- **(At least 200 words) Describe “9.5.1 – Buffer Overflow Attacks”.**

Buffer overflow attacks refer to a type of security vulnerability that occurs when a program writes more data to a buffer than it was originally designed to hold. This can be dangerous in some programming languages like C and C++, where memory management is manual and there are no built-in bounds checks unlike languages such as Python or java where they do this automatically.

When a buffer receives more input than its allocated space, the excess data can overflow into adjacent memory locations. This may overwrite critical data, including function return addresses, local variables, or control data, leading to unpredictable behavior. Attackers can exploit this issue by injecting malicious input, such as shellcode, that overwrites the return address with a pointer to their code. This enables the attacker to hijack the program’s control flow and execute potentially harmful commands to the system.

To reduce these attacks, developers can use techniques such as bounds checking to validate input size, address space layout randomization (ASLR) to shuffle memory addresses, and non executable stacks to prevent execution of injected code. stack canaries known values placed before return addresses can detect changes caused by overflow and halt execution if tampered with.

Even with these current defense techniques, buffer overflow attacks remain relevant today

- **(At least 200 words) Describe “9.5.5 – Null Pointer Dereference Attacks”.**

Null Pointer dereference attacks are a type of vulnerability where a program tries to access memory through a pointer that is set to null or None type (in other words it points to nothing). In C and C++, this can lead to unexpected crashes or even be exploited by attackers in certain situations, especially in kernel-level code or low-level system programs this is very easy to accomplish.

When a program dereferences a null pointer, it typically causes a segmentation fault and crashes. However, in some systems, particularly older ones or poorly secured environments attackers can take advantage of this by tricking the program into jumping to a controlled area of memory. For example, if the operating system allows memory to be mapped at address zero (which is usually where a null pointer would point), the attacker can inject malicious code there. When the program dereferencing the null pointer, it ends up executing the attacker's payload or thing they want to execute.

- **(At least 200 words) Describe "9.5.9 – Double Fetch Vulnerability".**

Double fetch vulnerability is a competition condition security issue that appears when a kernel mode process retrieves data from user space multiple times without proper synchronization or validation between accesses. This vulnerability is a specific type of time of check to time of use issue, where an attacker can manipulate data between the kernel's initial validation and its current use, leading to memory corruption, or often system crashes. The problem occurs because the kernel first fetches a user supplied value (such as a pointer, buffer size, or flag) to verify its validity, then fetches it again later to perform an operation

If a malicious user-mode thread modifies the data between these two fetches (validation and the current use), the kernel may act on inconsistent or maliciously altered data, bypassing security checks. As an example, if the kernel checks a buffer size and later uses it for a memory copy, an attacker could increase the size after validation, causing a buffer overflow.

To counteract this vulnerability, developers can copy user data into kernel space once and validate it there, use synchronization mechanisms like locks to prevent race conditions, and use hardware protections such as supervisor mode access prevention in summary double fetch vulnerabilities are particularly dangerous primarily in operating systems and drivers, as they can compromise security mechanisms and enable exploitation of kernel-level privileges.

2. (p. 698) question#8

Represent the ownerships and permissions shown in this UNIX directory listing as a protection matrix. (*Note*: *asw* is a member of two groups: *users* and *devel*; *gmw* is a member only of *users*.) Treat each of the two users and two groups as a domain, so that the matrix has four rows (one per domain) and four columns (one per file).
– rw– r– – r– – 2 gmw users 908 May 26 16:45 PPP– Notes
– rwx r– x r– x 1 asw dev el 432 May 13 12:35 prog1
– rw– rw– – – – 1 asw users 50094 May 30 17:51 project.t
– rw– r– – – – – 1 asw dev el 13124 May 31 14:30 splash.gif

- Note sure of this is what you're looking for

Domain	PPP-Notes	prog1	project.t	splash.gif
gmw	r	-	-	-
asw	rw	rwx	rw	rw
users	r	rx	r	-
devel	-	rx	-	r

3. (p. 699) question#21

Internet cafes are businesses where tourists away from home can rent a computer for an hour or two to do business that needs a computer. Describe a way to produce signed documents from one using a smart card (assume that all the computers are equipped with smart-card readers). Is your scheme secure?

Scheme:

1. We can have the individual insert their smart card into the cafe smart-card reader.
2. Then a secure application/software will prompt the user (on the computer) to authenticate using a pin number (usually on the smart card).

3. After the user is authenticated the user writes or loads a document and chooses “sign”.
4. Then the smart card computes a digital signature using that users private key and returns the signature to the application.
5. Finally the document and its signature are saved or sent.

In review, yes this scheme is secure if the private key never leaves the smart card, signing of the document is done on the card and the system uses trusted software.

4. (p. 699) question#23

After getting your degree, you apply for a job as director of a large university computer center that has just put its ancient mainframe system out to pasture and switched over
to a large LAN server running UNIX. You get the job. Fifteen minutes after you start work, your assistant bursts into your office screaming: “Some students have discover-ed the algorithm we use for encrypting passwords and posted it on the Internet.” What should you do?

Some initial steps to take to resolve this issue are *assuming we have some basic knowledge about computers and algorithms

1. Verify if the concern is that the password file was accessed or a weak algorithm was used
2. If the file was accessed we can immediately change all passwords and notify users
3. Audit logs and block the located breach path
4. Finally we can upgrade password hashing by using stringer algorithms such as bcrypt which is a very popular library today for password hashing and verify that the salted hash passwords

5. (p. 701) question#41

in Sec. 9.7.3 we described login spoofing as an attack in which the attacker starts a program on a computer that displays a fake login screen on a computer. This would typically be used in a room full of computers at a university that students could use for assignments. When the student sits down and enters a login name and password, the fake program sends them off to its owner and then exits. The second time the student tries to login in it works and the students thinks there

must of been a typing error the first time. Devise a way in which the operating system could defeat this kind of spoof-ing attack.

We could only allow trusted system components to display login prompts. Additionally for another layer of security we could Implement a secure attention key that when pressed the OS switches to secure mode and launches the real login screen so no other program can mimic it and potentially steal information from users or individuals.