Reverse Engineering & Vulnerability Analysis

Haisav Chokshi
Cyber Security Student

Reverse Engineering is actually a process in which an object or a code is deconstructed or disassembled in a way to examine or analyze in detail in order to reveal its design, architecture, code to extract knowledge.

Reverse Engineering is everywhere. Everyone is aware that China is world's largest manufacturer and it makes almost a second copy or a duplicate of everything. What they do is buy a product, strip it down and then look or find out how the original is made. Now buy cheaper components, labour also cost them less so reassemble it and represent it a new form. Thus, any item can be reverse engineered. Similarly, software, codes can also be reverse engineered. Essentially, it's like a third party who was not involved in writing code finds bugs or loopholes that can exploit the code or any related functionality from it. We actually learn a lot from reverse engineering. In layman's term let us say we have a software that requires a key so now essentially whatever you enter is getting compared to some thing or the other in the code. Now I get into the code, find the place where the conditional statement is there. Now I modify the conditional jump to an unconditional jump (in assembly code) by just editing and adding a or statement. That's it now whatever value we enter gets satisfied and thus you are able to use the software. This was just exploiting the code with a little technical knowledge but this was very analogous in order to understand reverse engineering. If we have the right engineering mindset and the skills so by reverse engineering, we will be able to understand different approaches to a solution, understand the engineered work deeply and then we can suggest or work upon improvising it.

Reverse Engineering Software

Reverse Engineering skills comes very handy when it comes to detect and neutralize the malwares or viruses. It can also protect Intellectual Property Rights (IPR). According to me all related activities of Software Reverse Engineering are mainly divided into 2 categories. The first one being related to **Software Development** and the other one related to **Software Security.**

Basically, SRE as I perceive is the practice to call upon your investigative nature when on needs to learn how and why of things often in the absence of proper documentation.

At first glance it may seem to us that need for Software reverse engineer can be eliminated by simply maintaining good documentation for all software that is written. Although the presence of that ideal would definitely lower the need, but that's not the reality.

For example, even a company that has brought software to market may no longer understand it because the original designers and developers may have left, or components of the software may have been acquired from a vendor – who may no longer be in business. Going forward, the vision is to include SRE incrementally, as part of the normal development, or "forward engineering" of software systems. At regular points during the development cycle, code would be reversed to rediscover its design so that the documentation can be updated. This would help avoid the typical situation where detailed information about a software system, such as its architecture, design constraints, and trade-offs, is found only in the memory of its developer.

It is extremely important to understand that to reverse engineer any code one must be well versed with the understanding of assembly language. The code of any software or even a malicious program in whichever language it may be written, be it a high-level language like Python, Java or any other language there is inherently an assembly code with it. Now the question is how is that possible. See its either our compiler that translates the code into Assembly or the interpreter does it for us in real time.

Figuring out Reverse Engineering in itself is a broad errand. The difficulty or the trouble profoundly depends on the product or the item or the code. I particularly feel that successful/effective reverse engineering requires a good domain knowledge. Now that thing can be related to security, cracking it or any other product in general. Reverse Engineering anything basically implies that we need to dissect and thoroughly analyse all the pieces of code to understand how it works and how it's made.

It should also be noted that you need to always consider the ethical and legal concerns around reverse engineering.

Reverse Engineering with respect to Malware Analysis

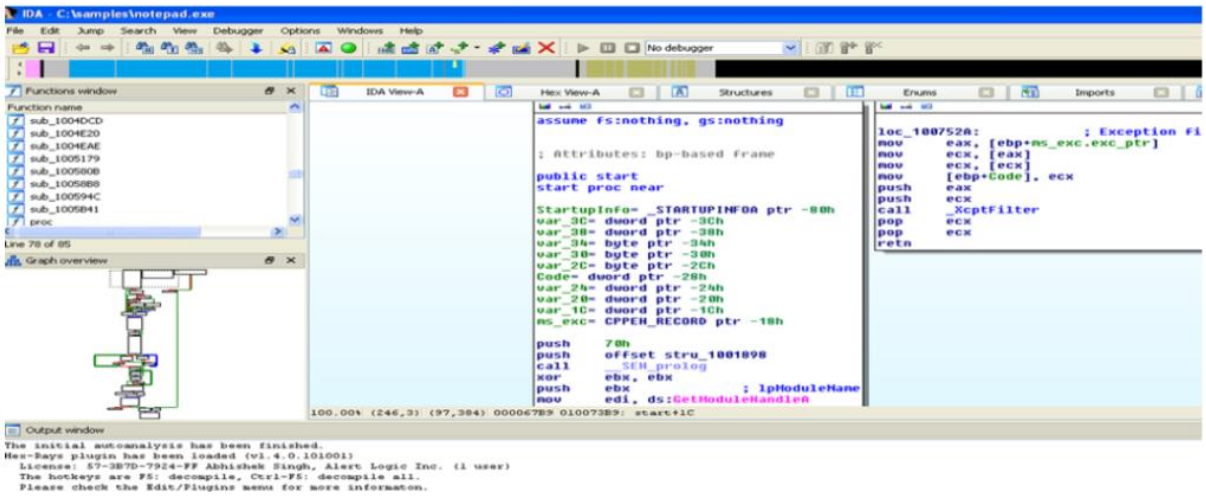
The most famous and interesting part of any Malware Analysis comes into place when Dynamic analysis takes place. Here we are running the code in a controlled isolated environment. Reverse Engineering and malware analysis have a very important role to play in Cyber Security. As a process Reverse engineering has evolved humongously as malware nowadays are becoming more sophisticated day by day and consequently tools have also improved, but still, it remains critical.

Reverse Engineering any piece of malware will surely involve disassembling and sometimes decompiling a software program or a malicious code. The main of this process is to convert the binary instructions or high-level language instructions to assembly code or in a form of a mnemonics so that the reverse engineer can examine the program and comment on the functionality and the severity or impact of the code. Now he/she (our reverse engineer) can create a solution that is able to mitigate the threat intended in the program. Once we found the main intention behind the code then we know which of our vulnerability was exploited and now can be patched in case of a software hit by a malware attack.

