II. Understanding Countermeasures

1. **Address Space Layout Randomization (ASLR)**
    - *What is ASLR? How does it affect the stack?*
      ASLR is Address Space Layout Randomization and it randomizes the stack by a few bytes/pages.

    - *How can ASLR be bypassed without turning it off?*
      ASLR can be bypassed if an attacker is able to find a memory location "leak" which allows the attacker to know where the user's applications execute code inside the memory. The attacker can use that to their advantage by exploiting specific applications on the user's computer (Cimpanu, 2020).

2. **Stack Canary**
    - *What is a Stack Canary? How does it affect the stack?*
      A stack canary is a value that is placed on the stack when the program is started, before the return address is stored in the stack frame. It detects overflows when an attempt to rewrite the address is made.

    - *Are Stack Canaries vulnerable and if so, how?*
      Stack canaries are vulnerable to brute-force attacks. This typically happens when the attacker forks the child processes and can overwrite the canary ("Stack Canaries", n.d.).

3. **Strongly Typed Languages**
    - *Do they suffer from the same buffer overflow vulnerabilities? Yes or No?*
      No, strongly typed languages do not suffer the same buffer flow vulnerabilities.

    - *Explain why or why not.*
      They typically do not suffer the same buffer overflow vulnerabilities due to automatic bound checks and automatic memory management.

4. **Safer Functions**
    - *How could you replace the toy example's vulnerable function to be safer?*
      I would check the number of bytes entered by the user for the password.

III. Understanding Foundational Concepts

5. **GNU and GDB**
    - *What is the GNU compiler?*
      A GNU compiler compiles languages like C and C++.

- *What is GDB?*
- *GDB is a debugger for GNU and can run on Unix systems. It is used for languages like C and C++.*

- *How do you invoke the GNU compiler?*
  gcc

- *How do you invoke GDB?*
  gdb

6. GDB Commands

- *View the processor registers.*
  info registers ("Advanced gdb features," n.d.)

- *Set a breakpoint in code.*
  break funcl ("How do I use breakpoints," n.d.)

- *Find the address of an OS function.*
  Info address [symbol name] ("Info address command," n.d.)

- *Inspect a memory location.*
  x/FMT ADDRESS ("Advanced gdb features," n.d.)

7. GCC Compiler Flags

- *-O0 and its effect*
  Optimization for compilation time ("Gcc option flags," n.d.)

- *-g and its effect*
  Generates debugging info (Weimer, 2019)

- *-fno-stack-protector and its effect*
  Disables stack protection ("Compiler reference guide," n.d.)

- *-z execstack and its effect*
  Passed directly on the linker along with an executable stack ("Link options," n.d.)

8. ESP and EBP Registers

- *ESP's purpose*
  ESP points to the next available byte on the stack so it's purpose is to provide temporary space on the stack for partial operations.

- *EBP's purpose*
  EBP points to the top of the stack frame. It's purpose is to reference local variables and function parameters.

IV. Works Cited

1. Compiler reference guide. (n.d.). Retrieved February 07, 2021, from https://www.keil.com/support/man/docs/armclang_ref/armclang_ref_cjh1548250046139.htm#:~:text=%2Dfno%2Dstack%2Dprotector%20disables,array%20larger%20than%208%20bytes.
2. Advanced gdb Features. (n.d.). Retrieved February 07, 2021, from http://www.unknownroad.com/rtfm/gdbtut/gdbadvanced.html#REGISTERS
3. Cimpanu, C. (2020, September 14). New blindside attack Uses speculative execution to bypass ASLR. Retrieved February 07, 2021, from https://www.zdnet.com/article/new-blindside-attack-uses-speculative-execution-to-bypass-aslr/
4. Gcc option flags. (n.d.). Retrieved February 07, 2021, from https://www.rapidtables.com/code/linux/gcc/gcc-o.html
5. How do I use Breakpoints? (n.d.). Retrieved February 07, 2021, from http://www.unknownroad.com/rtfm/gdbtut/gdbbreak.html#BLINE
6. Info address command. (n.d.). Retrieved February 07, 2021, from https://visualgdb.com/gdbreference/commands/info_address
7. Link options (using the gnu compiler collection (gcc)). (n.d.). Retrieved February 07, 2021, from https://gcc.gnu.org/onlinedocs/gcc/Link-Options.html#index-z
8. Stack canaries. (n.d.). Retrieved February 07, 2021, from https://ctf101.org/binary-exploitation/stack-canaries/
9. Weimer, F. (2019, March 07). Recommended compiler and linker flags for gcc. Retrieved February 07, 2021, from https://developers.redhat.com/blog/2018/03/21/compiler-and-linker-flags-gcc/