Q1. [8 Marks]

a. Consider the following SQL query intended to verify user login credentials:

SELECT *
FROM Users
WHERE username = '$username' AND pin = '$pin';

i. Explain how an attacker could supply crafted input values for $username and $pin to perform a successful SQL injection attack that bypasses the authentication check. Describe the specific input the attacker might use and explain why this causes the query to succeed without a valid password**. (2 Marks)**

Solution:-

SQL Injection Attack – Steps for a Successful Exploit: An attacker can exploit the given query by inserting malicious SQL in the $eid or $password input. For example, they could set the employee ID input to **' OR 1=1--** (and supply any password). This crafts the SQL query as:

SELECT *
FROM employee
WHERE eid='' OR 1=1--' AND password='...';

Here, the condition eid='' OR 1=1 is always true, and the -- marks the rest as a comment.

Step 1: The attacker finds that user inputs are directly concatenated into the SQL without validation .

Step 2: They input a specially crafted string like ' OR 1=1-- to manipulate the WHERE clause.

Step 3: The database interprets the query such that the authentication check is bypassed (since 1=1 is always true). As a result, the attacker logs in without valid credentials, possibly gaining unauthorized access to all employee records.

ii. The organization stores employee data in an encrypted database and needs to support range queries such as: **SELECT * FROM Employee WHERE eid < 300;**
When using indexes on encrypted data, the order of values can sometimes be revealed. Explain what order leakage is, and describe how index randomization helps hide this order information while still allowing range queries to work. **(2 Marks)**

Solution:-

Order leakage occurs when the order of encrypted index values (e.g., I(eid)=1 < I(eid)=5) reveals the relative order of employee IDs, allowing the server to infer data patterns or ranges. **Index randomization** prevents this by assigning random index values to encrypted data so the true order is hidden, yet the client can still perform range queries like eid < 300 correctly.

b. Employees at TechSolve Ltd. notice their systems slowing down and strange network activity. Later, company files become inaccessible due to encryption. Investigators find that the infection began when one employee installed a free screen-recording app. The malware spread to other computers automatically, hid itself from antivirus tools, and allowed remote attackers to use the systems for sending spam.

i. Identify and explain three different types of malware shown in this scenario. **(2 Marks)**

Solution:-

- Trojan Horse: The fake screen-recording app disguises malicious code to trick the user into installation.
- Worm: The malware spreads automatically to other computers on the network without user action.
- Ransomware: The malware encrypts company files and allows attackers to use infected systems for spam (botnet behavior).

ii. In the TechSolve Ltd. attack, the malware hid from antivirus tools. What is a rootkit, and which two modes of operation could it use to stay hidden? **(2 Marks)**

Solution:-

A rootkit hides malicious activity from users and security tools. In this case, it could use:

User-mode: Hides files or processes by intercepting API calls.

Kernel-mode: Conceals malware by modifying kernel-level process or file listings.

Q2. [7 Marks]

a. A hospital database contains tables for patients, doctors, nurses, and billing staff. Access control must ensure that doctors can view and update only their assigned patients' data, while billing staff handle financial records only.

   i. Explain how Capability lists and Access Control Lists (ACLs) can be implemented in this system to define and enforce permissions for doctors, nurses, and billing staff. **(2 Marks)**

   Solution:-

   - **Capability Lists:** Each user holds tokens defining which objects they can access and operations allowed (e.g., doctors → read/update assigned patients; billing staff → billing records).
   - **ACLs:** Each object lists users or roles with permissions (e.g., patient file → doctor: read/write, nurse: read, billing: none).

   ii. In Attribute-Based Access Control (ABAC)**,** identify and briefly describe the three categories of attributes that are used to make access control decisions. **(2 Marks)**

   Solution:-
   In **Attribute-Based Access Control (ABAC)**, access decisions are based on three sets of attributes:
   Subject Attributes – characteristics of the user or process (e.g., role, department, clearance level).
   Object Attributes – properties of the resource being accessed (e.g., file type, classification).
   Environment Attributes – contextual factors (e.g., time, location, system status).
   These attributes together determine whether access is granted or denied.

b. A company uses a remote challenge–response authentication system where passwords are stored as SHA-256 (salt || password)**.**

   i. Outline the key steps of the remote challenge–response authentication process between a user and a server. **(2 Marks)**

   Solution:- The process works as follows:
   User Identification: The user sends an identity (e.g., username) to the server.
   Challenge: The server generates a random number (*nonce r*) and sends it to the user.
   Response Generation: The user computes a response using a one-way function, typically *f(r, h(P))*, where *P* is the password and *h()* is a secure hash function.
   Verification: The server uses its stored *h(P)* to compute *f(r, h(P))* and compares it with the received response. If they match, authentication succeeds.
   This approach confirms identity without transmitting the actual password and resists replay attacks, since the challenge changes for every session.

   ii. Explain how adding salt to stored password hashes helps protect against rainbow table and precomputed hash attacks. **(1 Marks)**

   Solution:-
   A salt is a random value added to a password before hashing (e.g., SHA-256(salt || password)). It ensures that even identical passwords produce different hashes, preventing attackers from using rainbow tables or precomputed hash lists, since each salted hash must be computed separately