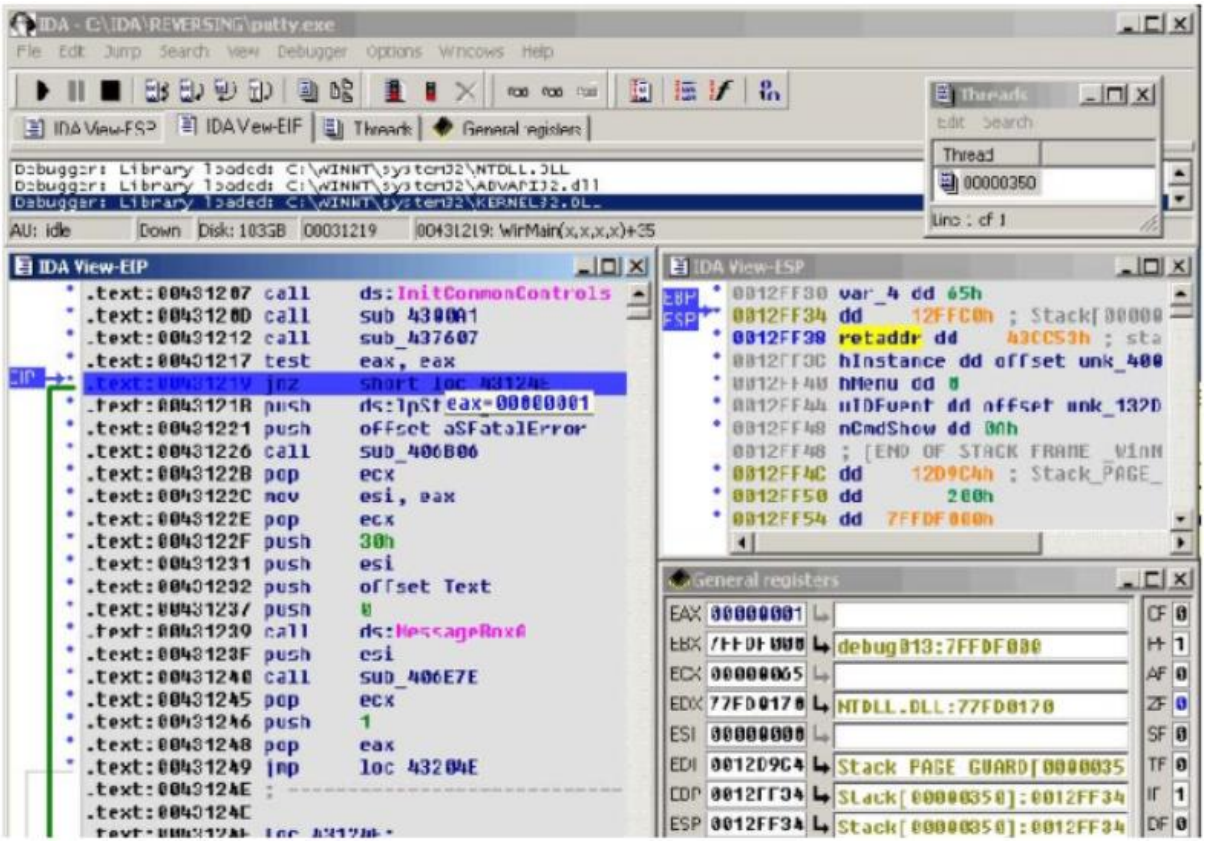
As a part of the job reverse engineers must be able to extract hints inside a malware program revealing when a program was created, with what embedded resources used, encryption keys and other metadata about the file. When a very famous WannaCry Ransomware was reverse engineered, in an attempt to find a way in order to track how it spread actually led to the discovery of what something people know today as “kill-switch”. And this was the turning point and proved incredibly helpful to stop its spread.

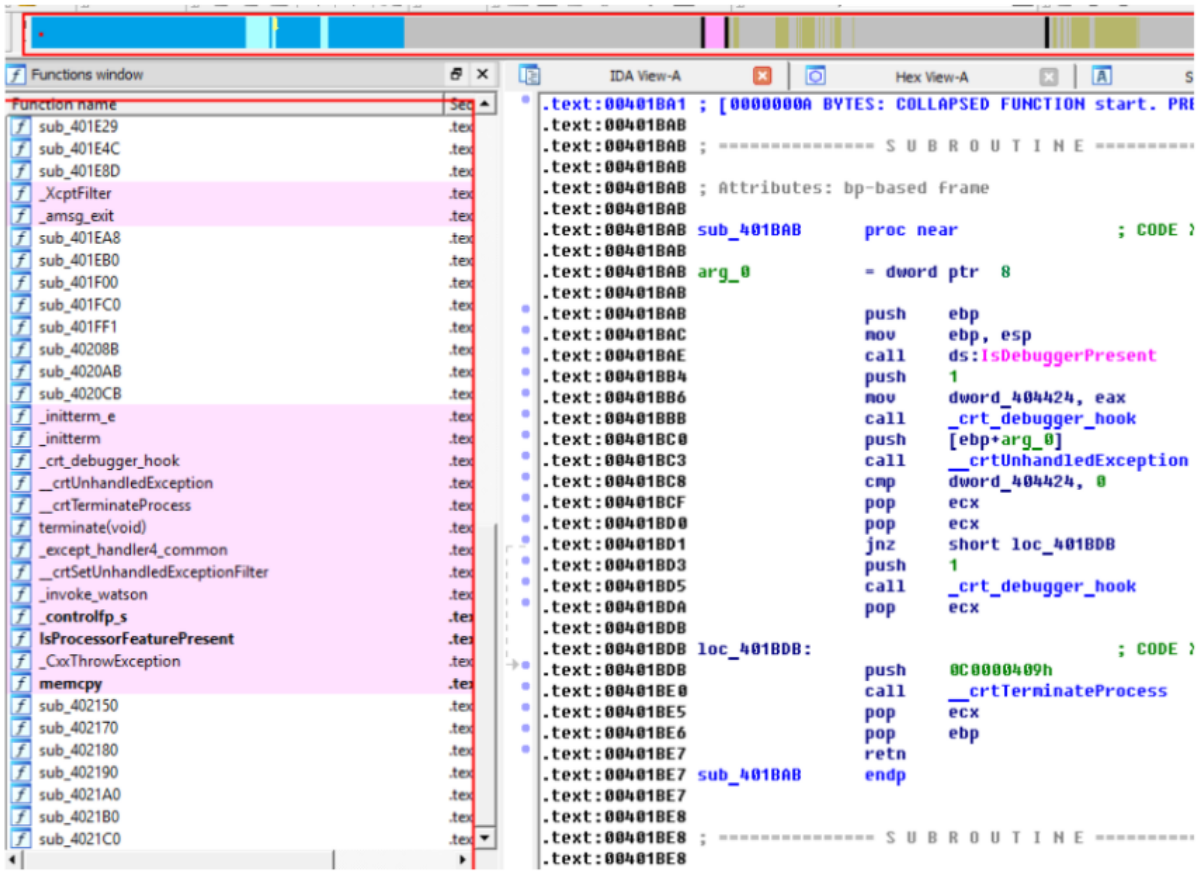
In order to perform Reverse Engineering many a times tools are used. Even I myself have used many. Here are some of them:

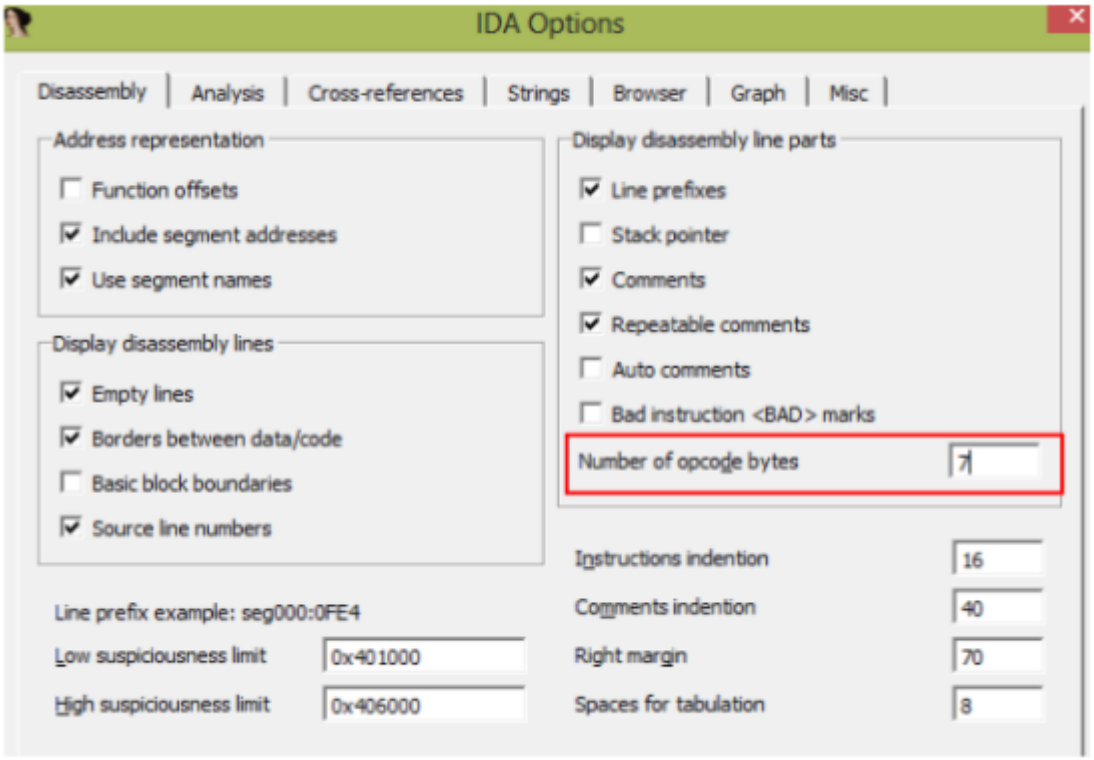
Disassemblers: It will change or take apart an application and convert it to produce assembly code. Essentially a program that can translate machine language to assembly language. There are many disassemblers available in the market but my favorite and most popular is IDA. Interactive Disassembler (IDA) also the better-known version is IDA-Pro. Apart from this there are Decompilers available for converting binary code into native code, although they are not available for all architectures. They are used to dissect binary codes to assembly codes. They are also useful in extracting strings, functions and libraries as well.



