Homework 2 for CS 165 (Fall 2021)

Due: on ilearn by the end of day on Dec 2, 2021

Steven Truong

Instructions:

***** Be brief in your answers. You will be graded for correctness, not on the length of your answers.

* Remember to submit online through ilearn if you didn't turn it in in-class. Paper copy will not be accepted.

I. Answer the following multiple choice questions (one or more correct answers). (1 point * 12)

1. Which of the following types of locations in a process address space where buffer overflow can occur? **__a, b__**

a) Stack

b) Heap

c) Code

2. Which of the following are defenses against stack buffer overflow? **__a,b,c___**

a) Address Space Layout Randomization (ASLR)

b) DEP/Non-executable stack

c) Control-Flow Integrity (CFI)

3. Which programming languages are vulnerable to stack buffer overflow? **___a___**

a) C

b) Java

c) Assembly

4. How can a stack buffer overflow hijack the control flow of the program? **___a, b___**

a) Overwriting the return address on the stack

b) Overwriting a function pointer on the stack

c) Overwriting a local boolean variable on the stack

5) With DEP defense enabled, which of the following becomes impossible? ___**b**____

a) Overwriting the return address on the stack

b) Injecting shellcode onto the stack and execute it by jumping to it

c) Finding a useful gadget to jump to in Return-Oriented Programming (ROP)

6) Which defenses have been (partially) deployed in modern operating systems? __**a, b**___

a) Access control list

b) ASLR

c) Control-Flow Integrity (CFI)

7) Which are the reasons why blind ROP attack against a web server works despite the fact that all modern defenses are deployed? ___**a, b**___

a) Web server forks a child process with the same address space layout every time to serve a new connection

b) Stack canary value stays the same even if a guess is wrong

c) The version of enabled ASLR does not provide sufficient randomness

8) How does the blind ROP attack determine if a code sequence contains the desired gadget (since it's blind)? ____**c**___

a) It learns the address of the gadget by obtaining a copy of the binary beforehand

b) It sets up the stack in special ways so that the detected gadget will be uniquely identifiable

c) It leverages the feedback about whether a server has crashed or not

9) Which of the following of x86 architecture makes overlapping instructions possible (i.e., one can jump to the middle of an instruction and the CPU can recognize it as a different yet valid instruction? **__b____**

a) Instruction lengths are not multiples of 8 bits

b) Instructions have variable length

c) Most byte sequences are legal instructions

10) What are some examples that conceptually map to the BLP or Biba model? **_a, b, c___**

a) In buffer overflow, command line argument is considered a low-integrity object. A root process is considered a high-integrity subject that should not be allowed to read the low-integrity data (thus allowing control flow to be hijacked)

b) In time-of-check, time-of-use attack, the file or directory controlled by an attacker is considered a low-integrity object. A root process is considered a high integrity subject that should not be allowed to read the low-integrity data (thus being tricked to perform unintended operations)

c) In directory traversal attack, the passwd file is considered the high-secrecy object. A root process (web server) is considered a low-secrecy subject since it needs to read the public HTML files and serve them to clients. A low-secrecy subject should not be allowed to read a high-secrecy object (thus leaking the passwd file unintendedly)

11) Which of the following about resource access attacks are correct? **_a, b,c ___**

a) They are caused by violations of BLP or Biba security policies.

b) They are caused by mismatches of expectations (e.g., high-integrity subjects expect high-integrity objects but mistakenly got low-integrity objects).

c) We need to look at both the code and access control policy to identify resource access attacks.

