Roll No. _____ Name _____ Group _____

PIS105: Secure Coding 01 March 2025 Time: 2 hours MM:30
Faculty: Dr. Lokendra Vishwakarma

Thapar Institute of Engineering & Technology
Computer Science & Engineering Department
MID SEMESTER EXAMINATION

Instructions:
1. Attempt all questions.
2. Attempt all the subparts of a question at one place.
3. Assume data if required.

---

1. a) Define what is Honeypots? Explain the types of honeypots based on the 1. interaction level and 2. complexities of honeypots. [**1+2=3 marks** | CO1 | BL1]
   b) What are countermeasures for format string vulnerability attack? Explain each one of these with suitable example. [**3 marks** | CO3 | BL2]

---

2. a) How does SSDLC integrate security into each phase of software development? [**4 marks** | CO5 | BL2]
   b) A company implements the "Principle of Least Privilege" to restrict user access, but a misconfiguration allows an employee to access sensitive financial data without proper authorization. Given that the likelihood of this incident occurring in a year is 8%, and the potential financial impact is $100,000, determine the company's expected financial risk per year based on the probability of occurrence and the estimated loss. [**2 marks** | CO2 | BL3]

---

3. a) An attacker is attempting to decrypt a message encrypted using the Affine Cipher. Before launching a brute-force attack, the attacker must first determine the possible key values based on the Affine Cipher's keyspace constraints—where the multiplicative key must be coprime with 26 and the additive key can be any value from 0 to 25. If testing each key combination takes 10 seconds, calculate the total time required to exhaust all possible keys, considering both the multiplicative key constraints and the full keyspace for the additive key. [**4 marks** | CO2 | BL4]
   b) In threat modeling, when we adopt the Attack Tree modeling? Draw an Attack Tree for Gaining Unauthorized Admin Access in an online banking system. [**1+1=2 marks** | CO5 | BL3]

---

4. a) When would you use STRIDE over DREAD in a security assessment? [**2 marks** | CO5 | BL2]
   b) An organization has identified a SQL Injection vulnerability in its web application and seeks to assess its potential risk impact using a structured threat modeling approach. The security team has evaluated the vulnerability across several key aspects: the extent of damage it could cause (8), how easily the attack can be replicated once discovered (9), the level of effort required to exploit the weakness (10), the number of users who could be affected (9), and the likelihood of attackers discovering this flaw (7).
      1. Identify which DREAD factor corresponds to each of the above vulnerability aspects.
      2. Using these values, compute the total risk score based on the DREAD model.
      3. Classify the risk level based on the computed score. [**2+1+1=4 marks** | CO5 | BL4]

---

5. a) Explain the best practices that are adopted to mitigate the Access Control Problems. [**2 marks** | CO2 | BL2]
   b) The following C program simulates a banking system where two threads try to withdraw

money simultaneously. Answer the following: [**1+1+2=4 marks** | CO1 | BL3]

1. Explain how a race condition occurs in this program.
2. What could happen if two users withdraw money at the same time?
3. Modify the program to prevent race conditions using a mutex.

```
#include <stdio.h>
#include <pthread.h>
int balance = 99;
void *withdraw(void *arg) {
int amount = *(int *)arg;
printf("Checking balance: %d\n", balance);
if (balance >= amount) {
    printf("Withdrawing: %d\n", amount);
    balance = balance - amount;
    printf("New balance: %d\n", balance);
    } else {
    printf("Insufficient funds!\n");
        }
    return NULL;
}
int main() {
    pthread_t t1, t2;
    int amount = 50;
    pthread_create(&t1, NULL, withdraw, &amount);
    pthread_create(&t2, NULL, withdraw, &amount);

    pthread_join(t1, NULL);
    pthread_join(t2, NULL);

    printf("Final balance: %d\n", balance);
    return 0;
    }
```

Answers

1. a) **Answer:** [**1+2=3 marks** | CO1 | BL1]

Honeypots: A honeypot is a cybersecurity mechanism designed to detect, deflect, or study unauthorized access attempts by simulating vulnerable systems or networks. By acting as decoy targets, honeypots attract attackers, allowing security professionals to observe their methods and gather intelligence without risking actual assets. **Types of Interaction Level Honeypots:** Honeypots can be categorized based on their interaction level and purpose:
1. Low-Interaction Honeypots: Simulate specific services or vulnerabilities with limited interaction.
2. High-Interaction Honeypots: Emulate entire operating systems or networks, offering deeper engagement with attackers.

b) **Answer:** [**3 marks** | CO3 | BL2]

How to Prevent Format String Attacks 1. Always Specify a Format String

```
printf(userInput); // Direct usage of user input (vulnerable)
printf("%s", userInput); // Explicitly specify the format
```

