Passwords and Logins

Study Guide

This is a skill-building guide. Its purpose is to train you to produce, test, and revise precise password- and login-related risk claims until they are ready for review by others. You will practice naming concrete credential assets, locating failures in the credential lifecycle, anchoring claims to evidence, and proposing containment-oriented improvements.

What You Must Remember

The Authentication Frame

Authentication answers one question:

Who are you?

Security reasoning must also answer:

- Where did this credential originate?
- Where is it stored or copied?
- How long is it valid?
- What else becomes reachable if it leaks?

If you forget everything else, credentials are assets, not just gates.

How Linux Handles Passwords

Linux separates identity from verification material:

- `/etc/passwd`: public identity registry
- `/etc/shadow`: protected password verification data

This separation exists so:

- identities can be mapped without revealing secrets
- raw passwords never need to be stored

If you forget everything else, the system never needs the real password once it is set.

The Stored Form

Password verification material has this structure:

```
$id$salt$hash
```

- id: hash scheme identifier
- salt: per-password randomness
- hash: result of hashing password + salt

You must be able to recognize this structure, identify common IDs, and explain why the choice of scheme affects the guessing cost.

Salt

A salt is not secret.

Its purpose is to:

- prevent identical passwords from producing identical hashes
- destroy the economics of rainbow tables

Salts do not stop brute force. They ensure each password must be guessed independently.

If you forget everything else: salts protect uniqueness, not secrecy.
Password Resistance Reasoning
Resistance grows with:

(character set size) ^ (length)

Key implications:

- Length increases resistance exponentially
- Complexity rules add far less than extra length
- At sufficient length, brute force becomes irrelevant

Practical baselines:

- 12 characters: minimum if unique and random
- 16+ characters: strong default
- 20–24 characters: ideal for manager-generated passwords

If you forget everything else, length beats complexity.

How Passwords Are Actually Compromised

Ranked by real-world frequency:

1. Reuse
2. Phishing
3. Exposure at rest (scripts, docs, tickets, logs)
4. Endpoint theft (malware, browser stores)
5. Offline guessing (only after hashes are stolen)

If you forget everything else, reuse and exposure dominate cryptography.

The Credential Lifecycle

You must be able to reason about credentials across all stages:

- Creation: length, randomness, uniqueness
- Storage: hashed vs raw, placement, access scope
- Use: interactive login vs embedded use
- Copying: scripts, docs, screenshots, chat
- Rotation: how quickly damage can be contained
- Revocation: disabling access safely

Security fails when any stage is ignored.

The Three Review Tests (Credential-Specific)

A password-related claim is ready for review only if it passes all three tests:

Test	Requirement
Asset Test	Name a specific credential or account, not “the system.”
Lifecycle Stage Test	Identifies the stage where failure occurs
Evidence Test	Anchored to observable evidence (file, repo, config, log, ticket)

Rewrite rule: If a claim fails one test, rewrite only the failing part.

CIA, Used Correctly

CIA is outcome vocabulary, not an explanation.

Category	Meaning
Confidentiality	An unauthorized identity can read information
Integrity	Unauthorized identity can modify behaviour or data
Availability	Required access fails due to lockout or containment

Rule: Apply CIA only after damage is written as an observable outcome.

Core Process to Practice

Use this sequence until it becomes automatic:

1. Name the credential asset: user account, service account, API key, database password.
2. Locate the lifecycle stage: where the failure actually occurs.
3. State the weak spot: present-tense condition that lowers resistance now.
4. Write observable damage: what would be noticed if misuse occurs.
5. Run the review tests: fix only what fails.
6. Classify with CIA: pick the primary harm from the damage.
7. Improve containment/friction: reduce blast radius without breaking legitimate use.

Common Traps

- Treating passwords as strings instead of assets
- Saying “weak password” without naming evidence
- Assuming hashing solves exposure
- Ignoring copying and reuse
- Proposing complexity rules instead of length and uniqueness
- Listing CIA labels without damage

Readiness Questions

Answer without notes. Hesitation shows what to practice next.

Recall

1. Write the $id$salt$hash structure from memory.
2. Explain why /etc/passwd and /etc/shadow are separated.
3. Explain what a salt does in one sentence.

Reasoning

4. Why does reuse multiply damage?
5. Why does hashing not protect leaked passwords from offline guessing?
6. Why does length matter more than complexity?

Application

7. Given a leaked service password, name the asset.
8. Given a password in a script, name the lifecycle stage that failed.
9. Given forced resets after a leak, identify the CIA impact.

Confidence Check

10. Which lifecycle stage do you personally overlook most often, and why?

Practice Questions

Answer in writing. Aim for short, precise statements.

Definitions and Recall

1. Define credential asset using the word “specific.”
2. Define lifecycle stage using the word “where.”
3. Explain why salts do not provide secrecy.
4. Explain why length changes brute-force economics.

Extraction from Scenarios

5. What is a sign that an asset is described too broadly?
6. What is a sign that the wrong lifecycle stage was chosen?
7. What is a sign that the weak spot is written as a possibility instead of a condition?
8. What is a sign that damage is still hypothetical?

Writing Reviewable Claims

9. Write a sentence template for naming a credential asset.
10. Write a sentence template for stating a weak spot in the storage stage.
11. Write a sentence template for observable damage a manager would notice.
12. What minimum information must a credential risk claim contain to be reviewable?

Review Test Practice

13. Give an example of a claim that fails the Asset Test.
14. Give an example of a claim that fails the Lifecycle Stage Test.
15. Give an example of a claim that fails the Evidence Test.
16. If only the Evidence Test fails, what are you allowed to rewrite?
17. Why is rewriting only the failing component important?

CIA Classification

18. A reused admin password allows access to multiple systems. Which CIA category is primary, and why?
19. A leaked password triggers emergency resets, halting automation. Which CIA category is primary?

20. A stolen hash enables unauthorized data changes after cracking. Which CIA category is primary?
21. Why is it misleading to list multiple CIA categories without choosing one?

Friction and Containment

22. Describe one change that reduces blast radius after a credential leak.
23. Describe one change that improves containment without increasing login friction.
24. Why does eliminating shared credentials usually improve security and operations?

Confidence Check

25. Which part of credential reasoning do you find hardest to make concrete, and why?
26. When revising a credential claim, what is the first thing you check to confirm improvement?

If you can consistently name credential assets, locate lifecycle failures, anchor claims to evidence, and propose containment-oriented fixes that pass all three review tests, you understand passwords and logins well enough to secure them.