CS964-Midterm Exam

Graded

Student

Mohammed Jawed

Total Points

49 / 50 pts

Question 1

Control Hijacking

14 / 15 pts

✔ **+ 15 pts** Correct

+ 12 pts Minor mistakes

+ 9 pts Partial correct

+ 6 pts Partial correct

+ 3 pts partial correct

+ 0 pts Click here to replace this description.

− 1 pt ii Ans: Importance of gets() function is not mentioned

Question 2

Linux

15 / 15 pts

Click here to replace this description.

✔ **+ 5 pts** Correct

+ 3 pts Partial correct

+ 0 pts Incorrect

Click here to replace this description.

✔ **+ 5 pts** Correct

+ 3 pts Partial correct

+ 0 pts Incorrect

Click here to replace this description.

✔ **+ 5 pts** Correct

+ 3 pts Partial correct

+ 0 pts Incorrect

Question 3

Race Condition

10 / 10 pts

Click here to replace this description.

- ✔ **+ 5 pts** Correct
- **+ 3 pts** Partial correct
- **+ 0 pts** Incorrect

Click here to replace this description.

- ✔ **+ 5 pts** Correct
- **+ 3 pts** Partial correct
- **+ 0 pts** Incorrect

Question 4

Rootkits

10 / 10 pts

- ✔ **+ 10 pts** Correct
- **+ 7 pts** Partial correct
- **+ 5 pts** Half correct
- **+ 3 pts** Attempt
- **+ 0 pts** Click here to replace this description.

Q1 Control Hijacking
15 Points

Consider the following code fragment:

```
#include <stdio.h>

void functionCopy() {
    char textLine[10];
    printf("Enter your line of text: ");
    gets(textLine);
    printf("You entered: %s", textLine);
}

int main() {
   functionCopy();
    return 0;
}
```

(i) Give an example input to this program that would cause a buffer overflow.

in the given code snippet, the length of variable "textLine" as 10 character, which is fixed.
However, before performing operation "gets" the code snipe doesn't check the length of inputted line of text which will cause overflow(if entered inputs >= 11 characters) as gets() funtion will write data beyond the bounds of the buffer.

(ii) Explain how your input will cause a buffer overflow?

lets say we have entered inputs as "AAAAAAAAAAA", and "textLine" has space for 9 characters plus a null terminator.

Buffer overflow to occurs with the provided input as gets() funtion:
- the input "AAAAAAAAAAA" will tries to store in "textLine".
- The funtion does not check if the inputs exceeds the size of the buffer , as a result it writes all 11 char into "textLine".
- the write operation surpasses the buffer's boundary, which is designed to hold only 10 char. the extra char spill over into the adjacent memory and this can corrupts data in nearby memory locations or even alter program flow or can be use to execute arbitrary code.

(iii) List 3 most critical security threats that can be realized exploiting buffer flow vulnerabilities in application programs.

Following are 3 most critical security threats that can be exploited,
1. Privilege Escalation: with the buffer overflow the program that has higher privileges than the user running it , then malicious actor can perform actions by bypassing normal security checks and gain elevated access to system resources.
2. Arbitrary Code Execution: An attacker can carefully craft input that includes executable as shell code and use the overflow to overwrite a program execution path to execute crafted shell code to gain control over system.
3. Denial of Service(DoS): using buffer overflow, an attacker can cause the program to crash by corrupts the memory.

(iv) What are the possible run-time protections you are familiar with that are available to protect against such programming flaws so that the application cannot be easily exploited by a cyber attacker?

few of the run-time protections which was thought during prof.Sandeep lectures videos as well as it was part of HW also:
- Stack canaries
- ASL(Address Space layout Randomization)
- Data Execution Prevention
- Control Flow Integrity(CFI)

(v) What are the possible compile-time protections you are familiar with?

Some of compiletime protection which i am using it with our DevOps pipeline are:
- Static Code Analysis
- Fuzz testing
- Stack canaries
- Code Access Security (CAS)

Q2 Linux
15 Points

In modern Unix file system, if you have a binary executable, how do you check if it is a setuid program?

we can use command such as "ls -l" followed by the executable's name and can look for the permission in the output; a setuid file will show as "s" in the user's permissions sections, e.g -rwsr-xr--". the "s" in output command indicates that the files is set to run with the owner's privileges when executed by any user.

WHat is the purpose of saved userID when you have real userID and effective userID?

The main purpose of the saved user ID is to enhance secuirty by minimizing the duration and scope of privilege operation within a program by allowing a process to switch back and forth between privileged and non-privileged modes.

Explain how a setuid program if not written carefully may pose a danger of privilege escalation?

The program can be exploited through vulnerabilities such as improper input validation or buffer overflows, which may execute arbitrary code or modify system files under elevated privileges.
This can lead to privilege escalation, where a non-privileged user gains unauthorized access to privileged operations, potentially compromising the entire system.

Q3 Race Condition
10 Points

(i) What is a race condition?

A race condition is a flaw that occurs when the output or behavior of a system depends on the sequence or timing of other uncontrollable events.

(ii) Explain the TOCTOU vulnerability.

TOCTOU vulnerability- when a program's actions become unsafe due to changes in conditions between checking a resource and using it.

Q4 Rootkits
10 Points

(i) What are the common methods to check for rootkits in your system?

Common methods to check for rootkits
- specialized anti-rootkit software that scans for discrepancies in system calls and file system inconsistencies, and comparing cryptographic hashes of system files with known good hashes to detect unauthorized changes.
- analyzing system behavior for hidden process using tools like process explorer,
- monitoring network traffic for unusual activity.
Few Anti-rootkit which i have heard of : GMER, Sophos RootKit removal, McAfee RootkitRemover.

(ii) What do the rootkit designers do to avoid being detected by the common methods of rootkit detection?

Designers uses techniques such as kernel-level manipulation to hide processes, files and registry keys,.
They also uses polymorphic code to frequently change their signatures, thereby evading detection by traditionally antivirus software(which relies on signature based).
Furthermore, they can go to an extend to exploit driver vulnerabilities to operate directly beneath the operating system level.