
U.S. ARMY CYBER SCHOOL
DEFEND
ATTACK
EXPLOIT

Reverse Engineering Workflow

IMPORTANT	Reverse engineering and software analysis is incredibly dangerous, as you are intentionally interacting with potentially malicious things. Bear in mind that some malware/malicious hardware can execute and/or propogate without user interaction. It is assumed that your analysis environment is properly established and secured. How to do this is far outside of the scope of this course. Organizations and individuals must establish and set their own standards and accreditation for analysis environments.

The Following is specific to reverse engineering of software

Initial Static Analysis

Initial static analysis of a binary gives an analyst, or team of analysts, several clues as to what the binary is designed to do and how it really works.

1. Determine file type - Is it an executable? What environment is it designed to run in? (OS,cpu architecture, etc)
2. Determine if file is packed/compressed (UPX)
3. Find plain text ascii and unicode strings
4. View imports/exports to get a hint of functionality/calls (is it importing a library that can open a socket, etc?)
5. Look for encrypted sections of the binary

Behavioral Analysis

Behavioral Analysis is the fastest way to gain insight into how a binary works.

1. Take a snapshot of the analysis environment - Important! Taking a snapshot on an OpenStack VM takes a substantial amount of time.
2. Take a snapshot of critical configurations of the analysis environment. (Things like the registry, important directories, etc)
3. Launch realtime analysis tools (Things like procmon and fakenet)
4. Execute and interact with the object/binary while taking note of observations.
5. Stop the binary and view how it affected critical configurations (registry, files, etc) by comparing to previous snapshots
6. Analyze results of realtime analysis tools (did fakenet catch network calls, did procmon show it writing to a file, etc)

Dynamic Analysis

Dynamic analysis is similar to behavioral, except the analyst is attaching the process to a debugger

1. Execute binary in a debugger
2. Step through binary, setting breakpoints as approriate
3. Continuously rerun the binary and edit it’s parameters through the debugger, as you learn more about how it works
4. Document all observations and modifications

Disassembly

An analyst will eventually get to a point where they need to disassemble a binary to learn more about how it runs

1. Disassemble binary in IDA, Ghidra, or other disassembler
2. Use notes to find artifacts within the disassembly
3. Find a good spot to work from within the binary. Then quickly browse from the top to the bottom of the disassembly to view the overall flow of the disassembly
4. Rename variables and functions as appropriate when quickly scanning top to bottom of the disassembly.
5. Work your way from the bottom to the top - if there are two outcomes choose the one you want to end at, then work your way up from there to determine what needs to happen for the program to flow to the desired outcome.

Document findings

Analysts must document their findings after performing reverse engineering or software analysis.

1. Document all discovered binary traits, capabilities, and behaviors to include the conditions they must run under.
2. Document potential uses for the binary.
3. Create mitigations for the binary if it is malicious.
4. Create signatures and indicators of compromise to detect the binary in the future.
5. Document and save the tools, scripts, code, methods used to analyze the software to better analyze related software in the future.
6. Document proof of concept for exploitation of the binary if it is found to be vulnerable and a potential target. For example, if the binary is running on an adversary network, or if a friendly network may be using the binary.

Cycle through all of the described phases

Reverse engineering is cyclical. There is no absolute best way to analyze all things. Continue to go back and forth through the various methods of analysis as needed.
For example, if an analyst finds something that looks like decryption when disassembling a binary, they may desire to return to the dynamic analysis phase and watch the decryption as it happens in a debugger.
An analyst may also patch a binary in a disassembler to prevent it from doing something like detecting if it is being debugged, then run this newly patched binary in a debugger.
Keep track of why you are performing the analysis. Analysts may only need to perform certain tasks to achieve their goals.
Always document your findings and actions as you perform analysis so that you do not lose your place, duplicate analysis efforts, or forget how you did something.

Below is a flow chart representing the above analysis phases


Receive or discover object to be analyzed
Re-evaluate or determine reason for analysis
Document binary capability
Create malware mitigations, signatures/IOC, exploit, or tools to better analyze related software
Static Analysis
Reverse Engineering Workflow
Disassembly
Behavioral Analysis
Dynamic Analysis