Address	Ordinal	Name	Library
0040A254		LoadLibraryA	KERNEL32
0040A258		GetProcAddress	KERNEL32
0040A25C		VirtualProtect	KERNEL32
0040A260		VirtualAlloc	KERNEL32
0040A264		VirtualFree	KERNEL32
0040A268		ExitProcess	KERNEL32
0040A270		?_Xbad_alloc@std@@YAXXZ	MSVCP110
0040A278		exit	MSVCR110
0040A280		NtQueryInformationProcess	ntdll
0040A288		MessageBoxA	USER32


IDA - C:\IDA\REVERSING\putty.exe
IDA View-EIP
IDA View-ESP
General registers
Threads
.text:00431207 call ds:InitCommonControls
.text:0043120D call sub_4300A1
.text:00431212 call sub_437607
.text:00431217 test eax, eax
.text:00431219 jnz short loc_43124E
.text:0043121B push eax=00000001
.text:00431221 push offset aSFatalError
.text:00431226 call sub_406B06
.text:0043122B pop ecx
.text:0043122C mov esi, eax
.text:0043122E pop ecx
.text:0043122F push 30h
.text:00431231 push esi
.text:00431232 push offset Text
.text:00431237 push 0
.text:00431239 call ds:MessageBoxA
.text:0043123F push esi
.text:00431240 call sub_406E7E
.text:00431245 pop ecx
.text:00431246 push 1
.text:00431248 pop eax
.text:00431249 jmp loc_43204E
.text:0043124E ;
.text:0043124C
EAX 00000001
EBX 7FFDF000
ECX 00000065
EDX 77FD0170
ESI 00000000
EDI 0012D9C4
EDP 0012FF34
ESP 0012FF34
CF 0
PF 1
AF 0
ZF 0
SF 0
TF 0
IF 1
DF 0


Functions window
IDA View-A
Hex View-A
Function name
sub_401E29
sub_401E4C
sub_401E8D
_XcptFilter
_amsg_exit
sub_401EA8
sub_401EB0
sub_401F00
sub_401FC0
sub_401FF1
sub_40208B
sub_4020AB
sub_4020CB
_initterm_e
_initterm
_crt_debugger_hook
__crtUnhandledException
__crtTerminateProcess
terminate(void)
_except_handler4_common
__crtSetUnhandledExceptionFilter
_invoke_watson
_controlfp_s
IsProcessorFeaturePresent
_CxxThrowException
memcpy
sub_402150
sub_402170
sub_402180
sub_402190
sub_4021A0
sub_4021B0
sub_4021C0
.text:00401BA1 ; [0000000A BYTES: COLLAPSED FUNCTION start. PRE
.text:00401BAB ; ============== S U B R O U T I N E ==========
.text:00401BAB ; Attributes: bp-based frame
.text:00401BAB sub_401BAB proc near ; CODE X
.text:00401BAB arg_0 = dword ptr 8
.text:00401BAB push ebp
.text:00401BAC mov ebp, esp
.text:00401BAE call ds:IsDebuggerPresent
.text:00401BB4 push 1
.text:00401BB6 mov dword_404424, eax
.text:00401BBB call _crt_debugger_hook
.text:00401BC0 push [ebp+arg_0]
.text:00401BC3 call __crtUnhandledException
.text:00401BC8 cmp dword_404424, 0
.text:00401BCF pop ecx
.text:00401BD0 pop ecx
.text:00401BD1 jnz short loc_401BDB
.text:00401BD3 push 1
.text:00401BD5 call _crt_debugger_hook
.text:00401BDA pop ecx
.text:00401BDB loc_401BDB: ; CODE X
.text:00401BDB push 0C0000409h
.text:00401BE0 call __crtTerminateProcess
.text:00401BE5 pop ecx
.text:00401BE6 pop ebp
.text:00401BE7 retn
.text:00401BE7 sub_401BAB endp
.text:00401BE8 ; ============== S U B R O U T I N E ==========

