1. Exploiting a stack based vulnerability is easier to exploit compared to a Heap based vulnerability since stack memory allocation happens on consecutive locations. This causes the function data like local variables to have a consecutive memory address, allowing them to be overwritable if the canary is bypassed or is not present. On the other hand, Heap based vulnerabilities will require that the attacker performs a series of allocations and deallocations due to random memory addresses of the data, causing them to happen in a specific patter to gain access to the local variables or other data.

2. For the given program, while getting the input from the user, the first “gets(user)” call will take input from the user. If the user passes a string that is greater than 8 bytes in length it will overflow from the buffer of “user” variable which is of 8 bytes and will overwrite the value in the “hash” variable. For example, attacker enters AAAAAAAABBBBBBBB as input which will cause the buffer to overflow and overwrite the hash buffer. The next comparison which is “hashpw(pw) == hash” will get converted to “hashpw(pw) == BBBBBBBB”

This will lead to an attacker gaining access to any user’s account without knowing their password. The vulnerability lies in the unbounded input that “gets” function takes. Since there are no validations performed on the size of the input, this is a critical vulnerability in the given program.

Diagram

Description automatically generated

3.

![A picture containing text, newspaper, receipt

Description automatically generated]()

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ciphertext | y | y | j | t | h | r | h | c | f | f | n | l | x | s | i | y | k | h | l | v |
| Key | c | u | r | a | h | e | e | c | u | r | a | h | e | e | c | u | r | a | h | e |
| Plaintext | w | e | s | t | a | n | d | a | l | o | n | e | t | o | g | e | t | h | e | r |

4.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Full Form** | **Example** | **Mitigation** |
| S | Spoofing | A malicious actor gaining access to an account or service by using a legitimate user’s identity | Verifying every user’s identity in someway when they interact with the system.  Examples:   * Using biometric or retina scan for verification * Using Multi-Factor Authentication while performing any critical change like login, account updates, etc. |
| T | Tampering | A malicious actor modifying the data while it flows through the application environment | Verifying the integrity of the data that flows through the application environment.  Examples:   * Using hashes to verify the integrity of data * Using a secure channel for transferring data |
| R | Repudiation | A user denying that they made modification to the data that was used as part of the application | Implementing a secure logging mechanism which records all the events in the application environment  Examples:   * Using appropriate logging levels inside an application environment for the events * Secure storage of logs with backups |
| I | Information Disclosure | A malicious actor gaining access to confidential or private data that shouldn’t be available to them | Assigning appropriate roles to the users present in the application environment and restricting access to data based on them.  Examples:   * Using OAuth to authorize users and approve appropriate permission for data access * Assigning roles to the users in the system |
| D | Denial of Service | An attacker flooding the server or application with requests such that a legitimate user is not able to access the service | Adding Rate limiting or throttling options  Examples:   * Adding throttling to the API calls that are coming in * Adding more bandwidth to the system access like horizontal or vertical scaling options |
| E | Elevation of Privileges | A malicious user gaining a higher-level privilege than the regular user | A user should be assigned minimal privileges aka Principle of Least Privilege  Examples:   * Adding access control list * Adding more security to the privileged users like admin in terms of verification |

5. 1. ASLR or Address Space Layout Randomization is a service provided by \*nix based operating systems to randomize the stack address of the running program. When ASLR is enabled, it maps the program’s stack variables address to a location that is different on each run of the program.

2. ASLR came into picture to prevent stack-based buffer overflow attacks which relied on predictable memory addresses. With ASLR, it was difficult to identify at which memory location was the program stack being created as they are moved in a different space on every run of the program.

6. Data Execution Prevention is a memory mechanism implemented for preventing malicious programs to not be a threat to the system. This is achieved by making sure that the memory usage is secure and bounded or else it will terminate the program and the user is notified about the event.

DEP utilizes the CPU to mark locations in the memory where the attributes are set to non-executable. When the program tries to work abnormally or maliciously, a system level exception is passed on to the operating system’s security mechanism.

7. The two uses of asymmetric cryptography are:

1. Digital Signature – Using the private key to sign a software or document which makes sure that the information is provided by the user who signed it. This is confirmed by using their public key, which verifies the integrity.

2. Diffie-Hellman Key Exchange – While a secure channel is required to prevent any eavesdropping, a secure way is required to share the keys between the two parties beforehand. To do that, Diffie-Hellman uses a key exchange protocol to share the keys securely between the two parties before a new secure connection for communication is established.

8. Given:

Ciphertext: **yvuhgvanyhqpundusipp**

a = 3, b = 8

Calculate:

m = 26

a^-1 = 23

Using the formula “p(x) = a^-1(x – b) mod m” where x is each character in the cipher text we get the plain text: **oneringtorulethemall**

9. One-Way Function: A function that generates a cryptographically secure random sequence of characters. This is easy to compute for any given value, but inversing is computationally difficult.

Trap-Door Function: A one-way function which can take in additional data as part of input or implementation that can help in reversing the generated sequence.

10. The boot process of x86 based environment is:

- The first instruction by the processor is to check for BIOS and execute it.

- BIOS in-turn will make a call to the Bootloader (like grub for linux based OS).

- The bootloader is responsible for loading the kernel and other operating system modules.

- Once the kernel is loaded, the init file is executed which sets up the environment for the user.

Circumventing security checks during the boot process can be done by injecting malicious code in Master Boot Record boot codes in BIOS, modifying kernel boot management files or drivers or firmware, side-channel based attacks to leak memory from the previous run of operating system.

11. NOP Sled is Non-Operation which can be used to increase the number potential offsets in the memory. The instruction indicates that no operation is to be performed when the program encounters this. The opcode for NOP is 0x90. Example shellcode[]=”\x90\x90\x90\x31\xdb\xb0\x01\xcd\x80”

12. The code is susceptible to Format String vulnerability which can lead to reading of data in the stack. To fix this, add a %s format specifier as part of printf

int func(char \* user)

{

printf(“%s”, user);

}

13. The code is susceptible to Format String vulnerability which can lead to reading of data in the stack. To fix this, add a %s format specifier as part of printf

int func(char \* user)

{

printf(“%s”, user);

}

14. The first three output lines display the file name(test.txt), owner(jdoe), and owning group(staff) of the directory.

The User octet has read and write permissions whereas group octet and other octet has no permissions.

The output of getfacl reflects the mapping of permission bits and ACL entries.

15. List of items:

1: Request (Data Flow)

Threat: Data tampering

2: Response (Data Flow)

Threat: Man-in-the-middle attack

3: HTTP web server (Process)

Threat: Cross-Site scripting, RCE

4: MySQL Database (Storage)

Threat: SQL injection, Insecure data store

16. A hash is generated on a message and is encrypted with user's private key and attached to the message. The recipient on receiving the message decrypts the hash using sender's public key and computes the hash of message and compares to the hash appended to the message.﻿