
WEEK 2 – DAY 13

SECURITY MISCONFIGURATION (OWASP TOP 10)

---

1. WHAT IS SECURITY MISCONFIGURATION?

Simple Definition

Security Misconfiguration occurs when systems are **deployed with insecure settings**, default credentials, exposed services, or unnecessary features enabled.

Important:

- No “hack” is needed
- Attackers simply **find what is already open**
- One misconfiguration can **expose the entire system**

---

2. WHY SECURITY MISCONFIGURATION IS SO COMMON

- Developers focus on functionality
- Defaults are left unchanged
- Debug features remain enabled
- Secrets are hard-coded
- Cloud resources are exposed publicly

Most breaches happen due to **human error**, not advanced exploits.

---

3. OPEN PORTS

What Are Open Ports?

Ports are communication endpoints.

Example:

- 80 → HTTP
- 443 → HTTPS
- 22 → SSH
- 3306 → MySQL

Why Open Ports Are Dangerous

- Expose internal services
- Enable brute-force attacks
- Allow direct database access

Attacker View

“What services are listening?”

Tool:

nmap -sS target_ip

Defender Best Practice

- Close unused ports
- Firewall rules
- Bind services to localhost
- Use VPN / private networks

---

4. DEBUG MODE (CRITICAL ISSUE)

Flask Debug Mode Danger

app.run(debug=True)

If exposed:

- Shows stack traces
- Reveals environment variables
- May allow **remote code execution**

Real Impact

- Attacker sees secrets
- Application internals exposed
- Full server compromise possible

Correct Production Setting

app.run(debug=False)

Or use:

export FLASK_ENV=production

---

5. DEFAULT CREDENTIALS

Examples

- admin / admin
- root / root
- test / test

Why This Happens

- Demo environments promoted to production
- Forgotten admin panels
- Third-party tools with defaults

Real Breaches

- Routers
- Databases
- Cloud dashboards

Prevention

- Force password change on setup
- Disable default accounts
- Enforce strong policies

---

6. SECRETS EXPOSURE

What Are Secrets?

- API keys
- Database passwords
- JWT secrets
- Encryption keys

Dangerous Practice

SECRET_KEY = "mysecret123"

DB_PASSWORD = "password"

If leaked:

- Token forging
- Database takeover
- Data breach

Secure Practice (Environment Variables)

import os

SECRET_KEY = os.environ.get("SECRET_KEY")

DB_PASSWORD = os.environ.get("DB_PASSWORD")

Additional Protections

- .env files (not committed)
- Secrets managers
- Key rotation

---

7. HANDS-ON: SECURE FLASK CONFIGURATION

Secure Flask App Example

import os

from flask import Flask

app = Flask(__name__)

app.config["SECRET_KEY"] = os.environ.get("SECRET_KEY")

app.config["SESSION_COOKIE_SECURE"] = True

app.config["SESSION_COOKIE_HTTPONLY"] = True

app.config["SESSION_COOKIE_SAMESITE"] = "Lax"

app.config["DEBUG"] = False

Production Checklist

- Debug disabled
- Secrets externalized
- Secure cookies
- HTTPS enforced
- Logging enabled

---

8. SECURITY HEADERS (IMPORTANT)

Recommended Headers

X-Content-Type-Options: nosniff

X-Frame-Options: DENY

Strict-Transport-Security: max-age=31536000

Content-Security-Policy: default-src 'self'

These:

- Prevent clickjacking
- Enforce HTTPS
- Reduce XSS impact

---

9. INTERVIEW QUESTIONS & STRONG ANSWERS

Q1: What is Security Misconfiguration?

“Security Misconfiguration refers to insecure default settings or improper system configurations that expose applications to attack.”

Q2: Why is debug mode dangerous?

“Because it exposes internal application details and can enable remote code execution.”

Q3: How do you protect secrets?

“By storing them outside source code using environment variables or secret managers.”

Q4: Are misconfigurations common in cloud?

“Yes. Publicly exposed storage and open security groups are frequent causes of breaches.”

---

10. ATTACKER VS DEFENDER THINKING

Attacker:

- What ports are open?
- Is debug enabled?
- Are default creds working?
- Are secrets exposed?

Defender:

- Are defaults removed?
- Is production hardened?
- Are secrets protected?
- Are logs monitored?