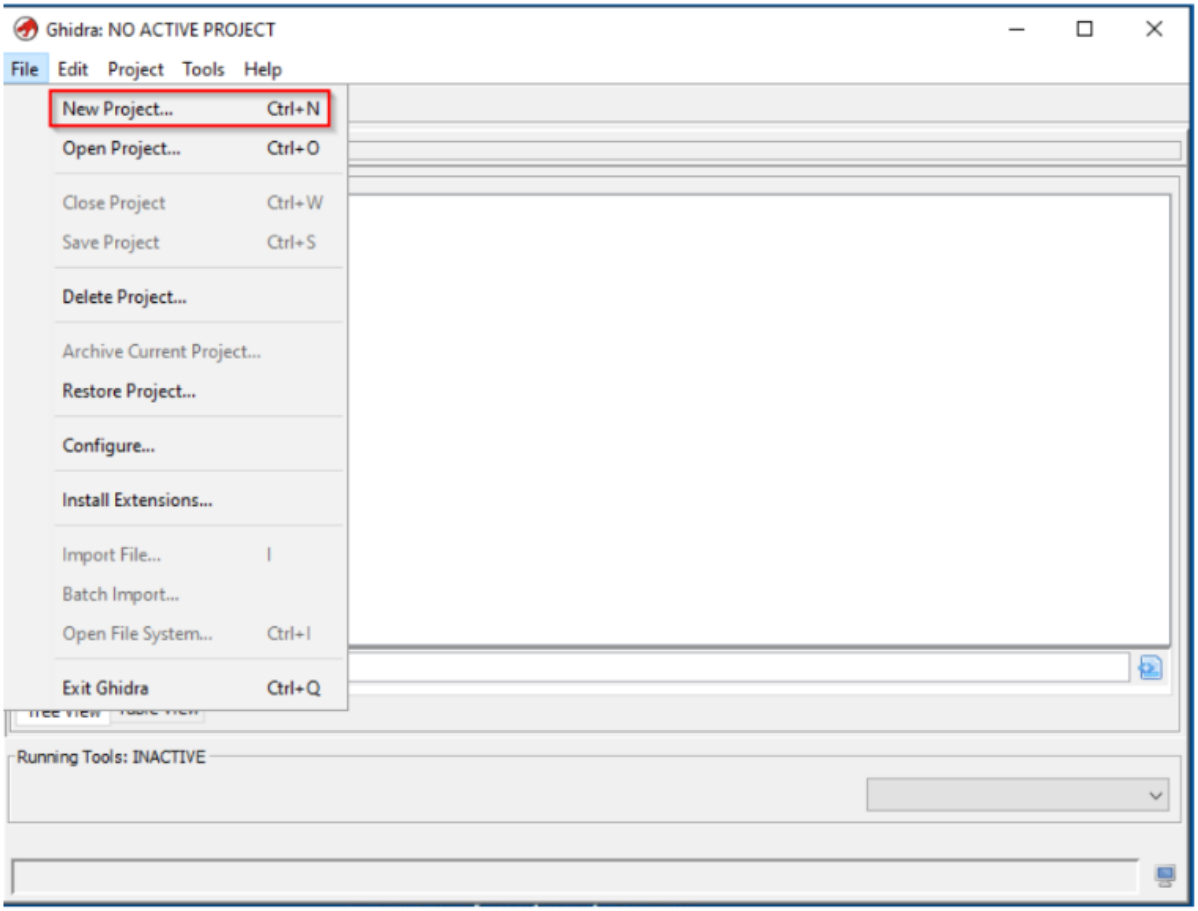


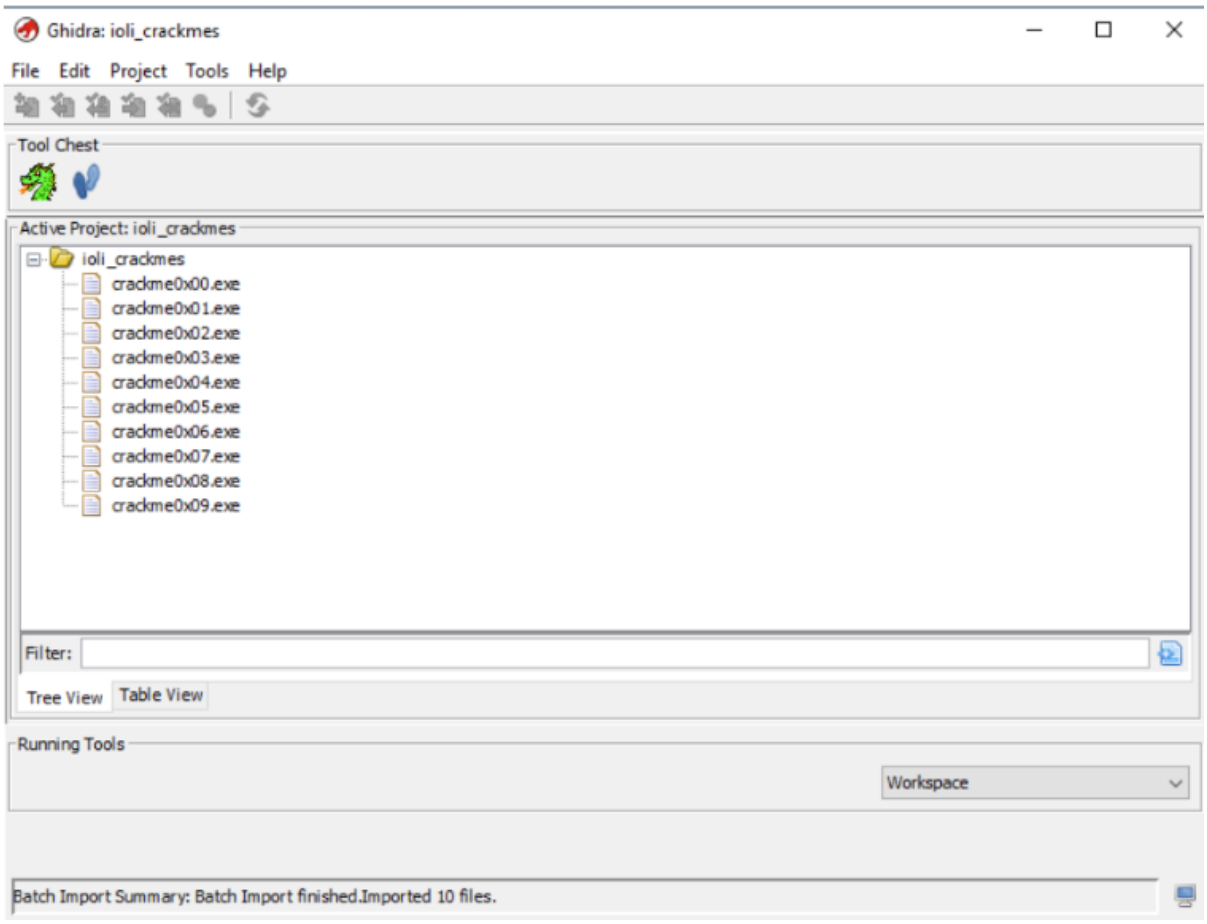
Ghidra

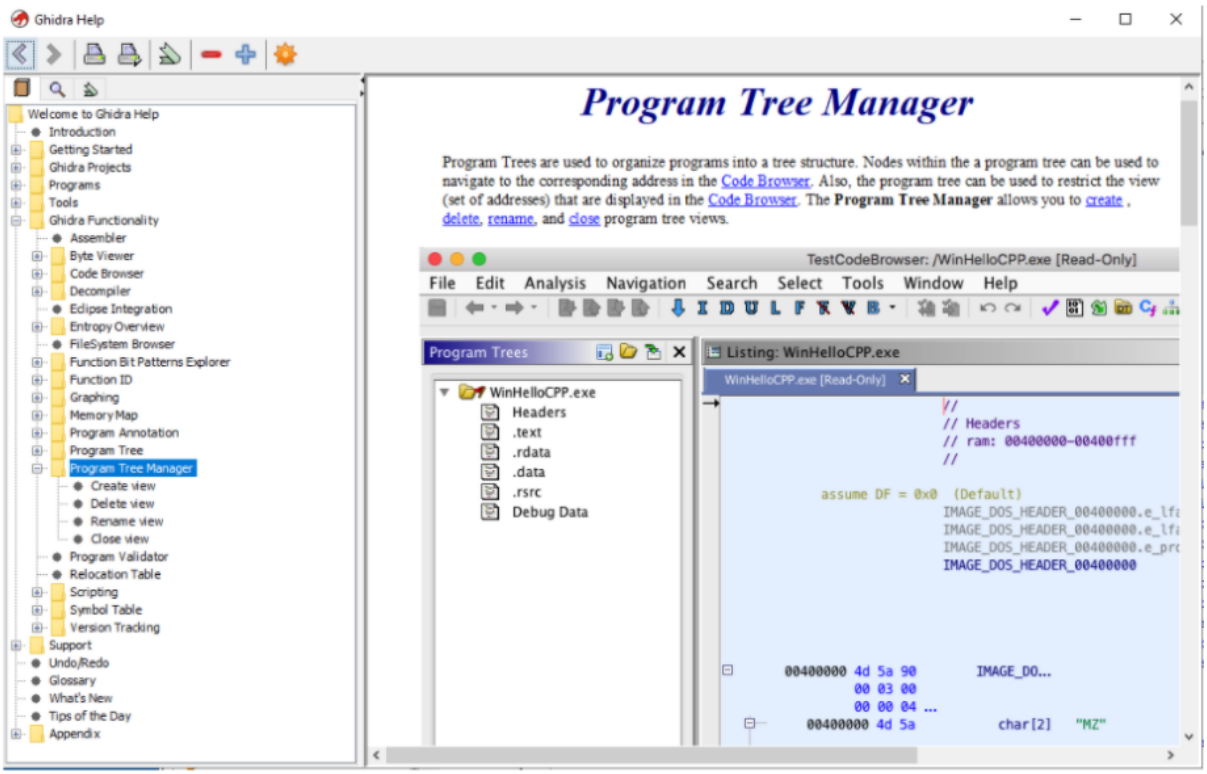
The general methodology we used to reverse engineer this program:

