Trieu Huynh Ba Nguyen 000405980

1. By storing an access list with each object, it becomes possible to control access privileges and limit or extend them in a localized manner. This is a significant advantage of this approach. However, there is a downside to associating access lists with objects, which is the overhead associated with checking whether the requesting domain is on the access list. This check is an expensive operation that must be performed each time the object is accessed. In addition, as the number of objects and access lists grows, the overhead can become a significant performance bottleneck. It's important to strike a balance between security and performance when implementing access control mechanisms.

2. Domains with associated capabilities offer a high degree of flexibility and faster access to objects. This approach allows the system to efficiently authenticate capabilities presented by a domain. Capabilities can also be easily passed from one domain to another, enhancing the system's flexibility. However, this approach also has its drawbacks. The flexibility associated with capabilities can result in a lack of control over the flow of capabilities and the ability to revoke them. Managing the revocation of capabilities and controlling their distribution can be a challenging task, making it difficult to maintain a secure and controlled system. As with access lists, it's important to find a balance between flexibility and control when using capability-based systems.

3. Role-based access control and access-matrix facilities share similarities in that domains and roles are granted access to resources. However, Role-based access control differs from access-matrix facilities in terms of flexibility and the type of access privileges granted to entities. Some access-control facilities permit modules to perform a switch operation that allows them to assume the privileges of another module, and this switch can be performed in a transparent manner. However, in role-based access-controlled systems, role switching is less transparent and requires the explicit use of passwords instead of being granted through the access-control system's mechanism. This approach provides more control over role switching but also reduces the system's flexibility. Role-based access-controlled systems grant roles with specific access privileges, whereas access-matrix facilities grant access at the domain level. This approach allows for more fine-grained control over access privileges, which is particularly useful in large and complex systems where security is a top concern.

4. When a user creates a password, the system generates a random number, known as a salt. This salt is then appended to the user-provided password, and the resulting string is encrypted and stored along with the salt in the password file. When a password check is needed, the password presented by the user is concatenated with the corresponding salt, encrypted, and checked for equality with the stored password. The use of a unique salt for each user makes it difficult for password crackers to check a single candidate password, encrypt it, and compare it against all of the encrypted passwords stored in the system. This approach significantly increases the difficulty of cracking passwords, as an attacker would need to perform the encryption process for each individual password, greatly slowing down the attack. It's important to use strong and unique salts to further increase the security of the password storage mechanism. Additionally, it's crucial to follow other best practices, such as enforcing password complexity requirements and regularly prompting users to change their passwords.

5. In symmetric encryption, the same key is used for both encrypting and decrypting messages. On the other hand, asymmetric encryption requires the use of two different keys: one for encryption and another for decryption. While symmetric schemes are generally faster than asymmetric ones, they don't provide theoretical guarantees on the intractability of reverse-engineering the encryption. Asymmetric schemes, on the other hand, are based on mathematical foundations that offer such guarantees. Additionally, asymmetric schemes have other advantages over symmetric schemes, including their ability to support authentication, confidentiality, and key distribution. Although asymmetric schemes are generally more expensive than symmetric ones, their added benefits make them well-suited for many applications where security is a top priority. By contrast, symmetric encryption is typically used in applications where speed is of the essence, and where theoretical guarantees are not a primary concern. It's important to choose the appropriate encryption scheme based on the specific needs of the application and the level of security required.