12) In computer security, there's a well-known principle called principle of the least privilege. The idea is that every subject (process, user, program) should have access to only the information and resources they absolutely need (no more should be allowed). Which of the following are correct based on your judgement? **_a, b, c___**

a) The reasoning behind the principle is to prevent an attacker to compromise a subject

b) Not running processes as root when not necessary (e.g., chrome or firefox) is a one example of principle of least privilege

c) The reasoning behind the principle is to reduce the damage once a subject is compromised

II. Consider the following code with a race condition vulnerability. The process_request() is the entry function where a program takes user input of request. The pid parameter denotes the id of the requesting process (e.g., different requesting processes would therefore cause different pids to be passed as parameters). The security_check() tries to use the pid to look up the uid of the requesting process and confirm if it is a root process. Can you explain how this code can be exploited to bypass the security check? (2 points)

```
void process_request(int request, int pid)  {
        if(security_check(pid) == true)    perform(request);
}

bool security_check(int pid) {
        string file = "/proc/" + pid + "/status";
        int fd = open(file);   // open the proc file for pid
        int uid = read_uid(fd);  // obtain the uid by reading the content of the file.
        If(uid == 0)  // if it is root process
                return true;  //passed the security check
        return false;
}
```

After a root process calls process_request(), a context switch may occur after security_check() gets called. The if statement condition will evaluate to true, and an attacker can bypass the security check when the OS context switches to the attacker's process and performs the attacker's request.

III. Briefly answer the following questions (1 point * 6)

1. Give two reference monitor examples where one is an inline reference monitor and the other is not.

Inline Reference Example: The canary defense for buffer overflows. It is added to a user program during compilation so it's inline.

Not Inline Reference Example: The kernel is a reference monitor that can enforce policies on user programs and files. It is not inline because the resource monitor does not live in the code of the user program.

2. What are the pros and cons for static vs. dynamic analysis (e.g., fuzz testing) for bug finding?

Static Analysis: identifies all vulnerabilities but has many false positives.

Dynamic Analysis: Can't find all vulnerabilities with limited number of inputs but being selective with inputs can reveal behavior that an adversary might try to evoke in a program.

3. In x86 and other modern CPUs, the processors often have different modes of privileges (e.g., kernel and user). If we think of the kernel code as high-integrity subject, kernel data as high-integrity data, user code as low-integrity subject, user data as low-integrity object, then according to the Biba policy, kernel code should not read user data. Give at least one reason why this may not be feasible in practice.

When user programs execute a system call and pass in a system call number, the kernel code would not be able to read the system call number under the Biba policy. This may not be feasible in practice because each system call would need to be mapped to a particular assembly instruction since system call numbers can't be passed to the kernel through registers.

4. In control-flow integrity (CFI), describe why an attacker cannot circumvent the checks that are inserted for every control transfer?

In control-flow integrity, a CFG is created at compile time and referenced to verify that a control transfer in a program hasn't been tampered with. An attacker can't stop this check because the CFG is created at compile time and can't be changed.

5. Describe the relationship between the return-to-libc attack and ROP attack.

In ret2libc and ROP, the return address of some function in a program is overwritten using a buffer overflow exploit. Ret2libc relies on finding the location of system() from libc and overwriting the return address to execute system("/bin/sh"). ROP relies on the use of gadgets to set up the correct values in registers, and using a final int 0x80 gadget to call execve("/bin/sh", 0, 0). Both attacks try to spawn a shell.

6. In network security, we discussed the following three threat models: 1) eavesdropper, 2) MITM, 3) Off-path. Please list them from the most realistic (how easy it is to achieve the threat model) to the least realistic.

Eavesdropper, Off-path, MITM

IV. Information flow is a useful mechanism that can enforce a variety of security policies that are hard to describe in simple access control lists. Describe how you can use information flow as a building block to catch the following attacks or vulnerabilities. Please specify the information sources and sinks that one should track (1 point * 4):

1. Link following attack where a victim process reads a file from an attacker-controlled directory.

Can check if a lower integrity file is a symbolic link to a high integrity file to catch such an attack.

2. Skype app on Android accidentally stores the password in plaintext in a file that's accessible to everyone.

Password is obtained when a user logs in on the skype app. Before writing to the file, a check can be made to ensure that the password is hashed first.

3. A buffer overflow attack that overwrites a function pointer.

During compilation, a CFG can be created to determine a set of addresses that a function pointer may hold. The address of this function pointer can be compared with this set at runtime to ensure that the address is legal.

4. The side channel enabled DNS cache poisoning attack.

Check for connections that keep reaching the global rate limit to catch the attack.