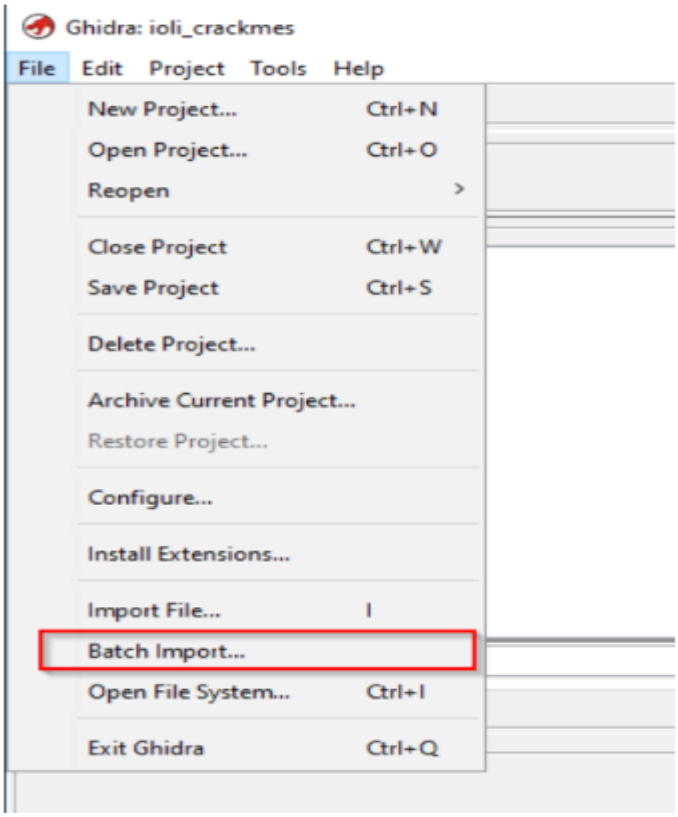
1. Do initial triage and also analyze by seeing what data/strings are found within the target.

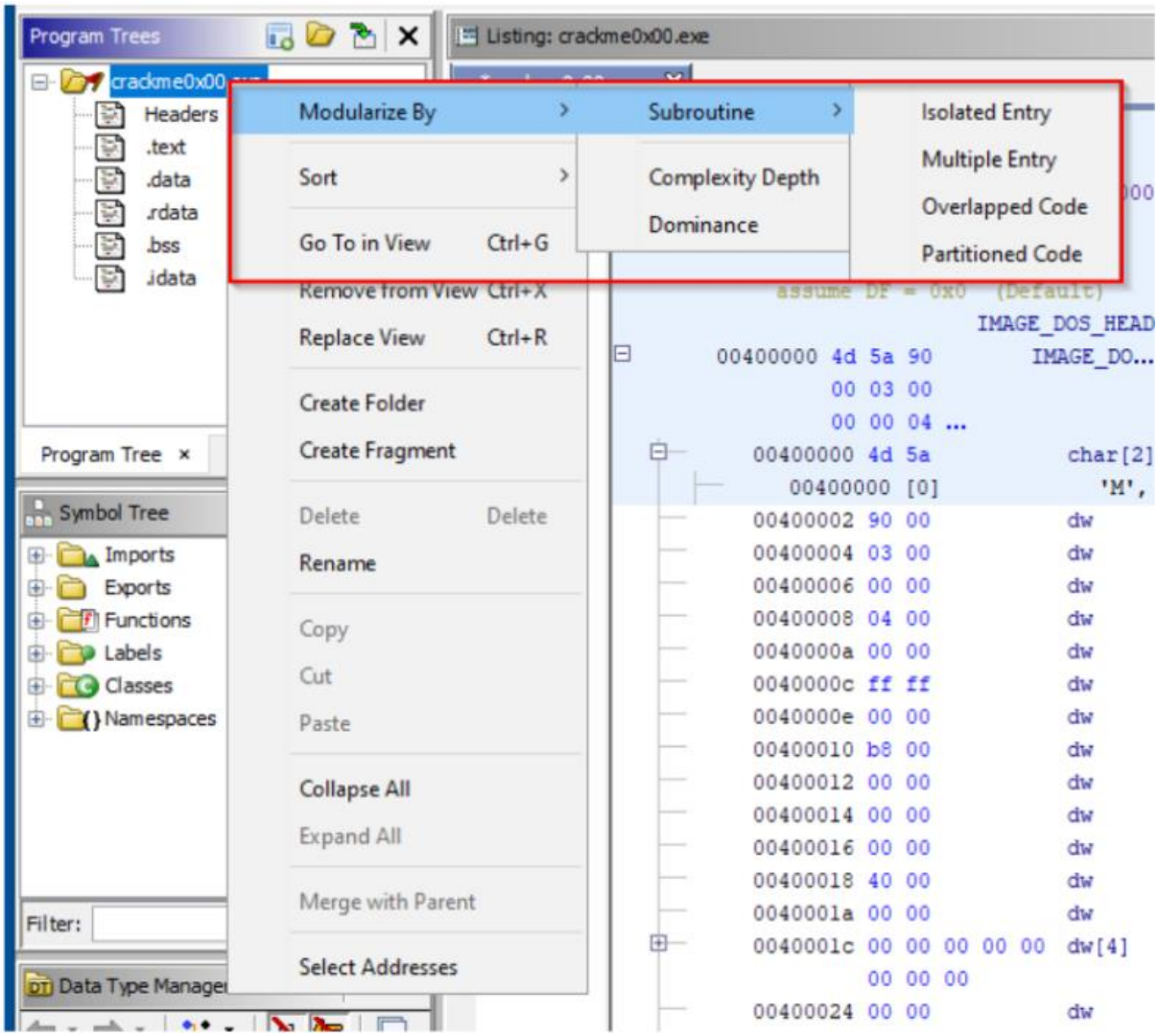
2. Follow references for interesting data/strings to surrounding assembly code.
3. Make Educated guess about what functions and variables in the assembly are responsible for.
4. Annotate guesses using comments and rename functions/variables. We must improve and revise and review as understanding of the program improves.
5. Use understanding and insights from the process.



