Class Test 1 NetSec

Describe 3 main concerns with the use of passwords for authentication. Explain what is meant by social engineering attacks on password

3 Main Concerns with the Use of Passwords for Authentication

1️⃣ Weak or Guessable Passwords

Many users choose passwords that are:

- Short or simple (e.g., `123456`, `password`)
- Based on personal information (DOB, name, phone number)
- Reused across multiple websites

🔴 **Problem:**
Attackers can break them using:

- **Brute-force attacks**
- **Dictionary attacks**
- **Credential stuffing** (using leaked passwords from other sites)

👉 Since passwords depend on human memory, users often prioritize convenience over security.

2️⃣ Password Theft and Storage Risks

Passwords can be stolen in multiple ways:

- Data breaches from compromised servers
- Malware (keyloggers)
- Network sniffing (if not encrypted)
- Phishing websites

Even if systems store passwords, poor practices like storing plaintext passwords or weak hashes can lead to compromise.

🔴 **Problem:**
If an attacker obtains the password, they can fully impersonate the user.

3️⃣ Replay and Sharing Problems

Passwords are **static secrets** — once known, they remain valid until changed.

- If intercepted once, they can be reused.
- Users sometimes share passwords.
- No built-in mechanism verifies *who* is typing the password.

🔴 **Problem:**
Passwords prove knowledge, not identity.

What is a Social Engineering Attack on Passwords?

A **social engineering attack** is when an attacker manipulates or tricks a person into revealing their password instead of breaking the system technically.

It exploits **human psychology**, not cryptography.

Common Types:

📨 Phishing

Fake emails or websites pretending to be legitimate services asking users to "log in."

Example:

- Fake bank login page
- Fake university portal

📞 Pretexting

Attacker pretends to be:

- IT support
- Bank officer
- Company admin

They request password “for verification.”

🎯 Shoulder Surfing

Physically observing someone entering their password.

💬 Impersonation

Attacker claims urgency:

> "Your account will be locked unless you confirm your password!"

In Simple Terms

Technical attacks break the **system**.
Social engineering attacks break the **person**.

Since passwords rely heavily on human behavior, they are vulnerable not just to cryptographic attacks but also to psychological manipulation.

If you want, I can also explain:

- How multi-factor authentication solves these issues
- Why passwords are being replaced with passkeys
- Exam-ready 6-mark answer version 🙂

An ideal password authentication scheme has to withstand a number of attacks. describe 5 of these

An ideal **password authentication scheme** must resist the following major attacks:

1️⃣ Offline Dictionary Attack

🔹 **What it is:**
An attacker obtains the password database (e.g., hashed passwords) and tries guessing passwords offline without interacting with the server.

🔹 **Why dangerous:**
The attacker can try millions or billions of guesses per second.

🔹 **Defense required:**

- Strong salted hashes
- Slow hashing functions (bcrypt, Argon2, PBKDF2)
- Unique salt per user

2️⃣ Online Guessing / Brute Force Attack

🔹 **What it is:**
The attacker repeatedly attempts login attempts directly on the authentication server.

🔹 **Why dangerous:**
If unlimited attempts are allowed, weak passwords will eventually be guessed.

🔹 **Defense required:**

- Account lockout after failed attempts

- Rate limiting
- CAPTCHA
- MFA (Multi-Factor Authentication)

3️⃣ Replay Attack

🔹 **What it is:**
An attacker captures a valid login message and reuses (replays) it later to gain access.

🔹 **Why dangerous:**
If authentication uses static credentials sent over the network, replay is possible.

🔹 **Defense required:**

- Nonces (random challenge values)
- Timestamps
- Challenge-response protocols
- Session tokens with expiration

4️⃣ Man-in-the-Middle (MITM) Attack

🔹 **What it is:**
An attacker intercepts communication between client and server and alters or reads authentication data.

🔹 **Why dangerous:**
Passwords can be stolen or session hijacked.

🔹 **Defense required:**

- End-to-end encryption (TLS/SSL)
- Certificate validation
- Mutual authentication

5️⃣ Stolen Verifier / Server Database Attack

🔹 **What it is:**
Instead of stealing plaintext passwords, the attacker steals the password **verifier** (e.g., hash) stored on the server.

