

Final control question from Reverse engineering

1. What is Reverse Engineering?
2. What are reverse engineering tools and their functions?
3. Advantages and disadvantages of GDB or IDA Pro tools?
4. Advantages and disadvantages of OllyDbg and Radare2 tools
5. Step 1 of Reverse Engineering: How is the process of collecting information carried out?
6. Step 2 of Reverse Engineering: How is the process of analyzing program information carried out?
7. Step 3 of Reverse Engineering: How is the process of studying the analysis results carried out?
8. What is dynamic analysis?
9. What is an emulation?
10. What are the types of emulsions and their types?
11. What are Game Emulators and compare them?
12. What are ZSNES and DOLphin emulators?
13. What are mobile device emulators and explain them?
14. What are the operating system emulators and explain them?
15. What are Noxplayer and Bluestacks emulators?
16. What is QEMU emulator?
17. What are assembly interpreters?
18. What are the advantages and disadvantages of GCC (GNU compiler Collection)?
19. What are the advantages and disadvantages of MASM (Microsoft Macro Assembler)?
20. What are the advantages and disadvantages of TASM (Turbo Assembler)?
21. What is the structure of a serial number?
22. How is serial number encryption done?
23. What are Active Keys?
24. What are Dynamic serial numbers?
25. What are batch partitions?
26. What are the purposes of delimiters?
27. What is a delimiter based on the available size?
28. What are the types of errors?
29. What happens to the return address when a function is called?
34. How can you identify the number of arguments a function receives by examining its stack frame?
35. How can tools like GDB or IDA Pro help in analyzing stack-based function calls?
36. How can you observe the function arguments pushed onto the stack using a debugger (e.g., GDB)?
37. What happens if a function takes more arguments than it was designed to handle in a specific calling convention?
38. How do calling conventions like cdecl or stdcall impact the order of arguments pushed onto the stack?

39. What is the primary purpose of reverse engineering in the context of software systems?
40. Name three specific applications of reverse engineering mentioned in the text.
41. How does reverse engineering contribute to security analysis?
42. What is the relationship between high-level code, assembly language, and machine code?
43. Describe the role of compilers, assemblers, and disassemblers in the translation process.
44. What is a hex editor, and how is it used in reverse engineering?
45. In the practical example, what are the steps taken to analyze the "add" function in a C program?
46. How does a debugger help confirm the functionality of a binary program?
47. Explain how machine code represents operations, using the example of "MOV EAX, 1".
48. Why is assembly language referred to as human-readable compared to machine code?
49. What are the differences between static and dynamic analysis in reverse engineering?
50. Explain the role of a debugger in reverse engineering. Give an example of a commonly used debugger.
51. What is a binary file, and why is it important in reverse engineering?
52. What is software reverse engineering?
53. What is hardware reverse engineering?
54. How is reverse engineering applied to analyze malware?
55. What is a disassembler?
56. Why is reverse engineering important in cybersecurity?
57. How can reverse engineering be applied in cybersecurity?
58. What is assembly language?
59. What is a decompiler and how is it used?
60. What does static analysis mean in reverse engineering?
61. What does dynamic analysis mean in reverse engineering?
62. What is variable in programming languages?
63. In what scenarios would a developer need to use both environment variables and constants in a single program?
64. What is the definition of reverse engineering, and what are its primary purposes?
65. List at least three major applications of reverse engineering and explain their importance.
66. How is addition used in the context of reverse engineering, and can you provide an example scenario?
67. Explain the role of subtraction in software calculations involving resources or time.
68. Why is multiplication significant in reverse engineering, and what does it typically represent?
69. What is the purpose of division in resource allocation during reverse engineering?
70. What role do logical comparison operations play in decision-making processes within software?

71. Explain bitwise operations and discuss their importance in low-level programming and hardware interaction.
72. Describe conditional logic and its significance in controlling the flow of execution in a program.
73. What is the significance of understanding syntax and semantics in programming when analyzing compiled code in reverse engineering?
74. How do incorrect variable usage or type mismatches impact the behavior of a program, and why is this crucial in reverse engineering processes?
75. Why are algorithms considered the foundation of any software system, and how do reverse engineers benefit from analyzing algorithms?
76. How does the concept of modular programming through functions facilitate the reverse engineering of large software systems?
77. How can you identify a literal value from a string in a disassembled program?
78. how can you distinguish between a constant value and an offset in an instruction?
79. why is it important to make this distinction when analyzing machine code or disassembling a program?
80. What is the process for determining the type of a string (e.g., ASCII, UTF-8, wide character)?
81. How can you identify a global variable in a program during reverse engineering?
82. What markers in the disassembled code indicate the presence of a global variable?
83. How does the symbol table of an executable aid in identifying global variables?
84. How do static and dynamic linking affect the identification and tracking of global variables in compiled programs?
85. When examining disassembled code, how would you manually recover cross-references to global variables?
86. What assembly instructions or patterns are commonly used to identify such references?
87. How does contextual search of a global variable's offset help in understanding the variable's role within the program?
88. Can you provide an example of how an offset might appear in a specific instruction like MOV, ADD, or SUB?
89. When examining the memory layout of a program, what patterns or structures in the memory dump can indicate the presence of an object or a structure?
90. What techniques can you use to map out the connections between different instances of objects and their methods or member data?
91. What syntax or memory access patterns should you look for?
92. When analyzing a program's heap memory, how can you identify an object that has been instantiated from a class?
93. What is the purpose of the new keyword in C++?

94. How do you identify the use of new in a disassembled binary?
95. How is memory deallocation done with the delete keyword in C++?
96. How can you differentiate between new and malloc in a program's memory allocation?
97. How can you identify memory allocated on the heap in a program?
98. What happens when you call delete on a pointer in C++?
99. How can you identify standard library functions like malloc, free, new, and delete in disassembled code?
100. What are the common heap implementation approaches used in C++ for memory allocation?