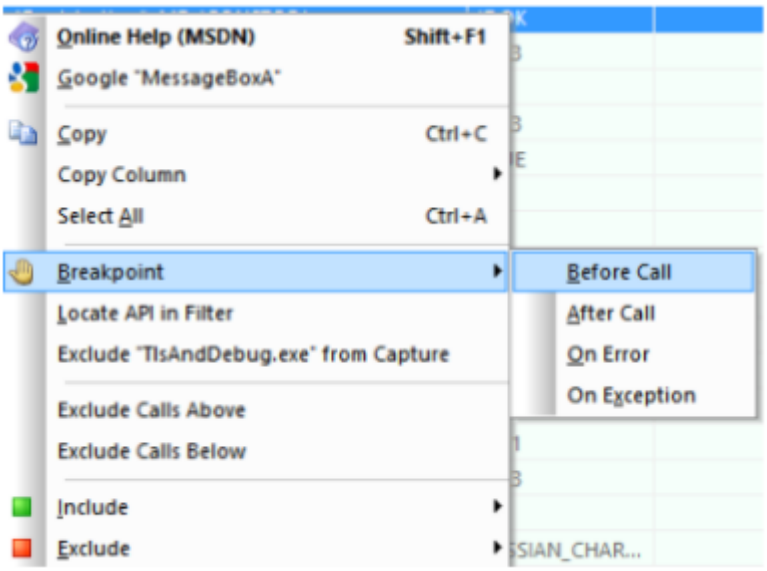

Ghidra: ioli_crackmes
File Edit Project Tools Help
Tool Chest
Active Project: ioli_crackmes
ioli_crackmes
crackme0x00.exe
crackme0x01.exe
crackme0x02.exe
crackme0x03.exe
crackme0x04.exe
crackme0x05.exe
crackme0x06.exe
crackme0x07.exe
crackme0x08.exe
crackme0x09.exe
Filter:
Tree View
Table View
Running Tools
Workspace
Batch Import Summary: Batch Import finished.Imported 10 files.

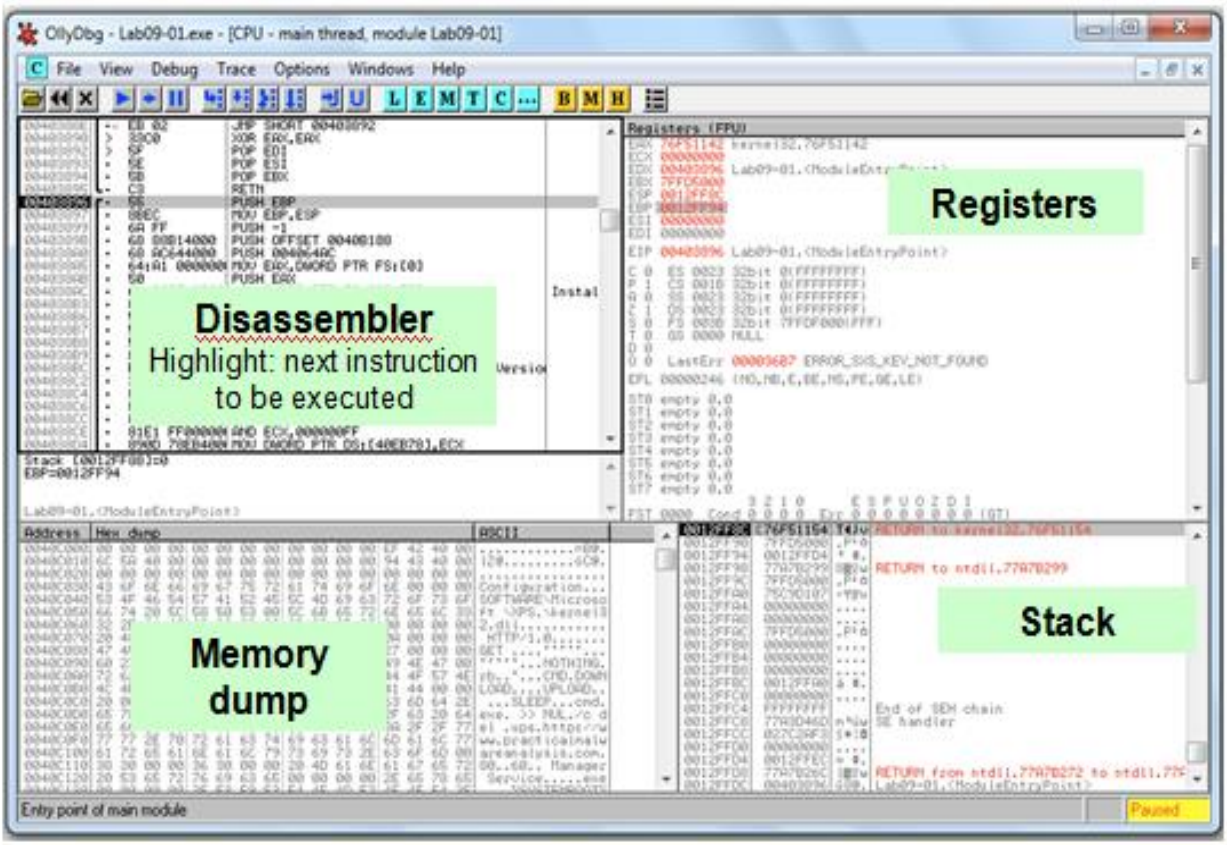

Ghidra Help
Welcome to Ghidra Help
Introduction
Getting Started
Ghidra Projects
Programs
Tools
Ghidra Functionality
Assembler
Byte Viewer
Code Browser
Decompiler
Eclipse Integration
Entropy Overview
FileSystem Browser
Function Bit Patterns Explorer
Function ID
Graphing
Memory Map
Program Annotation
Program Tree
Program Tree Manager
Create view
Delete view
Rename view
Close view
Program Validator
Relocation Table
Scripting
Symbol Table
Version Tracking
Support
Undo/Redo
Glossary
What's New
Tips of the Day
Appendix
Program Tree Manager
Program Trees are used to organize programs into a tree structure. Nodes within the a program tree can be used to navigate to the corresponding address in the Code Browser. Also, the program tree can be used to restrict the view (set of addresses) that are displayed in the Code Browser. The Program Tree Manager allows you to create , delete, rename, and close program tree views.
TestCodeBrowser: /WinHelloCPP.exe [Read-Only]
File Edit Analysis Navigation Search Select Tools Window Help
Program Trees
WinHelloCPP.exe
Headers
.text
.rdata
.data
.rsrc
Debug Data
Listing: WinHelloCPP.exe
WinHelloCPP.exe [Read-Only]
// Headers
// ram: 00400000-00400fff
assume DF = 0x0 (Default)
IMAGE_DOS_HEADER_00400000
00400000 4d 5a 90 IMAGE_DO...
00400000 4d 5a char[2] "MZ"

