Kevin Gamage

Assignment 3

Virtual Machine	Load the binary file into a virtual machine since we do not know exactly what it is or can do. It is possible that the file is malicious.
Goal	Determine the goal of reversing the binary file. Do we plan to obtain the source code of the file, change the control flow, or extract information from the file?
Binary File Type	Determine the type of the file. This can be done using different methods. The simplest would be to use the commands file, vim, or hexdump and search for the magic number in the file. Additionally, if you know that the file is an exacutable file then you can use objdump.
Big Endian / Little Endian	Using the same commands as for determining the file type, we can see whether the binary file is little endian or big endian.
Entry Point	We can find the entry point of the program depending on the type of the binary file. If it is an ELF we can use readelf to determine the entry point or examine the hexdump of the file and check the 4 bytes starting from the 0x18 offset.
Binary Analysis	At this step we decide which methods we plan to use to analyze and reverse the binary file. Dynamic analysis will have us run the binary file and see what the program is doing. Static analysis will have us examine the structure of the binary file to determine what the program is doing.
Dynamic Analysis	We can run the binary file and see what it does by stepping through the program and observing its behavior. There are a number of different tools that can be used to do this.
Static Analysis	We can view the structure of the binary file and attempt to decipher what it will do once executed. There are several different tools that can be used to do this as well.
objdump	objdump has many different uses in the topic of reverse engineering. It can be used to disassemble an executable binary file to retrieve its assembly code. This can allow us to obtain breakpoints in the execution of the file for use with debuggers like GDB. Additionally, we can use it to learn a great deal about the file such as its sections, headers, and variables. It can also be used to statically analyze the control flow of the

	program by determining where calls, jumps, and functions are executed in the assembly code.
strings	Using the strings command, we are able to see what strings are present in the binary file. These strings can possibly be critical data such as passwords or information that allows us to change the control flow of the program.
IDA / GHIDRA	Assuming we are reversing an executable binary file, we can potentially use GHIDRA or IDA to examine the file. Both tools have their pros and cons. For example, while IDA has a free version, GHIDRA is completely free and open-sourced. Both IDA and GHIDRA allow you to view the hex and disassembled versions of the binary file. They can additionally be used to view the critical sections of the program that modify the control flow as well as a Control Flow Diagram of the binary file. While GHIDRA has a decompiler that can recreate a version of the original source code, IDA does not offer that functionality in the free version.
GDB	Assuming that the binary file is executable, we can disassemble it using objdump and create breakpoints in the execution of the program using GDB. As the program executes, we can check the values present in the registers to find useful information. It can also be used to examine the control flow of the program as it executes.
strace / ltrace	In dynamic analysis, strace and ltrace can be used to show the system calls and library calls performed by a program. These can be useful in extracting keywords when comparisons are performed and for learning other valuable information from the program as it executes.
Critical Information	This can be information that is itself the goal of reverse engineering the binary file such as a password or it can be information that allows us to change the control flow of the program. It can also be information such as imported libraries.
Control Flow Diagram	The logical flow of the program which enables us to see which sections of the binary file are potentially visited in the execution of the program.
Source Code	The code that the binary file was originally compiled and assembled from.