Software Reverse Engineering for Security Course

Assignment 1

In this assignment you will analyze two ELF binaries: passwordCheckEasy (source code will also be provided for this one) and passwordCheckDifficult (no source code and the binary will be stripped), where both binaries take a list of command line arguments that are treated as password candidates. If one of the command line arguments matches one of the valid hardcoded passwords, each binary terminates with a ***Login successful*** message. Both passwordCheckEasy and passwordCheckDifficult will have statically (with a _static suffix) and dynamically (with a _dyn suffix) linked versions for your convenience. Please indicate which version you use in your answer. These files are provided under Assignment1 folder on CANVAS.

You are expected to examine these binaries using some Linux utilities that we have discussed in class including but not limited to gdb, readelf, objdump, file, checksec, and tools like Ghidra to answer the following questions. You are encouraged to provide screenshots of your analysis. Alternatively, you can explain the steps you took to answer the questions. Either way, you need to provide supporting evidence for your answers to get full credit.

1. (8 pts) For passwordCheckEasy and passwordCheckDifficult, answer the following questions:
    a. Is the binary using big endianness or little endianness for data encoding? Explain.
    b. What is the entry address for the binary?
    c. What is the entry address for the main function?
    d. How many functions are defined in the binaries by the main program (excluding the library functions) and what are their entry addresses?
    e. What is the base address of the environment variables array?
    f. Display the first environment variable setting using your answer to e).
    g. At which address is the .bss section located?
    h. What is the size of the .bss section? Which addresses in the .bss section are referred by the main program? Give example instructions from the main program that use those.
    i. Are there any local variables defined in any of the functions defined by the main program? If so, specify their addresses on the stack and show their initial contents.
    j. Find out how the parameter values are passed to each function defined by the main program and explain.
    k. Inspect the values of the parameter values for functions defined by the main program.
    l. Find the return address of each function defined by the main program.
2. (7 pts) For the passwordCheckEasy binary, draw the state of the stack right before the checkPassword function is called and right after the prologue of checkPassword has been executed. Explicitly show the value of rbp, rsp, the location of the return address, the saved frame pointer, the local variables, and the parameters on the stack. Provide a stack figure similar to the one provided in Lab2.pdf.
3. (10 pts) For one of the passwordCheckDifficult binaries (static or dynamic), find a command line argument list other than the hardcoded passwords that results in the termination of the binary with the ***Login successful*** message (yes, there is a backdoor!). Note that these binaries do not have any debugging symbols. *Hint*: You can assume that there are some commonalities between the passwordCheckEasy and passwordCheckDifficult binaries. Feel free to leverage

the source code for passwordCheckEasy and the decompiled version to have a better understanding of the passwordCheckDifficult binaries. To get full credit, you need to provide a command line argument list that will make the passwordCheckDifficult print the ***Login successful*** message along with a brief explanation of the logic that implements the backdoor.