2. Validate User Input. Example:

```
if (strchr(input, '%') != NULL)
```

3. Use Secure Libraries: For instance,
`snprintf()`
is a safer alternative to
`sprintf()`.

---

2. a) **Answer:** [**1+1+2=4 marks** | CO5 | BL2]

SSDLC is an enhancement of SDLC and embeds security principles and practices at each phase.
SSDLC augments the phases traditional SDLC in the following ways:

- **Requirement Analysis:** At the beginning of software development, security requirements must be defined alongside functional requirements. Key actions:
  - Identify security goals (e.g., confidentiality, integrity, availability).
  - Define access control and authentication mechanisms.
  - Conduct a risk assessment to classify potential threats.
  - Ensure regulatory compliance (GDPR, HIPAA, PCI-DSS).
- **Design:** In addition to creating architectural blueprints and user interfaces, a detailed security architecture is also designed. Key actions:
  - Threat modeling (Identifying attack vectors and mitigation strategies).
  - Minimizing attack surface (Avoid exposing unnecessary services or APIs).
  - Least Privilege (Restrict user permissions).
  - Fail Securely (defaults to a secure state)
- **Development:** During development, writing secure code is crucial to preventing vulnerabilities. This minimizes common vulnerabilities such as injection attacks, etc. Key actions:
  - Input validation (Prevent SQL Injection, XSS, and command injection).
  - Secure authentication (password hashing instead of storing plaintext).
  - Proper error handling (Avoid exposing sensitive system information).
  - Using secure libraries and frameworks (dependencies up to date).
- **Testing and Deployment:** Security testing ensures that vulnerabilities are identified and fixed before deployment. Key actions:
  - Static Application Security Testing (Static Application Security Testing.)
  - Dynamic Application Security Testing (Tests running applications for vulnerabilities e.g., OWASP ZAP, Burp Suite).
  - Penetration Testing (Simulates real-world attacks to identify weaknesses).

- Fuzz Testing (Provides random input to detect unexpected crashes).
- Security is not just about coding; the deployment environment must also be secure. Harden server configurations, Use HTTPS and TLS encryption and Implement Web Application Firewalls.

- **Maintenance:** Security is an ongoing process, and systems must be continuously monitored. Compared to traditional SDLC, the SSDLC takes this a step further by continuously monitoring the security aspects of the software. Any new vulnerabilities discovered post-launch are quickly addressed, and security updates are rolled out as needed. E.g., Patch vulnerabilities, Monitor logs for suspicious activity, and Conduct regular security audits.

b) **Answer:** [**1+1=2 marks** | CO2 | BL3]
To calculate the expected financial risk per year, you use the formula for Expected Loss (Risk Exposure):

$$\text{Expected Risk} = \text{Probability of Occurrence} \times \text{Potential Loss}$$

Given Data:
Probability of Occurrence = 8% = 0.08
Potential Loss = $100,000
Expected Risk = 0.08 x 100,000 = 8,000
The company's expected financial risk per year is $8,000.

---

3. a) **Answer:** [**3+1= 4 marks** | CO2 | BL4]
Step 1: Count Total valid keys:
Valid values for a (multiplicative key): We need all integers from 1 to 25 that are coprime with 26. The valid values of a are: {1,3,5,7,9,11,15,17,19,21,23,25}, i.e., Total valid a values = 12.
Valid values for b (additive) : Any value from 0 to 25 = 26 values.
Thus, Total key combinations: 12×26=312.
Step 2: Multiply by time per attempt
Total time=312×10 = 3,120 seconds
In minutes: 3,120 / 60 = 52 min.

b) **Answer:** [**1+1=2 marks** | CO5 | BL3]
Attack Tree: An Attack Tree is a visual representation of all possible attack paths an attacker can take to compromise a system or achieve a malicious goal. It helps in analyzing security threats, identifying vulnerabilities, and prioritizing defenses.

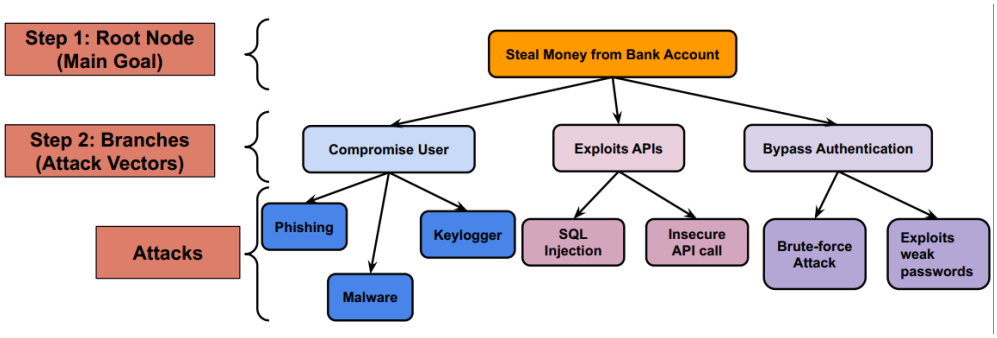

Figure 1: Attack Tree to Gain Unauthorized Access in an Online Banking System

---

4. a) **Answer:** [**0.5 x 4= 2 marks** | CO5 | BL2]

