Assignment 1

Peter Sheehan

R00197366

# Question 5

1. Canaries:  
   A Canary is a security mechanism used to detect stack-based buffer overflow attacks. A special value, called the ‘canary value’ is placed between the stack buffer and control data. If the canary value is altered during a buffer overflow attack, the program detects it before overwriting the return address and halts execution.
2. Effectiveness vs Format String Vulnerabilities:
   1. Strengths
      1. Protects against attacks that overwrite the return address directly since the canary is checked before returning from a function.
      2. Adds a layer of defence for stack-based attacks.
   2. Limitations:
      1. Format String vulnerabilities often do not directly overwrite the stack’s return address but instead leverage memory corruption. Canaries cannot prevent these attacks unless they explicitly involve the stack frame.
      2. If the attacker can bypass the stack canary by overwriting the memory of protection is ineffective.
      3. Advanced format string exploits can leak the canary value and replicate it, bypassing the protection.
3. Address space layout randomisation
   1. Definition
      1. ASLR randomises the memory addresses of key program areas. (e.g. stack, heap, shared libraries) each time a program is run. This makes it difficult for an attacker to predict the addresses they need to exploit a vulnerability.
      2. Effectiveness vs. Format String Vulnerabilities:
         1. Strengths:
            1. Randomises memory layout, making it difficult to find predictable memory addresses required for format string exploitation.
            2. Reduces the chance of successfully leveraging predictable return addresses or function pointers.
         2. Limitations:
            1. ASLR can be bypassed If an attacker uses a format string vulnerability to leak memory addresses, which are then used to compute other addresses in the memory space.
            2. Partial overwrites (e.g., overwriting only the least significant bytes of an address) can still work if the attacker guesses correctly.
4. Executable Space Protection
   1. Definition:
      1. **Executable Space Protection** prevents execution of code from memory regions that are meant to hold data (e.g., the stack or heap). This blocks attacks that inject malicious code into writable memory, like stack buffer overflow attacks.
   2. Effectiveness vs. Stack Buffer Overflow Vulnerabilities:
      1. Strengths
         1. Prevents the execution of injected shellcode by marking data segments as non-executable.
         2. Forces attackers to use advanced techniques, such as return-oriented programming (ROP) or jump-oriented programming (JOP), increasing the complexity of attacks.
      2. Limitations:
         1. Does not prevent attackers from corrupting control data (e.g., return addresses or function pointers) and redirecting execution to existing executable code (e.g., ROP gadgets).
         2. if combined with a format string vulnerability, an attacker can potentially bypass DEP by manipulating the program's memory to disable the protection or use legitimate executable memory areas for their payload.
         3. Some legacy systems or applications may not enable NX/DEP, leaving them vulnerable.