Full-Stack AppSec & Infrastructure Auditor

Examination Scope: 35 Questions | 40 Minutes | Passing Standard: 80%

1. EXECUTIVE OVERVIEW & CREDENTIAL STANDARDS

Executive-level certification for comprehensive external attack-surface auditing: Subresource Integrity (SRI), cross-origin credential

2. KEY TOPICS & SYLLABUS

* Subresource Integrity (SRI) and CDN compromise defenses
* CORS misconfiguration, Origin reflection, and credential exfiltration
* Sensitive file leaks: /.git/HEAD, /.env, /backup.sql, /composer.json
* Dangerous port identification: Database (3306, 5432, 6379, 27017) & Admin (3389, 23)
* CMS hardening: WordPress XML-RPC abuse, user enumeration, and sensitive API shielding
* Threat intelligence, malicious domain reputation, and abuse reporting

Examination Integrity & Anti-Cheat Protocols:

* Correct answer keys are evaluated strictly server-side and never sent to client devices.
* Questions and multiple-choice options are dynamically shuffled per individual session.
* Time limits are strictly enforced on the server; background blur/tab switches are audited.

CHAPTER 1: Subresource Integrity (SRI) & Supply Chain Security

Preventing compromised CDNs from injecting malicious JavaScript into user sessions.

Modern websites routinely load third-party JavaScript libraries (jQuery, Bootstrap, analytics) from public CDNs. If a CDN is compromised or an attacker modifies the hosted file, every website including that script executes malicious code (as seen in major supply chain attacks).

Subresource Integrity (SRI) enables browsers to verify that resources fetched from CDNs match an exact cryptographic hash (e.g. SHA-384). If even a single byte differs, the browser immediately refuses execution.

Best practice: Always pair integrity hashes with the crossorigin="anonymous" attribute.

```
<script
  src="https://cdnjs.cloudflare.com/ajax/libs/lodash.js/4.17.21/lodash.min.js"
  integrity="sha384-H6MmXflB4yB9J1kM2oB1M4I+gqY6K6v9a4c1D0b2e8="
  crossorigin="anonymous">
</script>
```

CHAPTER 2: CORS Security & Exploitation Pitfalls

Stopping unauthorized cross-domain data reading and credential exfiltration.

Cross-Origin Resource Sharing (CORS) relaxes the Same-Origin Policy so specific trusted domains can read API responses. Common catastrophic mistakes:

1. Dynamically echoing the incoming Origin header into Access-Control-Allow-Origin while setting Access-Control-Allow-Credentials: true. This gives any website on the internet the ability to issue authenticated requests and steal user data.

2. Using naive regex matching (e.g. allowing "example.com" which mistakenly matches "attacker-example.com" or "example.com.attacker.com").

3. Allowing the "null" origin, which can be triggered by sandboxed iframes or local HTML files.

```
# Secure Express.js CORS handler
const ALLOWED_ORIGINS = new Set(['https://app.example.com', 'https://admin.example.com']);
app.use((req, res, next) => {
  const origin = req.headers.origin;
  if (ALLOWED_ORIGINS.has(origin)) {
    res.setHeader('Access-Control-Allow-Origin', origin);
    res.setHeader('Access-Control-Allow-Credentials', 'true');
  }
  next();
});
```

© 2026 SurfaceScan · CFSAA Certification Curriculum

CHAPTER 3: Sensitive Path Leakage & Reconnaissance

Eliminating exposed .git metadata, .env credentials, and database dumps.

During automated reconnaissance, attackers probe hundreds of sensitive endpoints. If these endpoints return HTTP 200 with raw file contents rather than 404 or 403, complete system compromise is imminent:

- /.git/HEAD: Allows dumping the entire repository history, uncovering committed API keys, internal endpoints, and hidden vulnerabilities.
- /.env & /config.json: Immediate exposure of production secrets, database credentials, AWS keys, and payment gateway tokens.
- /wp-json/wp/v2/users: WordPress default REST API leaks internal administrative usernames, enabling targeted brute-force attacks.

```
# Nginx rule: Block all hidden dotfiles and sensitive backups
location ~ /\.(?!well-known) {
  deny all;
  return 404;
}
location ~* \.(env|git|sql|bak|yml|yaml|ini|log|conf)$ {
  deny all;
  return 404;
}
```

CHAPTER 4: Database & Management Port Hardening

Securing database ports (MySQL, PostgreSQL, Redis, Mongo) and admin services.

Exposing database or remote administrative management services directly to 0.0.0.0 is an egregious architectural flaw:

- MySQL (3306), PostgreSQL (5432), MongoDB (27017): Exposed databases are aggressively targeted by ransomware bots that wipe data and demand Bitcoin payments.
- Redis (6379): Often deployed without authentication by default; attackers write SSH authorized_keys files to gain root access.
- RDP (3389) and Telnet (23): High-value administrative interfaces frequently breached via automated credential stuffing.

Remediation: Bind all internal services strictly to 127.0.0.1 or private VPC subnets, and configure host firewalls (UFW / iptables) to drop all unauthorized inbound packets.

```
# Ubuntu UFW Firewall rules
ufw default deny incoming
ufw default allow outgoing
ufw allow 22/tcp
ufw allow 80/tcp
ufw allow 443/tcp
ufw enable
```

© 2026 SurfaceScan · CFSAA Certification Curriculum