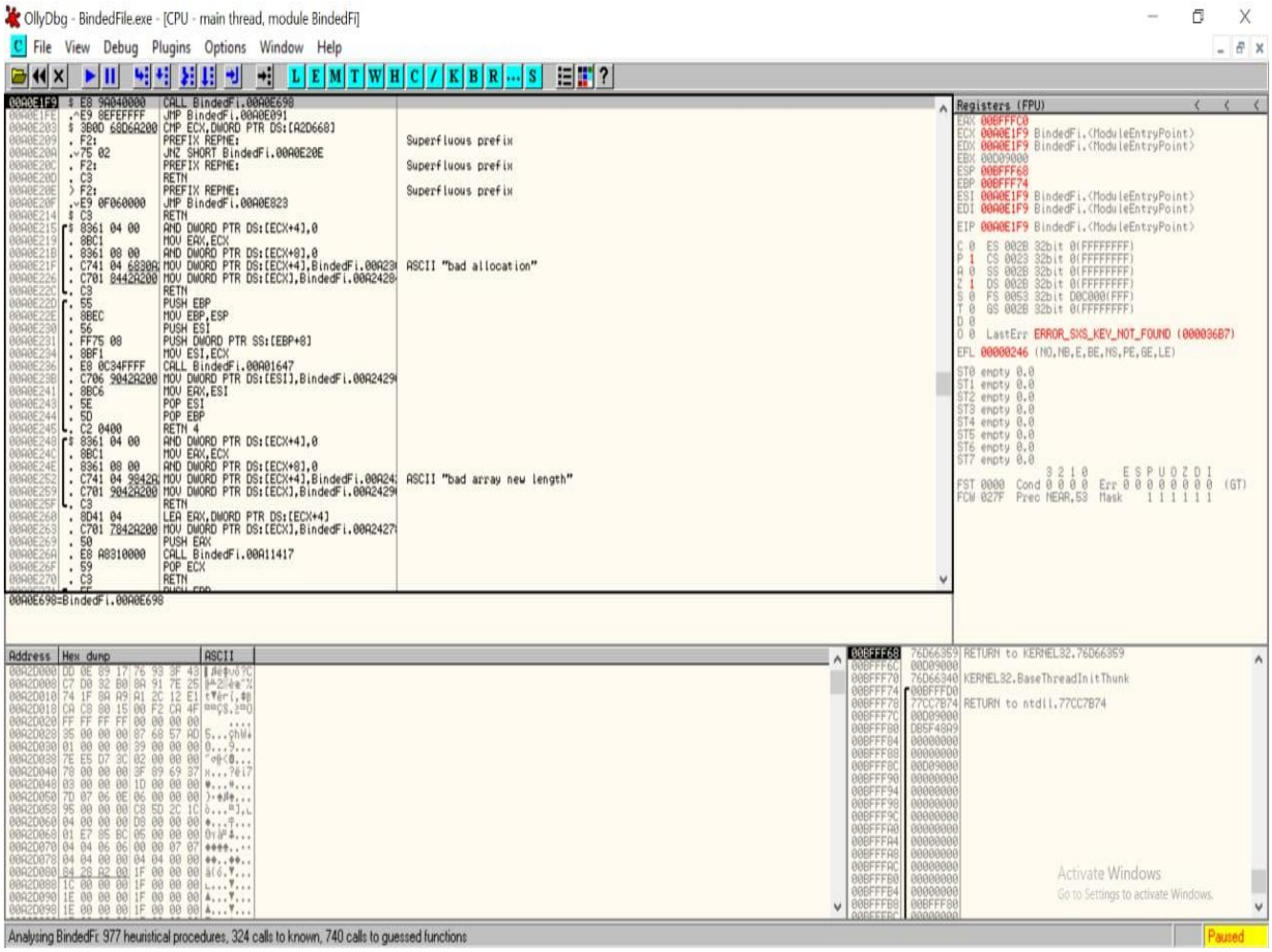

Ghidra: ioli_crackmes
File
Edit
Project
Tools
Help
New Project... Ctrl+N
Open Project... Ctrl+O
Reopen
Close Project Ctrl+W
Save Project Ctrl+S
Delete Project...
Archive Current Project...
Restore Project...
Configure...
Install Extensions...
Import File... I
Batch Import...
Open File System... Ctrl+I
Exit Ghidra Ctrl+Q



Debugger: They are very helpful when it comes to program manipulation in order to gain insights into what the program is doing when running. As a reverse engineer these debuggers let us control a certain part of program's memory. This definitely allows more insight into what program is capable of doing and how actually it's impacting our system or network. As far as debuggers are concerned, I use and like Olly DBG, Immunity Debugger etc. They expand the horizon of disassemblers and contribute by providing CPU registers, hex dumps and view of the stack as well. It's a common practice that many programmers follow is to set breakpoints as well as edit assembly codes at run time. Here we can also run the code line by line so as to investigate the results.


Online Help (MSDN) Shift+F1
Google "MessageBoxA"
Copy Ctrl+C
Copy Column
Select All Ctrl+A
Breakpoint
Locate API in Filter
Exclude "TlsAndDebug.exe" from Capture
Exclude Calls Above
Exclude Calls Below
Include
Exclude
Before Call
After Call
On Error
On Exception


OllyDbg - Lab09-01.exe - [CPU - main thread, module Lab09-01]
File View Debug Trace Options Windows Help
Registers
Disassembler
Highlight: next instruction to be executed
Memory dump
Stack
Entry point of main module
Paused



Other tools like PE Viewers, Network Analyzers like Wireshark comes handy and play an integral part in monitoring.

Apart from all the Advantages that Reverse Engineering has there are some **disadvantages** as well. Let us look at them:

You can face legal ramifications or court issues if you don't follow copyright and patent law.

Not every component or product is a great candidate of reverse engineering.

The Challenge with Reverse Malware Engineers Today

As malware programs today are becoming more intricate, it becomes increasingly likely that disassemblers may fail somehow or the decompiler produce obfuscated code. So malware reverse engineers need more time in figuring out everything and meanwhile in this time the malware will cause a great havoc to the network. And because of this the focus on dynamic malware analysis has increased. We all are quite aware of the fact that dynamic analysis takes place in a isolated system (sandbox) to watch what the malware is capable of doing and simply watch if required.

There as actually a ton of advantages in utilizing a sandbox but there are some downsides as well. For example, many sophisticated malicious programs use evasion techniques in order to detect their presence in a sandbox and goes to the dormant or inactive state when detect that they are in a sandbox.

If a sandbox is detected, the malware will abstain itself from showing its true malicious nature. Some very advanced malware programs also posses a suite of tools they use to outmaneuver sandboxes and evade detection. They can postpone their vindictive excersices possibly act only when the user is active and hide malicious code in areas where it will not be detected.

This implies that reverse engineers can’t depend solely on the dynamic techniques. And also at the same time, reverse engineering or figuring out every new malware threat is unrealistic.

Vulnerability Analysis

Vulnerability means susceptibility to some kind of loophole in the code or something that can be compromised to gain benefits especially a kind of security flaw in this digital world.

Vulnerability analysis or assessment is the process of recognizing, measuring, and prioritizing (or ranking) the vulnerabilities or the weakness in the system. Every system in this world requires a vulnerability analysis be it any website, information technology system, transportation system, communication system or any other system. Actually it's very analogous to our real life where vulnerability is a weakness in anyone or a defect in a gadget or similar to look at the potential hazards in a building or an infrastructure. This was all in general vulnerability analysis but to be specific to IT Sector, it's one of the ways to secure IT assets, to keep an attention to the vulnerabilities in an environment and react very quickly to mitigate a potential danger or threat. As per the new Government norms or rules in the respective country it is necessary to follow some compliances and a comprehensive vulnerability analysis report provides the organization with the knowledge, awareness and risk background necessary to understand threats to their environment so that they can respond in an appropriate manner.

Benefits of Performing Vulnerability Analysis

- We can easily identify loopholes or better known as security exposures before the attackers or the malicious persons find them.
- Create an inventory or a database of all the devices present on the network, including purpose and system information. This also include vulnerabilities that are associated with a specified device.
- It helps the enterprise with the planning of upgrades and future assessments of all devices present.
- It defines the level of risk that already exists in the organization or the network. Basically characterize the degree of danger that exists on the organization.
- Establish a business risk/benefit curve and optimize security investments.

Vulnerability Analysis with respect to Cyber Security

Cybersecurity is a growing concern for anyone operating a business today. As our technology progresses, so do the dangers that we face are innovating. Today scenario is such that security services have to do a lot rather than just to defend. They must stay ahead of these hazards. In real time we need security testing precautions that effectively gain and learn from encounters with malicious software or be ready to mitigate hazards like phishing attacks and data breaches.

