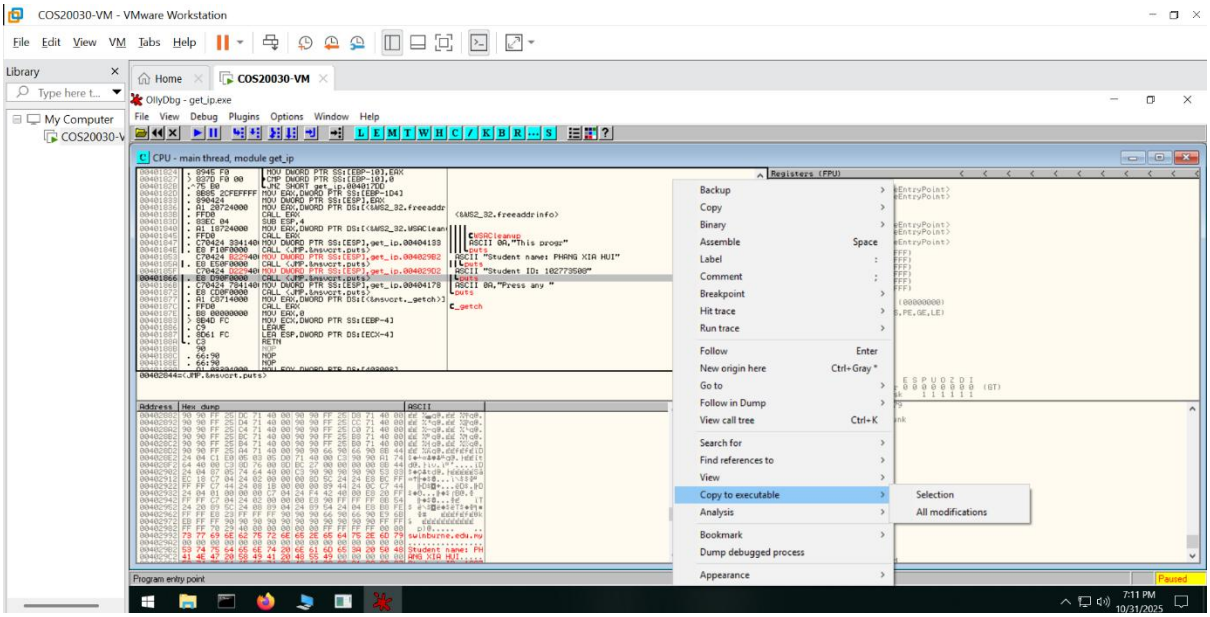


Figure 70: Saving file

15. Run the get_ip_edited.exe.

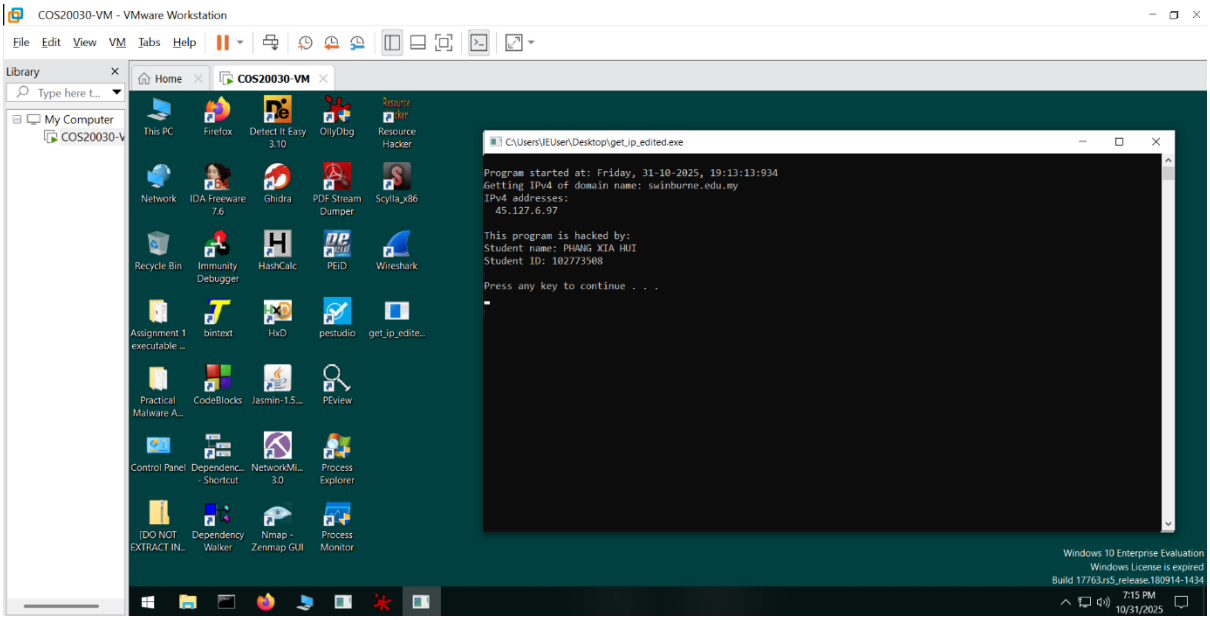
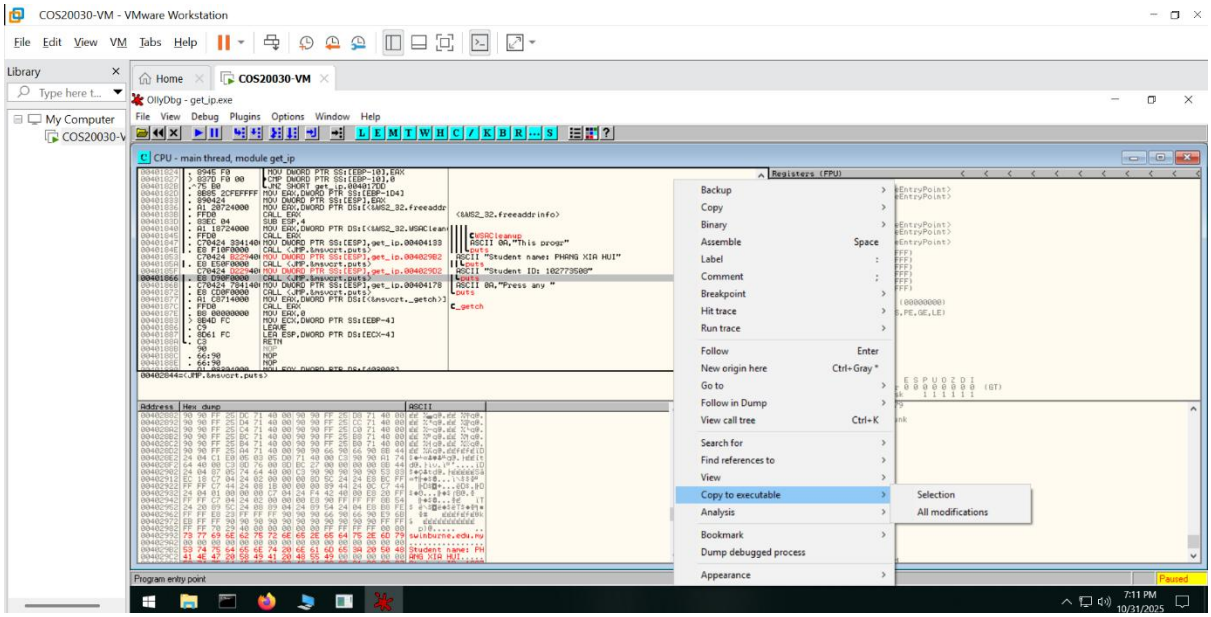


Figure 71: Output of get_ip_edited.exe

b) Save your modified executable as **get_ip_edited.exe**, then place it in a ZIP file.
Note: The ZIP file should contain only the modified executable.

(2 marks)

Right-click the code section then select “copy to executable” > “all modifications” then click on copy all on the pop up menu, it shows another pop up panel, right click that and select “save file”. Then save it as “get_ip_edited.exe”.



Marking Criteria

Criteria	Standards achieved		
Question 1	**Excellent** **12.0 – 9.0 pts** Demonstrated good competency in basic static and basic dynamic analysis techniques. Results are well-documented and supported with screenshots. Claims are supported with evidence and reasonable justifications. Demonstrates good investigative skills.	**Meets Expectation** **8.5 – 6.0 pts** Demonstrated baseline competency in basic static and basic dynamic analysis techniques. Results are documented and supported with screenshots. May occasionally make claims or conclusions without proper evidence.	**Needs Improvement** **5.5 – 0.0 pts** Lacking competency in basic static and basic dynamic analysis techniques. Provided simplistic or incomplete results. Minimal interpretation of results.
Penalty for Question 1 responses	**3.0 – 0.0 pts deduction** Written answers resembling Generative AI responses that are generic, out-of-context or unnatural. Instructor can apply penalty of not more than 3 pts.		

Question 2a	**Excellent** **6.0 – 4.5 pts** Demonstrated strong competency in applying patching or binary modification techniques. Steps are clearly shown with well-annotated screenshots and accurate explanations.	**Meets Expectation** **4.0 – 2.5 pts** Demonstrated basic competency in applying patching or binary modification techniques. Most steps are shown with reasonable explanations and some minor gaps or errors.	**Needs Improvement** **2.0 – 0.0 pts** Lacking competency in patching or binary modification. Evidence of work is incomplete, unclear, or technically inaccurate. Explanations show limited understanding.
Question 2b	**Good** **2.0 pts** The modified executable functions as intended.	**Average** **1.0 pts** The modified executable works partially or with minor issues.	**Poor** **0.0 pts** The modified executable does not function as intended.