Nanyang Technological University

AY2021-2022 Semester 2

CE/CZ4067 SOFTWARE SECURITY

Tutorial 3: ASLR and JOP

1. Compile and run the program shown in Fig. 1. Use a debugger (e.g., `gdb`) to analyze the stack layout of the program. Document the system architecture and compiler options you used.

```c
#include <stdio.h>

int read_req(void) {
  char buf[128];
  int i;
  gets(buf);
  i = atoi(buf);
  return i;
}

int main(int ac, char **av) {
  int x = read_req();
  printf("x=%d\n", x);
}
```

Figure 1: A simple C program with a buffer allocated on the stack.

(a) What is the memory address of the buffer in `read_req`, i.e, `&buf`?

(b) What is the memory address of the variable `i`?

(c) What is the return address (saved EIP) of `read_req`?

(d) Fill in the stack diagram shown in Fig. 2 according to the state before returning from `read_req`. You must include `buf`, `i`, `x`, as well as saved EIP and saved EBP of `read_req`, according to their relative positions.

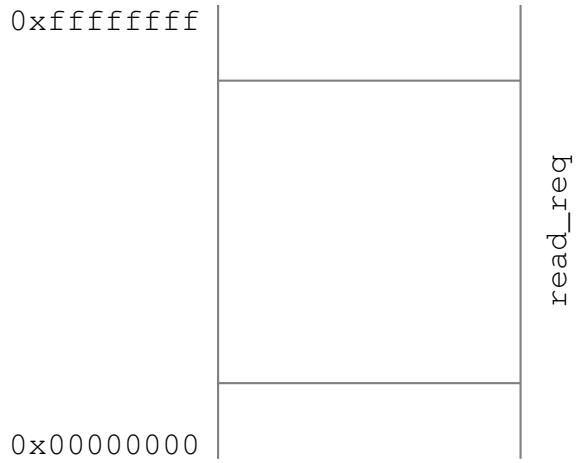

Figure 2: Stack diagram of the simple C program.

2. Address space layout randomization (ASLR) is a memory-protection process for operating systems that guards against buffer-overflow attacks by randomizing the location where system executables are loaded into memory. Collect information on ASLR in Linux and in Windows.

   (a) What is being randomized in each operating system?

   (b) Entropy is generally defined as the amount of disorder. Entropy is also a metric used to evaluate randomization techniques. It shall be considered as the level of uncertainty that an attacker have about the location of a given object. The function `randomize_stack_top` shown in Fig. 3 is used to perform stack randomization in 64-bit Linux. How much entropy is introduced for the stack?

```
1  static unsigned long randomize_stack_top(unsigned long stack_top)
2  {
3  unsigned long random_variable = 0;
4
5  if ((current->flags & PF_RANDOMIZE) &&
6  !(current->personality & ADDR_NO_RANDOMIZE)) {
7    random_variable = (unsigned long) get_random_int();
8    random_variable &= STACK_RND_MASK; // 0x3fffff on x86_64
9    random_variable <<= PAGE_SHIFT; // 12 on x86_64
10 }
11 #ifdef CONFIG_STACK_GROWSUP
12 return PAGE_ALIGN(stack_top) + random_variable;
13 #else
14 return PAGE_ALIGN(stack_top) - random_variable;
15 #endif
16 }
```

Figure 3: The `randomize_stack_top` function in the file `fs/binfmt_elf.c`.

3. Jump-Oriented Programming (JOP), is similar to Return-Oriented Programming (ROP). In an ROP attack, the software stack is scanned for gadgets that can be strung together to form a new program. ROP attacks look for sequences that end in a function return (RET). Jump-oriented programming creates its own trampoline for linking gadgets instead of using the call stack.

   (a) Read the research paper [1] on JOP, and explain the fundamentals of the attack method.

   (b) What are key advantages of JOP comparing with ROP?

References

[1] Tyler Bletsch, Xuxian Jiang, Vince W Freeh, and Zhenkai Liang. Jump-oriented programming: a new class of code-reuse attack. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pages 30–40, 2011.