Attacks like this can result in loss of data or corrupted data. That sort of outcome will surely affect the income and the trust that clients have on the organizations. Those are two things that are hard to

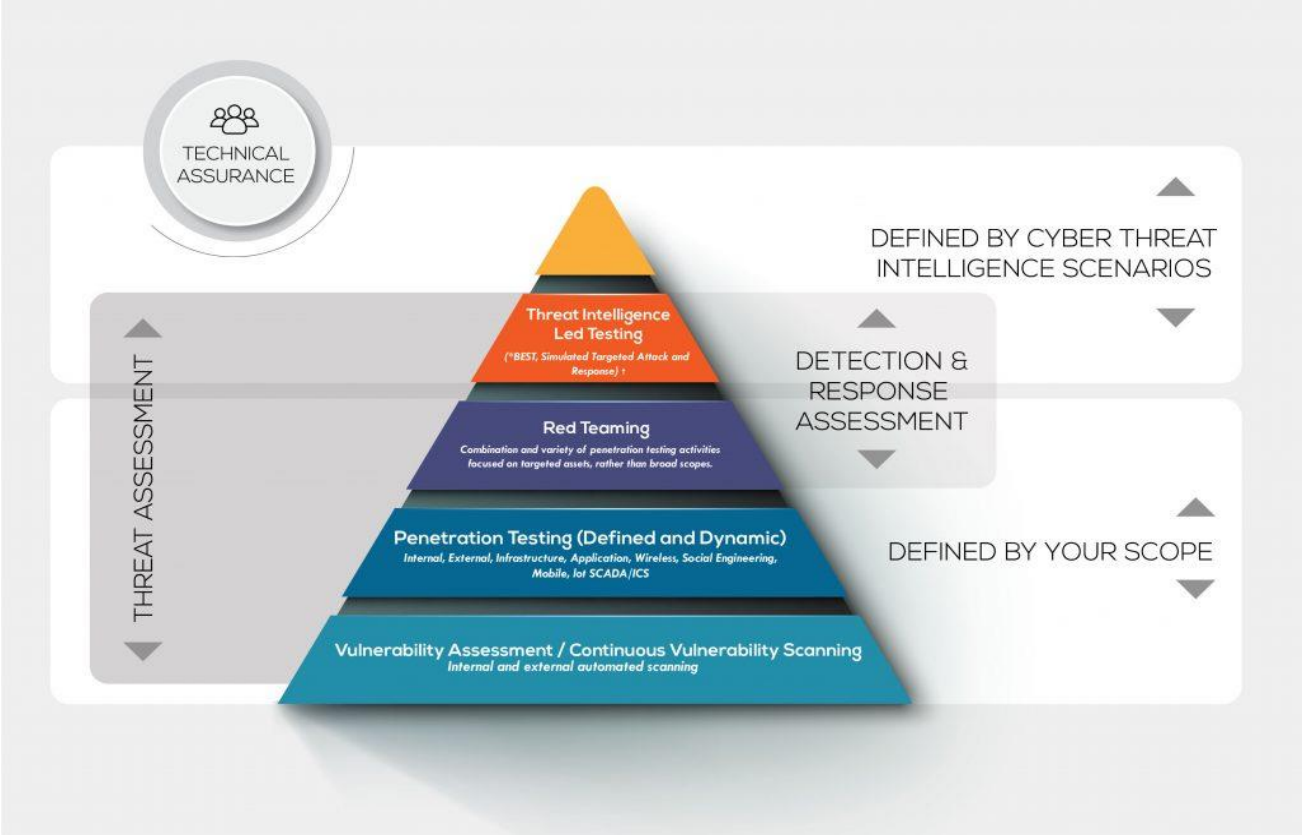
acquire back once whenever they are gone. This makes it fundamental to learn as much as possible about the dangers or the threats focuses on the weakness present in the frameworks.

It's very common that many people lump up vulnerability analysis and penetration testing in the same category. But it comes to truth, they are two different strategies for expanding our security development and a superior security program will include both approaches.

This is trailed by an evaluation that delves further into the information gathered, presents solution and provides risk management for the threats that we are dealing with. It is a detailed cycle that outcomes in a more unique and proactive way to deal with security.

Common Steps of Vulnerability Analysis:

1. Determine the hardware and software assets in an environment.
2. Determine the quantifiable value (criticality) or the worth of these assets.
3. Recognize the security vulnerabilities impacting the assets i.e our resources.
4. Determine a quantifiable threat or risk or danger score for each vulnerability.
5. Mitigate the highest risk vulnerabilities means the most elevated danger from the most valuable assets.



Vulnerability Analyses as a Technical Process

Information Gathering & Discovery

At the root level it has 3 phases. In the first phase, organizations conduct an information gathering and discovery in order to understand the hardware and software present in their environment. Here they frequently do network scanning to discover hosts, port scanning to discover the services and protocols that might be vulnerable, and a review of directory service and DNS information to understand which hosts might be targeted by attackers.

Review and Enumeration

Once we complete our analysis a full discovery effort to understand the hosts present in the environment, a more thorough review and enumeration of operating systems, applications, ports, protocols and services determines the full extent of the attack surface vulnerable to attackers. In this phase it's very important determine the version information of assets of the organization as subsequent versions frequently patch old vulnerabilities and introduce new ones.

Detection and Reporting

The final phase of this process generates reports, complete with scores and risk information. The final step of the phase is to use remediation tools to patch, configure, or debug assets as necessary to reduce or eliminate the security risks present due to the vulnerabilities detected.

It's like getting an MRI Scan of all our systems.

Security researchers, bug bounty programs, and product vendors nowadays are discovering and reporting new vulnerabilities daily. These vulnerabilities are frequently caused by either coding errors or by security misconfigurations. Coding errors, including the failure to check user input, allow attackers to improperly access system memory, data, or to execute commands (including buffer overflow and injection attacks).

Common Vulnerabilities and Exposures (CVE)

The Common Vulnerabilities and Exposures (CVE) system provides a reference-method for publicly known information-security vulnerabilities and exposures.

CVE is a list of records — each containing an identification number, a description, and at least one public reference — for publicly known cybersecurity vulnerabilities.

CVE Records are used in numerous cybersecurity product and services from around the world, including the U.S. National Vulnerability Database (NVD).

~ Reference CVE from Official Website and website

Vulnerability Scanning Tools

There are many vulnerability scanners available in the market. They can be free, paid, or open-source. Most of the free and open-source tools are available on GitHub.

Here are some of them:

Nikto2

Netsparker

OpenVAS

W3AF

Arachni

Golismero

Intruder

Comodo Hacker Proof

Aircrack

Retina CS Community

Acunetix	Microsoft Baseline Security Analyzer
Nmap	Nexpose
OpenSCAP	Nessus Professional

Solar Winds Network Configuration Manager

Conclusion

A vulnerability analysis informs organizations about the weaknesses present in their environment and provides direction on how to reduce the risk. This process helps to reduce the chances for an attacker to breach an organization's IT systems – yielding a better understanding of assets, their vulnerabilities, and the overall risk to an organization.

For organizations seeking to reduce their security risk, a vulnerability analysis report is a good place to start. It provides a thorough, inclusive assessment of hardware and software assets, identifying vulnerabilities and providing an intuitive risk score. A regular assessment program assists organizations with managing their risk in the face of an ever-evolving threat environment, identifying and scoring vulnerabilities so that attackers do not catch organizations unprepared.

Submitted By : Haisav Chokshi