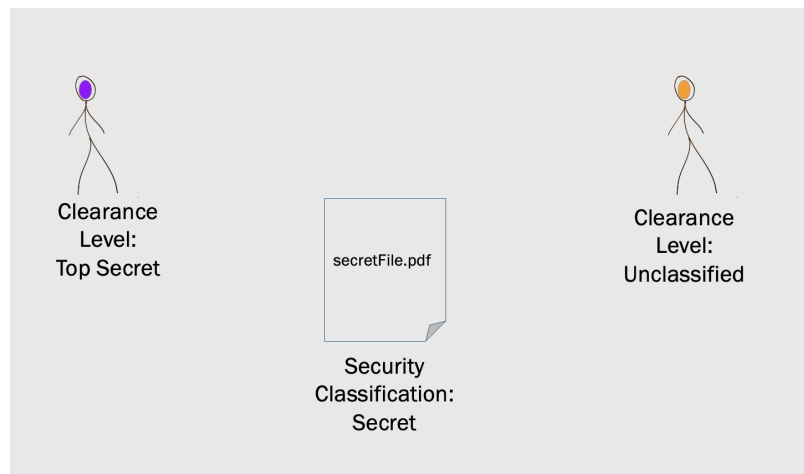




Question 1: Mandatory Access Control Models 1.1. In the above figure, Purple has a Top Secret clearance level, Orange has an Unclassified clearance level, and the file has a security classification of Secret. Under the Biba mandatory access control model, who is allowed to read the secret file?

1.2. Using the same figure, under the Bell-LaPadula mandatory access control model, who is allowed to write the secret file?

Question 2: More Mandatory Access Control Mandatory Access control models make use of which security principles?

- ☐ Defense in depth and type checking
- ☐ Separate data and control channels and fail-safe defaults
- ☐ Principle of least privilege and complete mediation
- ☐ Separation of privilege and open design

Question 3: Confused Deputy Attack The code in the following listing runs with root privileges (the executable has the setuid bit set) and is vulnerable to a confused deputy attack.

```
1 FILE *fp1, *fp2;  
2 // open argv[2] for writing  
3 fp1 = fopen(argv[2], "w");  
4  
5 // open fstab file to append  
6 fp2 = fopen("/etc/fstab", "a");  
7  
8 /* update the /etc/fstab file, write any error messages  
9 to the user-provided file, and exit */
```

3.1. Identify the line of code that is vulnerable.

3.2. Explain the vulnerability.

3.3. What is the mechanism for making authorization decisions that is designed to stop confused deputy attacks like this one?

Question 4: Protecting the OS A student suggests letting user programs modify their own base-bounds registers so that they can request more memory without making a system call. Explain how a malicious program could abuse this design. Why must the OS control these registers, and how does a system call allow a program to request a service without giving it unrestricted kernel privileges?

Question 5: Memory Reuse and Isolation The OS prevents two processes from accessing each other's memory while they are running. After one process exits, however, the OS gives its memory to a new process without clearing the old contents. Is process isolation alone sufficient to protect the first process's secrets? Explain the problem and identify the trusted-system feature that addresses it.

Question 6: Discretionary Access Control Below we have an access control matrix for a shared instructional computer, which uses a discretionary access control policy. Alice is an instructor for the course and should have read and write access to the exam and the syllabus. Bob is a TA for the course and should have only read access to the exam, and read and write access to the syllabus. Carol is a student in the course and should have no access to the exam and only read access to syllabus. Everyone can print to the attached printer. The access control matrix shown in the figure enforces this policy. Describe one way in which the spirit of the policy could be violated, despite the correct enforcement of the access control matrix

	exam.txt	printer	syllabus.txt
Instructor Alice	RW	W	RW
TA Bob	R	W	RW
Student Carol	--	W	R

Question 7: Online and Offline Password Guessing A login service allows only five failed login attempts per account per minute. An attacker then steals a copy of its password database, which contains password hashes and the information needed to compute them.

- 7.1. Why does the rate limit slow online guessing but not offline guessing? Explain how the attacker can check an offline guess without reversing the hash function.

- 7.2. How does increasing the number of password-hashing iterations affect the attacker? What cost does this impose on the legitimate login service?

Question 8: What Does a Salt Protect? Two users choose the same password. The service hashes each password with a different, unique salt and stores the salts alongside the hashes.

- 8.1. Should the two stored hashes generally match? Explain why unique salts make a single pre-computed password dictionary less useful across many accounts, even though the salts are public.

