JS Contextual API Hacking Checklist

JSMON

Introduction

APIs are the backbone of modern applications, but they often leak sensitive information through JavaScript files, misconfigurations, and weak security measures. This checklist focuses on API hacking techniques, with special attention to JavaScript file analysis.

---

1. Extract API Endpoints from JavaScript Files

- Inspect JavaScript files loaded in the browser (`DevTools > Sources`).
- Search for fetch(), axios(), or `XMLHttpRequest calls.`
- Look for API base URLs and hidden paths.
- Use grep for automated extraction:

```
grep -Eo 'https?://[^"']+' *.js
```

---

2. Identify Hardcoded API Keys & Secrets

- Scan JS files for exposed API keys:

```
grep -E 'API_KEY|token|Bearer' *.js
```

- Look for Firebase, Stripe, AWS, and third-party service keys.
- Test API keys using Postman or curl to check access levels.

---

3. Find Undocumented API Endpoints

- Analyze JavaScript code for endpoints not listed in public docs.
- `Identify /admin, /internal, /private routes.`
- Use API fuzzing tools like ffuf or wfuzz to discover hidden endpoints.

---

4. Analyze GraphQL Operations from JavaScript

- Look for `GraphQL` queries and mutations in JS files.
- Check for `introspection` leaks:

```
{
  __schema {
    types {
      name
      fields { name }
    }
  }
}
```

- Test for authorization bypass using direct GraphQL operations found in JS files.

---

5. Extract WebSockets URLs from JavaScript

- Identify WebSocket connections (`ws://` or wss:// in JS files).
- Test for unauthenticated data leaks by connecting directly.
- Use `wscat` to interact with WebSocket endpoints:

```
wscat -c ws://target.com/socket
```

---

6. Enumerate API Parameters & Manipulate Requests

- Use Burp Suite's Param Miner to find hidden parameters.
- Modify request parameters to check for IDOR (Insecure Direct Object References).
- Test with various payloads: null, true, false, ../, etc.

---

7. Test for Broken Authentication & Authorization

- Check if APIs present in JS file accept unauthenticated requests.
- Try using a guest token or JWT tampering.
- Test different user roles (admin, user, guest) to escalate privileges.

---

8. Exploit Rate-Limiting & Brute-Force Vulnerabilities

- Check if APIs enforce rate limits on login, OTP, and sensitive actions.
- Use Burp Intruder or ffuf to test for brute-force vulnerabilities.
- Modify request headers (X-Forwarded-For) to bypass rate limits.

---

9. Test for Server-Side Request Forgery (SSRF)

- Identify API endpoints fetching external URLs (`url=`, `file=`, `redirect=`).
- Try SSRF payloads:

```
https://target.com/api?url=http://169.254.169.254/latest/meta-data/
```

- Check for responses leaking internal IPs or AWS metadata.

---

10. Check for CORS Misconfigurations

- Analyze the Access-Control-Allow-Origin response headers.
- Test if wildcard (*) allows cross-origin requests.
- Try executing unauthorized API calls from a different domain.

Conclusion

APIs often expose sensitive data due to weak security configurations. Hackers can leverage JavaScript files to extract valuable API information and test for vulnerabilities. Always ensure secure API design and implement strong authentication, rate limiting, and access controls.

Stay Secure & Happy Hacking!