
WEEK 2 – DAY 9

BROKEN AUTHENTICATION (OWASP TOP 10)

---

1. WHAT IS BROKEN AUTHENTICATION?

Simple Definition

Broken Authentication occurs when authentication mechanisms are **incorrectly designed or implemented**, allowing attackers to **compromise user identities**.

This is **not about one bug** — it is about **system-level weakness** in how identities are managed.

---

2. WHY BROKEN AUTHENTICATION IS SO COMMON

Authentication systems fail because:

- Developers focus on **functionality over abuse scenarios**
- Attackers reuse leaked credentials
- Systems trust tokens/sessions too much
- Rate limiting and monitoring are missing

---

3. CREDENTIAL STUFFING

What Is Credential Stuffing?

Attackers take:

- Email/password lists from previous breaches
- Automatically try them on other websites

This works because **users reuse passwords**.

How Attack Happens

1. Attacker gets breach dump
2. Uses bot or script
3. Tries thousands of logins per minute
4. Successful logins are sold or abused

Why Traditional Login Fails

- Correct credentials look “legitimate”
- No brute-force pattern per account
- Same IP may rotate

Real-World Impact

- Banking fraud
- Account takeovers
- Privacy breaches

---

4. WEAK PASSWORD POLICIES

Examples of Weak Policies

- Minimum 6 characters
- No complexity requirement
- No password rotation
- Allowing common passwords (123456, password)

Why Weak Passwords Are Dangerous

- Easier to brute-force
- Easily cracked if hashes leak
- Combine with credential stuffing

Secure Password Policy (Interview-Ready)

- Minimum 12 characters
- Block common passwords
- Rate-limit attempts
- Hash with bcrypt / Argon2
- Optional MFA

---

5. SESSION FIXATION

What Is Session Fixation?

An attacker forces a user to use a **known session ID**, then hijacks the session after login.

How Attack Works

1. Attacker creates a session
2. Sends victim a link with that session ID
3. Victim logs in
4. Session becomes authenticated
5. Attacker reuses the same session ID

Why This Happens

- Session ID is **not regenerated after login**

Prevention

- Always regenerate session ID on login
- Use secure cookies
- Set HttpOnly and Secure flags

---

6. HANDS-ON: SIMULATE BRUTE FORCE ATTACK

Simple Attack Simulation (Concept)

```
passwords = ["123456", "password", "admin", "qwerty"]


for pwd in passwords:
   response = login("admin", pwd)
   if response.success:
      print("Password found:", pwd)
```

This demonstrates:

- No rate limit
- Unlimited attempts
- Predictable behavior

---

7. ADDING RATE LIMITING (DEFENSE)

Why Rate Limiting Works

- Slows attackers
- Makes brute force expensive
- Creates detectable patterns

Example: Flask Rate Limiting

```
from flask_limiter import Limiter
from flask_limiter.util import get_remote_address

limiter = Limiter(
    get_remote_address,
    app=app,
    default_limits=["5 per minute"]
)

@app.route("/login", methods=["POST"])
@limiter.limit("5 per minute")
def login():
    ...
```

Additional Protections

- CAPTCHA after failures
- Temporary account lock
- IP + user-based limits
- Logging & alerts

---

8. INTERVIEW QUESTIONS & STRONG ANSWERS

Q1: What is Broken Authentication?

“Broken Authentication refers to weaknesses in identity verification mechanisms that allow attackers to compromise user accounts.”

Q2: Difference between brute force and credential stuffing?

“Brute force guesses passwords, while credential stuffing uses real leaked credentials at scale.”

Q3: How do you prevent session fixation?

“By regenerating session IDs after login and using secure cookie flags.”

Q4: Is rate limiting enough?

“No. It must be combined with strong password policies, MFA, and monitoring.”

---

9. SECURITY MINDSET

Attacker Thinking:

- How many attempts can I make?
- Can I reuse leaked credentials?
- Is the session predictable?

Defender Thinking:

- How fast can someone fail?
- What happens after 5 failures?
- Can sessions be hijacked?