- 8.2. A developer claims that salting makes a common password safe from offline guessing. Do you agree? Explain what an attacker can still do after stealing a user's salt and hash.

Question 9: Password Guessing Strategies 9.1. One attacker uses personal information to guess a particular user's password. Another tries a few common passwords against thousands of accounts. Which is a targeted attack and which is a trawling attack? Why might a per-account rate limit still allow the second attacker to test many guesses across the system?

9.2. An attacker obtains working username–password pairs from a breach of one website and tries them on another. Name this attack and explain why even a long, randomly generated password can be vulnerable if it is reused across websites.

Solutions

1. Mandatory Access Control Models

- (a) **Orange only.** Biba prohibits reading down. Treating the levels in the question as integrity levels, Orange may read up from Unclassified to Secret, but Purple may not read down from Top Secret to Secret.
- (b) **Orange only.** Bell–LaPadula prohibits writing down. Orange may write up to Secret, but Purple may not write down to Secret.

2. More Mandatory Access Control

Principle of least privilege and complete mediation (the third choice). Subjects should receive only the access needed for their work, and every access must be checked against the policy. A reference monitor enforces the policy through complete mediation; the policy must also be configured to grant appropriately limited privileges.

3. Confused Deputy Attack

- (a) Line 3: `fp1 = fopen(argv[2], "w");`
- (b) The caller controls the pathname, but the program opens it using its root privileges. An unprivileged caller can supply a protected file's pathname, causing the program to truncate that file and possibly write error messages to it. The program is the deputy: it incorrectly uses its own authority for a caller-directed operation.
- (c) **Capabilities.** Require a capability that grants write access to the caller's chosen output file, and use a separate capability for appending to `/etc/fstab`. A pathname alone must not let the caller exercise the program's privileged authority.

Code note: The declaration should be `FILE *fp1, *fp2;;` the repeated `fp1` is a typo, separate from the security flaw.

4. Protecting the OS

A malicious process could change its permitted address range to include another process's memory or the kernel's memory, allowing it to steal secrets or corrupt OS code and protection data. The registers must be controlled by privileged OS code, with hardware enforcing their limits. A system call transfers control to a designated kernel entry point. The kernel validates the request and its arguments, performs permitted work, and returns to user mode. The caller's own code does not gain unrestricted kernel privileges.

5. Memory Reuse and Isolation

No. Isolation between running processes does not erase data left behind by a previous owner. The new process can read secrets from memory that it is now legitimately allowed to access. **Object sanitization**, such as zeroing memory before exposing it to a different process, prevents this disclosure. Merely freeing the memory is not sufficient.

6. Discretionary Access Control

Bob can read the exam and copy its contents into the syllabus, which he may write. Carol can then read the exam contents from the syllabus. Every individual access is permitted by the matrix, yet the exam's confidentiality is lost. The matrix controls direct accesses but does not prevent this indirect information flow.

7. Online and Offline Password Guessing

- (a) Online guesses go through the login service, which can enforce the rate limit. Offline guesses run on the attacker's own hardware, so the service cannot count or restrict them. The attacker applies the same password-hashing procedure to each candidate, using the stored salt and parameters if applicable, and compares the result with the stolen hash. This tests guesses without inverting the hash function.
- (b) More iterations increase the computational cost of each guess, reducing the attacker's guessing rate for fixed computing resources. The legitimate service must also do more work for each password verification, increasing login latency and server load.

8. What Does a Salt Protect?

- (a) The hashes should generally differ because the salts make the hash inputs different; accidental collisions are possible but unlikely with a suitable hash function. A table computed for one salt cannot generally be reused for another. The attacker must compute candidate hashes separately for each distinct salt. The salts need not be secret to prevent this reuse of work.
- (b) No. The attacker knows the user's salt and can hash common password guesses with it, comparing each result with the stolen hash. Salting limits precomputation and sharing of work across accounts; it does not make an easily guessed password unpredictable.

9. Password Guessing Strategies

- (a) The first is a **targeted attack**; the second is a **trawling attack**. A per-account limit restricts guesses against each account separately. By trying only a few guesses per account, the trawling attacker can stay within each account's limit while making many guesses in total across thousands of accounts.
- (b) This is **credential stuffing**. If the same credentials work on another website, the attacker can submit the already known password directly. Its length and randomness do not help once it has been exposed; using a different password for each website prevents this reuse of a stolen password.