🔹 **Why dangerous:**
If hashes are weak or unsalted, they can be cracked offline.

🔹 **Defense required:**

- Salted password hashing
- Computationally expensive hash functions

- Zero-knowledge protocols (e.g., SRP)
- Avoid storing plaintext passwords

Summary

An ideal password authentication scheme must resist:

1. Offline dictionary attacks
2. Online guessing attacks
3. Replay attacks
4. Man-in-the-Middle attacks
5. Stolen verifier/database compromise

In short, it must protect passwords:

- In storage
- During transmission
- During verification
- Even if part of the system is compromised

If you want, I can also give you a compact **exam-ready 6 or 10 mark answer format**.

> Explain how access control Lists are used to represent access control matrices. Describe the environments in which they are widely used and their advantages and disadvantages

1️⃣ Access Control Matrix (ACM) – Basic Idea

An **Access Control Matrix** is a conceptual model:

- **Rows** → **Subjects** (users/processes)
- **Columns** → **Objects** (files/resources)
- **Cell (S,O)** → **Permissions** (read, write, execute, etc.)

Example:

	File1	File2
UserA	R,W	R
UserB	R	—

⚠️ The full matrix is usually very large and mostly empty (sparse).
So in practice, we do **not store the full matrix**.

2️⃣ How Access Control Lists (ACLs) Represent the Matrix

An **Access Control List (ACL)** is a **column-wise representation** of the access control matrix.

Instead of storing the whole matrix:

👉 For each **object**, we store a list of:

- Subjects
- Their corresponding permissions on that object

So each **column of the matrix** becomes an ACL.

Example

For **`File1`**, ACL might be:

```
mathematica

File1 ACL:
UserA → Read, Write
UserB → Read
```

For **`File2`**, ACL:

```
yaml

File2 ACL:
UserA → Read
```

Thus:

- Matrix column = ACL of that object
- Empty cells are simply not stored → saves space

3️⃣ Environments Where ACLs Are Widely Used

ACLs are widely used in **object-centered security systems**, especially:

🖥️ Operating Systems

- File systems
- Directory permissions

Examples:

- Microsoft Windows NTFS

- UNIX (extended ACLs)
- Linux file permissions

🌐 Network Security

- Routers and firewalls
- Packet filtering rules

Example:

- “Allow IP A to access Port 80”
- “Deny IP B”

🗄️ Databases

- Table-level or row-level permissions
- User-specific access rights

4️⃣ Advantages of ACLs

✅ 1. Easy to Know Who Can Access an Object

For a given file/resource, you can quickly see:

> “Who has access to this object?”

Useful for:

- Auditing
- Resource sharing control

✅ 2. Efficient for Object Protection

Since protection is usually defined **per object**, ACLs are natural for file systems.

✅ 3. Flexible

Supports:

- Fine-grained permissions
- Different rights per user

✅ 4. Space Efficient

Because most matrix cells are empty, ACL only stores non-empty entries.

5️⃣ Disadvantages of ACLs

❌ 1. Hard to Answer Subject-Based Questions

If you ask:

> "What all files can UserA access?"

You must search through *all ACLs* of all objects.

This is inefficient.

❌ 2. Revocation Complexity

Removing a user completely from the system requires:

- Editing every object's ACL.

❌ 3. Scalability Issues

In systems with:

- Millions of users
- Millions of objects

ACL management becomes complex.

❌ 4. Not Ideal for Capability-Based Systems

If security model focuses on **what subjects can access**, capability lists are better.

6️⃣ Summary

An ACL is:

- A **column-wise implementation** of the Access Control Matrix.
- Each object stores a list of users and their permissions.
- Widely used in:
  - Operating systems
  - Networks
  - Databases

Advantages:

- Object-centered control
- Easy auditing of object access
- Space efficient

Disadvantages:

- Hard to determine user-wide permissions
- Revocation complexity
- Scaling issues

If needed, I can also explain:

- Capability Lists (row-wise representation)
- Comparison between ACL and Capability Lists for exams
- Diagram-based explanation for 8-mark answer 😊