CP-15

Malware Analysis Using Reverse Engineering

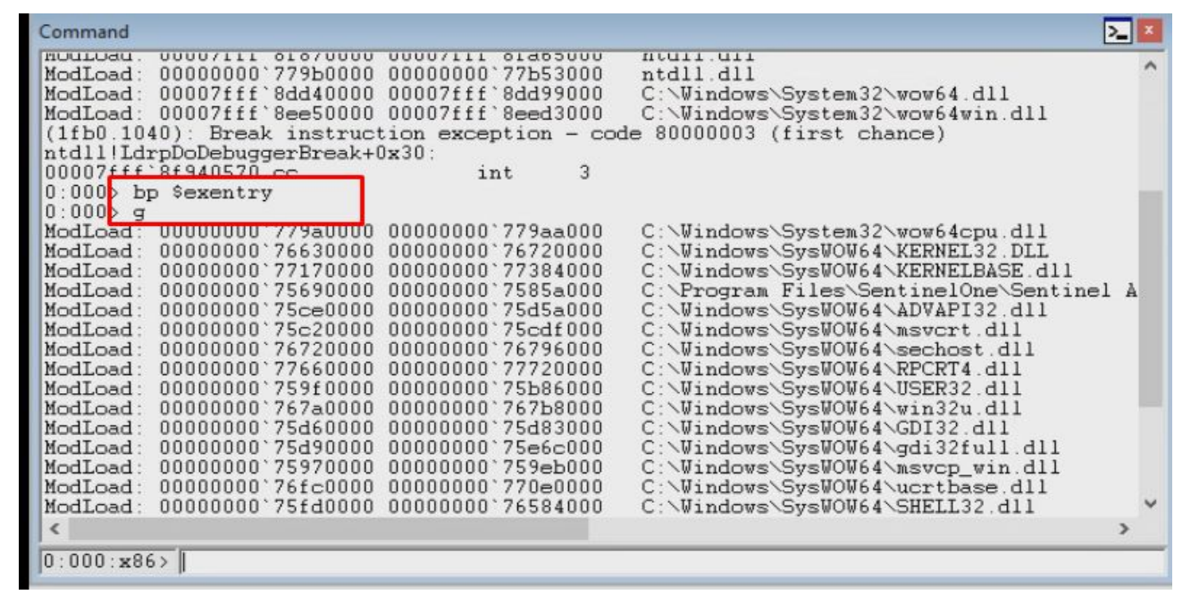
INTRO/ABSTRACT

This project aims to analyze malware applications using open source tools such as IDA Pro and WinDbg to identify malicious activities by reverse-engineering binary and source code. We've researched best methods and mitigation strategies for malware prevention and have included supporting documentation for other related areas such as malware recovery, training manuals, and an employee quiz. We created a secure and safe test environment for malware reverse engineering, pulled real malware samples, and put our selected tools to the test. We then found that IDA Pro provides a better UX and far lower learning curve than WinDbg, and has the added benefit of being a decompiler, whereas WinDbg is only a debugger.

METHODS

WinDbg: Some of the methods that were used to debug the malware were setting multiple breakpoints at certain addresses to inspect the code & see what it was doing, getting information about the memory address by doing !address and stopping it before it gets returned and then saving it to a dump file by using the info that was given from !address so that we could further review it and see what the code did when executing.

