CS 260 Binary Analysis

Dr. Heng Yin

Associate Professor

Department of Computer Science and Engineering

UC Riverside

Syllabus

- Check course homepage:
  - https://www.cs.ucr.edu/~heng/teaching/cs260-winter19/index.html

What is binary analysis?

Why binary analysis?

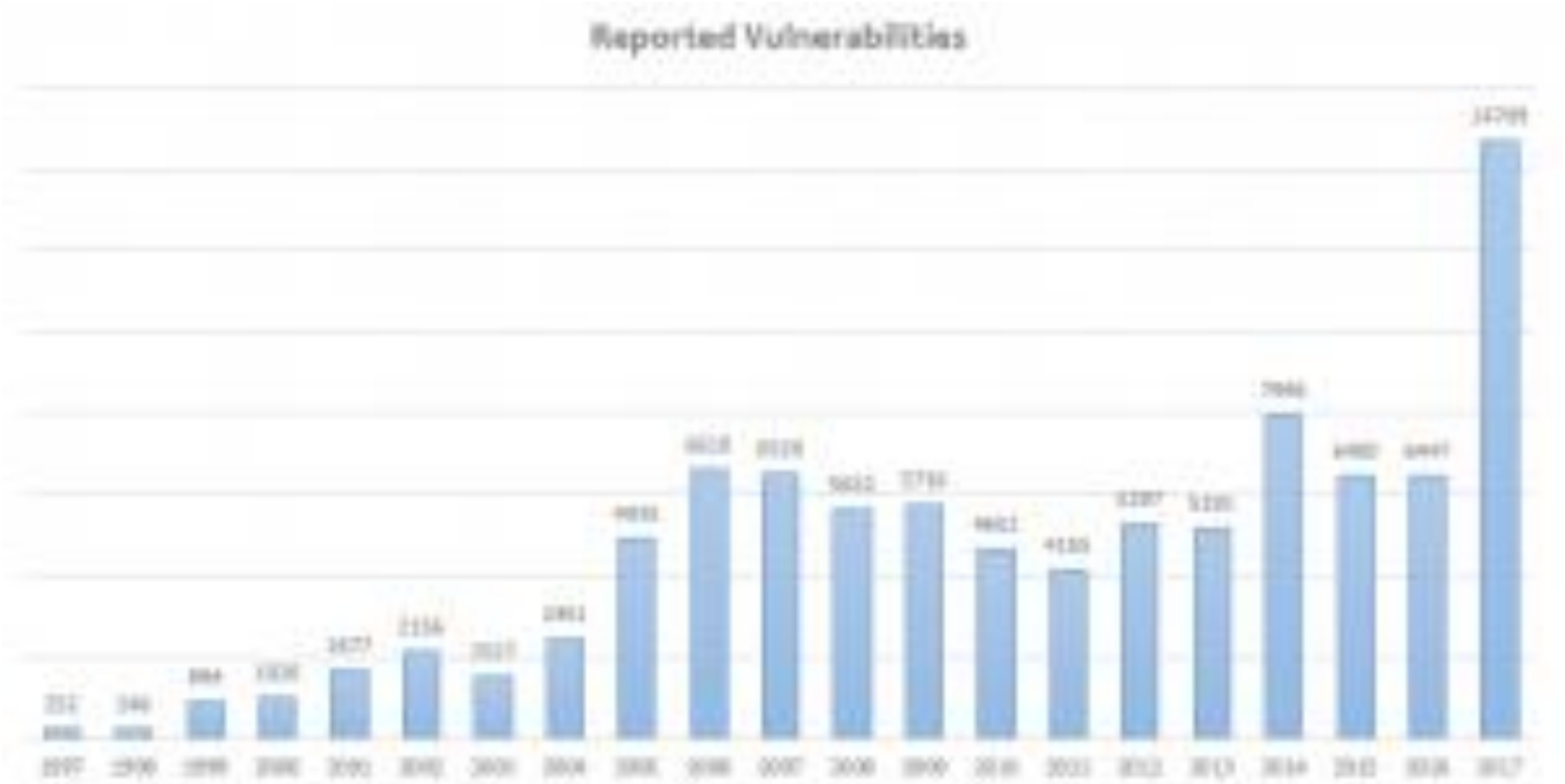
- Programs may be vulnerable
- Programs may contain malicious logic
- Understand a piece of unknown code

