Tanish Arora
CS 255 Computer Security
Homework #2

1. From the stack buffer overflow paper, what is NOP sledge and why it is useful?

Ans 1) NOP sled, also known as “No-Operation” sled is a set of no operation instructions which are meant to just bypass CPU’s instruction execution flow. Almost all processors have a NOP instruction, it is usually used to delay execution process in processor for the sole purpose of timing. It is useful as it helps in memory alignment, prevents hazards, to occupy a branch delay slot, to render void an existing instruction such as a jump, or as a place-holder to be replaced by active instructions later on in program development.

2. From the stack buffer overflow paper, what is the simplest way to find potential buffer overflow vulnerabilities?

Ans 2) The easiest/simplest way to find potential buffer overflow vulnerabilities is to understand the code written. Usage of languages such as C, C++ helps a programmer to preserve memory but bugs in the code written in these languages can lead to memory violations and potential buffer vulnerabilities. The first step is to detect buffer overflows in source code is understanding how they work, and the second step is knowing to look for external input and buffer manipulations, then the third step is to know what functions are susceptible to this vulnerability and can act as red flags for its presence.

3. From the StackGuard paper, we know stack canary can be bypassed, why we still use it virtually everywhere?

Ans 3) Stack canary can be bypassed, still the canary version is considered to be more performant while the guard version is more secure. The variant in canary can defend itself against guessing by exiting and replacing the attacked Canary-guarded daemon with a guard one, the Canary variant still consumes more CPU time than the generic program, it is overlapped with disk I/O, and the program completes in the same amount of real time. The MemGuard variants consume so much CPU time that the program’s real time is dramatically impacted.

4. From the ROP paper, could you explain how the ROP shellcode in Section 4 works?

Ans 4) Return Oriented Programming (ROP) is the general case of a technique often used when exploiting security vulnerabilities caused by memory corruption issues. The shellcode is the application of the above mentioned definition. The shellcode invokes the execve system call to run a shell, rest the shell code injects itself in order to avoid nul bytes from the null pointers. Word 1 sets the %eax to zero, words 2-4 helps in prepping the system call index, words 5 sets the second word in argv to Null.Word 6 sets %eax to 0x0b, and lastly 7-12 helps trapping into kernel.

5. From the CFI paper, what are the three requirements/assumptions for CFI enforcement?

Ans 5) CFI enforcement provides protection even against adversaries which control the data memory of the currently executing process. The three assumptions or requirements for CFI enforcement is Unique ID's in bit pattern, Non changeable or rewritable code (anything should not be able to modify or change the code during run time), and Non executable data (data cannot be corrupted while running the program).

6. From the CFI paper, what is the strategy mentioned in the paper that can improve the accuracy of CFG?

Ans 6) There are two strategies discussed in the paper in order to increase precision or accuracy of CFG, first one is code duplication, two code sets with different destinations can prevent the possibility of overlapping. Or the other method which is refining the instrumentation (more than one ID-check at destination sets) can also be used to increase the accuracy of CFG.