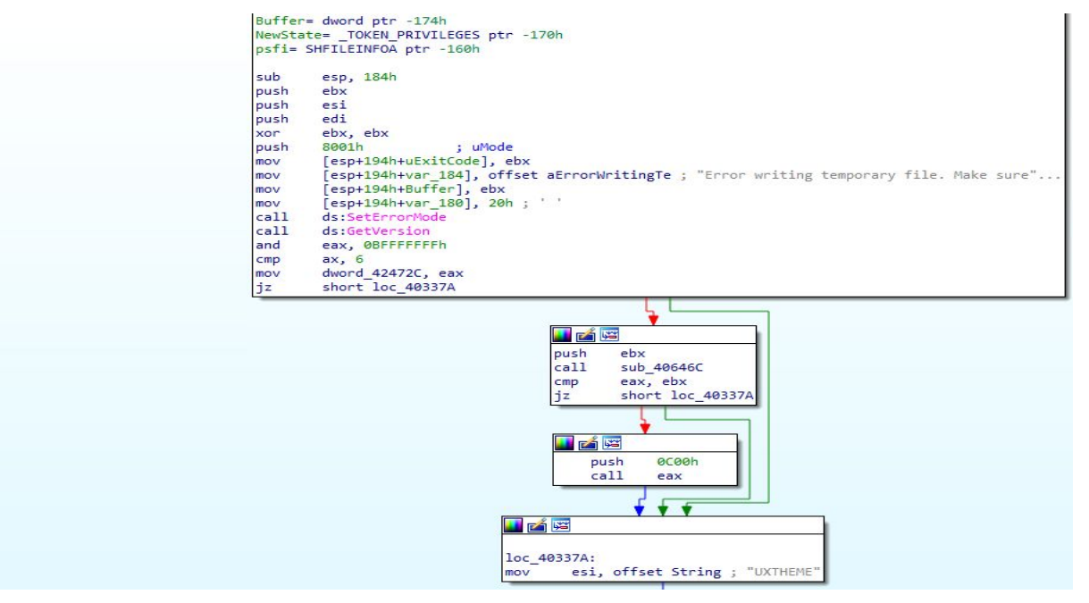
IDA Pro: We utilized the Imports tab to locate the GlobalAlloc memory call. From here, we went into the Graph view to find the dword variable, renamed for readability purposes, and traced the malware call. Using breakpoints from WinDbg, we could follow the malware call to then end.



```
ModLoad: 00000000`779b0000 00000000`77b53000   ntdll.dll
ModLoad: 00007fff`8dd40000 00007fff`8dd99000   C:\Windows\System32\wow64.dll
ModLoad: 00007fff`8ee50000 00007fff`8eed3000   C:\Windows\System32\wow64win.dll
(1fb0.1040): Break instruction exception - code 80000003 (first chance)
ntdll!LdrpDoDebuggerBreak+0x30:
00007fff`8f940570 cc              int     3
0:000> bp $exentry
0:000> g
ModLoad: 00000000`779a0000 00000000`779aa000   C:\Windows\System32\wow64cpu.dll
ModLoad: 00000000`76630000 00000000`76720000   C:\Windows\SysWOW64\KERNEL32.DLL
ModLoad: 00000000`77170000 00000000`77384000   C:\Windows\SysWOW64\KERNELBASE.dll
ModLoad: 00000000`75690000 00000000`7585a000   C:\Program Files\SentinelOne\Sentinel A
ModLoad: 00000000`75ce0000 00000000`75d5a000   C:\Windows\SysWOW64\ADVAPI32.dll
ModLoad: 00000000`75c20000 00000000`75cdf000   C:\Windows\SysWOW64\msvcrt.dll
ModLoad: 00000000`76720000 00000000`76796000   C:\Windows\SysWOW64\sechost.dll
ModLoad: 00000000`77660000 00000000`77720000   C:\Windows\SysWOW64\RPCRT4.dll
ModLoad: 00000000`759f0000 00000000`75b86000   C:\Windows\SysWOW64\USER32.dll
ModLoad: 00000000`767a0000 00000000`767b8000   C:\Windows\SysWOW64\win32u.dll
ModLoad: 00000000`75d60000 00000000`75d83000   C:\Windows\SysWOW64\GDI32.dll
ModLoad: 00000000`75d90000 00000000`75e6c000   C:\Windows\SysWOW64\gdi32full.dll
ModLoad: 00000000`75970000 00000000`759eb000   C:\Windows\SysWOW64\msvcp_win.dll
ModLoad: 00000000`76fc0000 00000000`770e0000   C:\Windows\SysWOW64\ucrtbase.dll
ModLoad: 00000000`75fd0000 00000000`76584000   C:\Windows\SysWOW64\SHELL32.dll

0:000:x86>
```

Figure 1: A screenshot of the execution of a breakpoint command on the entry of the executable.



```
Buffer= dword ptr -174h
NewState= _TOKEN_PRIVILEGES ptr -170h
psfi= SHFILEINFOA ptr -160h

sub     esp, 184h
push    ebx
push    esi
push    edi
xor     ebx, ebx
push    8001h           ; uMode
mov     [esp+194h+uExitCode], ebx
mov     [esp+194h+var_184], offset aErrorWritingTe ; "Error writing temporary file. Make sure"...
mov     [esp+194h+Buffer], ebx
mov     [esp+194h+var_180], 20h ; ' '
call    ds:SetErrorMode
call    ds:GetVersion
and     eax, 0BFFFFFFFh
cmp     ax, 6
mov     dword_42472C, eax
jz      short loc_40337A

push    ebx
call    sub_40646C
cmp     eax, ebx
jz      short loc_40337A

push    0C00h
call    eax

loc_40337A:
mov     esi, offset String ; "UXTHEME"
```

Figure 2: A screenshot of how the screen looks after a file is decompiled in IDA Pro. The user is presented with a graph of the program.

RESULTS

WinDbg was able to help debug the malware into simple code that made it easy to read and understand what the malware was doing by allowing us to download step by step information and save it to inspect later.

IDA Pro was able to show a manipulative call to the system memory & corrupted the users system by abusing user privileges, the clipboard, disabling the firewall, and other malicious intentions.

Resource:
https://www.youtube.com/watch?v=QuFJpH3My7A

This Capstone project focused on utilizing two applications, IDA Pro and WinDbg, to debug real malware. Our research also extended into supporting areas such as industry standards, and enterprise EPP/EDR solutions.

Utilize this QR code, and visit our website to access all our presentations and reports, as well as a full testing demonstration video, where we reverse engineer a piece of malware, utilizing the tools researched in this project!

KENNESAW STATE UNIVERSITY
COLLEGE OF COMPUTING AND SOFTWARE ENGINEERING

Author(s): Andy Pham, Cynthia Marcellus, Josh Rowland, Nathan Rowe, Shamour Jones
Advisors(s): Dr. Hossain Shahriar