i. STRIDE threat modeling is helpful because it can tell us 'what can go wrong' on the application, system, IT landscape, or business process that we're (threat) modeling.
ii. Compare that to DREAD threat modeling, where the focus is on quantifying threats and less emphasis on identifying threats.
iii. STRIDE and DREAD threat modeling can work well together because STRIDE can identify threats, and DREAD can quantify the threats identified.

iv. Quantifying threats with DREAD is helpful because this information can help with the prioritization of threats

b) **Answer:** [**2+1+1=4 marks** | CO5 | BL4]

i. Given Aspects and Matching DREAD Factors:
- The extent of damage it could cause (8): D (Damage Potential).
- How easily the attack can be replicated once discovered (9): R (Reproducibility).
- The level of effort required to exploit the weakness (10): E (Exploitability).
- The number of users who could be affected (9): A (Affected Users).
- The likelihood of attackers discovering this flaw (7): D (Discoverability).

ii. Compute the Total DREAD Risk Score: The DREAD risk score is the average of the five factors:

$$\begin{aligned}\text{DREAD Score} &= \frac{D+R+E+A+D}{5} \\ &= \frac{8+9+10+9+7}{5} \\ &= 8.6\end{aligned}$$

iii. Classify the Risk Level: Risk levels based on average DREAD scores (commonly used scale): 0–3 as Low, 4–6 as Medium, and 7–10 as High. The computed DREAD score is 8.6, which lies in the high-risk category.

---

5. a) **Answer:** [**0.5 x 4 = 2 marks** | CO2 | BL2]

Best Practices to Mitigate Access Control Vulnerabilities:

i. **Enforce Access Controls Server-Side:** Do not rely on client-side checks. Always validate access permissions on the server.

ii. **Role-Based Access Control (RBAC):** Assign roles (e.g., admin, user) and enforce restrictions based on roles.

iii. **Use Session-Based Validation:** Verify user ownership or permissions for every resource request using session data.

iv. **Audit and Monitor Access:** Log all access attempts and regularly audit logs for suspicious behavior.

v. **Apply the Principle of Least Privilege:** Restrict user permissions to the minimum necessary for their tasks.

b) [**1+1+2=4 marks** | CO1 | BL3]

1. Explain how a race condition occurs in this program.

Answer: A race condition occurs when multiple threads access and modify a shared resource (in this case, the balance variable) concurrently without proper synchronization. In this program:

i. Two threads (t1 and t2) are created to withdraw money.

ii. Both threads check the balance almost simultaneously before withdrawing.

iii. If the balance is 99, both threads may see this value before any withdrawal occurs.

iv. Both threads assume that sufficient funds are available and proceed to withdraw.

v. This results in an incorrect final balance, as both withdrawals reduce the balance without considering the other.

2. What could happen if two users withdraw money at the same time?

i. Suppose the initial balance = 99, and both threads attempt to withdraw 50 at the same time.

ii. Both check if (balance $\geq$ amount), which evaluates to true for both threads.

iii. Each thread subtracts 50, assuming sufficient funds exist.

iv. The final balance might be incorrectly set to -1 (instead of 49), causing inconsistent or incorrect behavior.

3. Modify the program to prevent race conditions using a mutex.

A mutex (mutual exclusion) ensures that only one thread accesses the critical section (balance modification) at a time.

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

// Global variables
pthread_mutex_t lock; // Mutex lock
int balance = 100;    // Initial balance

void *withdraw(void *arg) {
int amount = *(int *)arg;

pthread_mutex_lock(&lock); // Lock the critical section
if (balance >= amount) {
    balance -= amount;
    printf("Withdrawal successful. Remaining balance: %d\n", balance);
    } else {
    printf("Insufficient funds. Balance: %d\n", balance);
    }
    pthread_mutex_unlock(&lock); // Unlock the critical section

    return NULL;
    }

    int main() {
    pthread_t t1, t2; // Thread variables
    int amount = 50; // Amount to withdraw

    // Initialize the mutex lock
    if (pthread_mutex_init(&lock, NULL) != 0) {
    printf("Mutex initialization failed\n");
    return 1;
    }

    // Create threads for withdrawing money
    pthread_create(&t1, NULL, withdraw, &amount);
    pthread_create(&t2, NULL, withdraw, &amount);

    // Wait for threads to complete
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);

    // Destroy the mutex lock
    pthread_mutex_destroy(&lock);

    printf("Final balance: %d\n", balance);
    return 0;
    }
```