Vulnerability Analysis

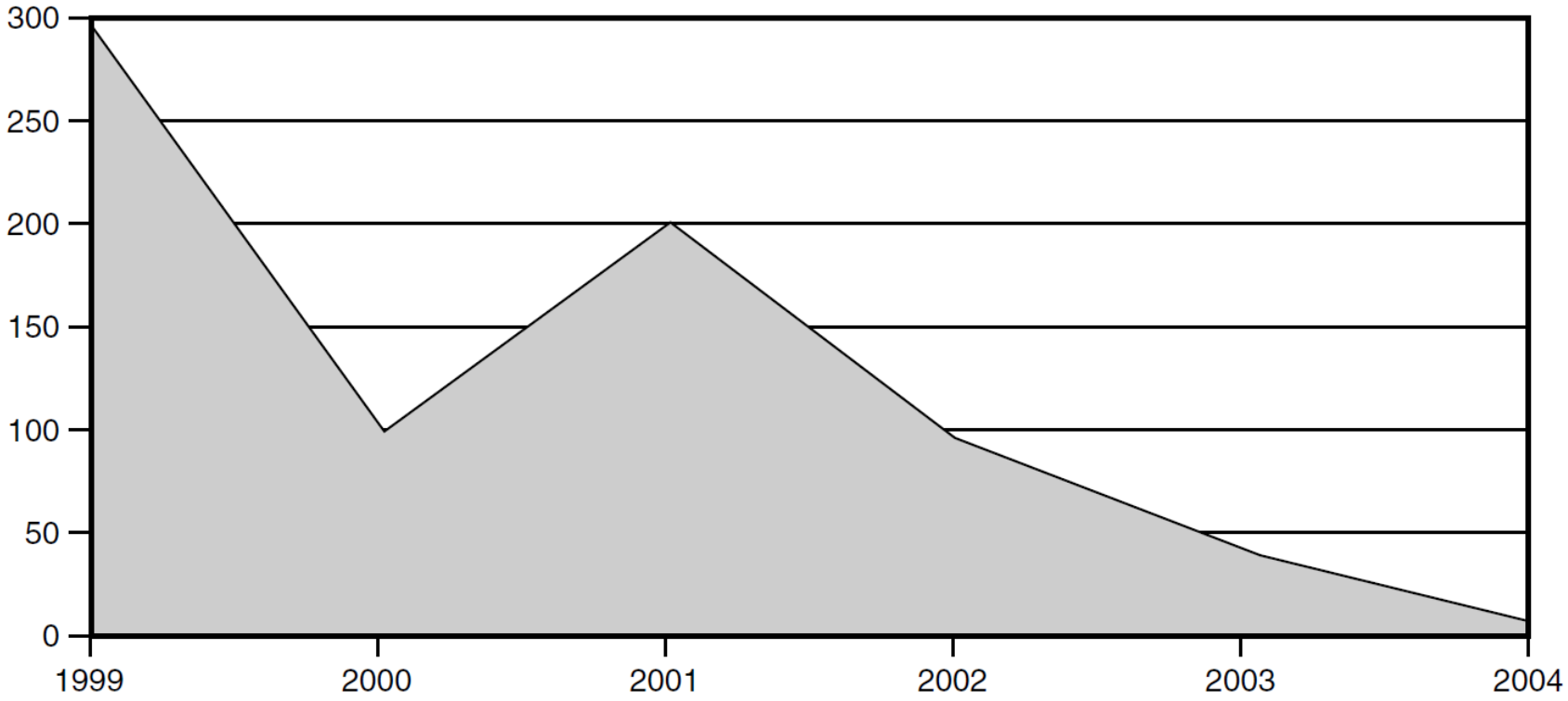
Software Vulnerability

- Common vulnerabilities:
  - Buffer overflow
  - Dangling pointer
  - Format string bugs
  - Time-of-check-to-time-of-use bugs
  - Symbolic link races
  - SQL injection
  - Directory traversal
  - Cross-site scripting
  - Cross-site request forgery
  - …

Vulnerabilities discovered per year



Days from patch to exploit (information security, July 2004)



Example - Exploit Buffer Overflow Vulnerability

```
1. int IsPasswordOkay(void) {
2.   char Password[12];
3.   gets(Password);
4.   if (!strcmp(Password, "goodpass"))
5.     return(true);
6.   else return(false);
7. }

8. void main(void) {
9.   int PwStatus;
10. puts("Enter password:");
11. PwStatus = IsPasswordOkay();
12. if (PwStatus == false) {
13.   puts("Access denied");
14.   exit(-1);
15. }
16.   else puts("Access granted");
17. }
```

Binary Code Comprehension

- Executable and Linkable Format
  - https://linux-audit.com/elf-binaries-on-linux-understanding-and-analysis/
  - Text, data, rodata, bss
- Calling Conventions
  - https://en.wikipedia.org/wiki/Calling_convention
- Stack Layout
  - https://eli.thegreenplace.net/2011/02/04/where-the-top-of-the-stack-is-on-x86/
- Relocation and Position-Independent Code
  - https://en.wikipedia.org/wiki/Relocation_(computing)
  - https://eli.thegreenplace.net/2011/11/03/position-independent-code-pic-in-shared-libraries/
- C++ Internals (will be discussed later)

Preparation

- echo 0 > proc/sys/kernel/randomize_va_space
- gcc –fno-stack-protector example01.c –o example-01

Problem 1

- Craft a malicious input to bypass the authentication:
  - Print “access granted” instead of “access denied”

Problem 2

- Inject arbitrary code to execute
  - A shell code template is given
  - Make a working exploit that runs “ps”

Problem 3

- Return into an existing function in libc
  - Make a working exploit that runs “ps”