Member1

Name:Mahmoud Said
ID:2305514

Member2

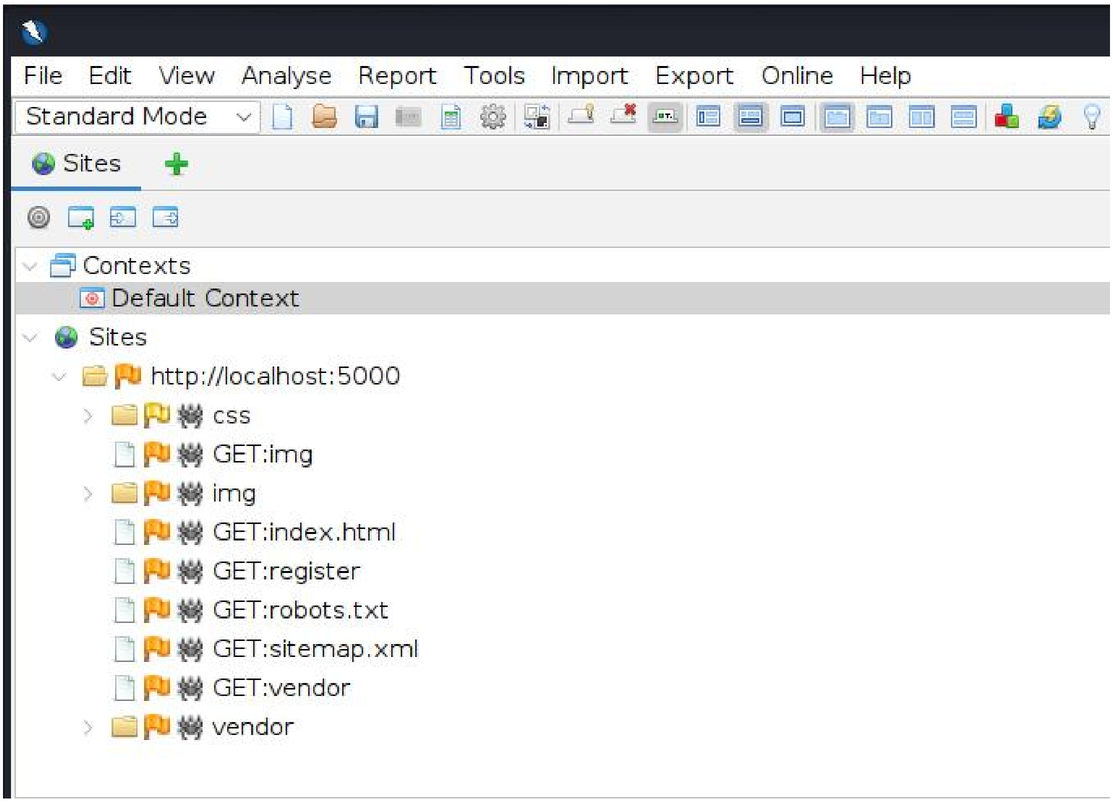
Name:Hossam Al-Nasser
ID:2305508

Member3

Name:Hassan Saeed
ID:2305143

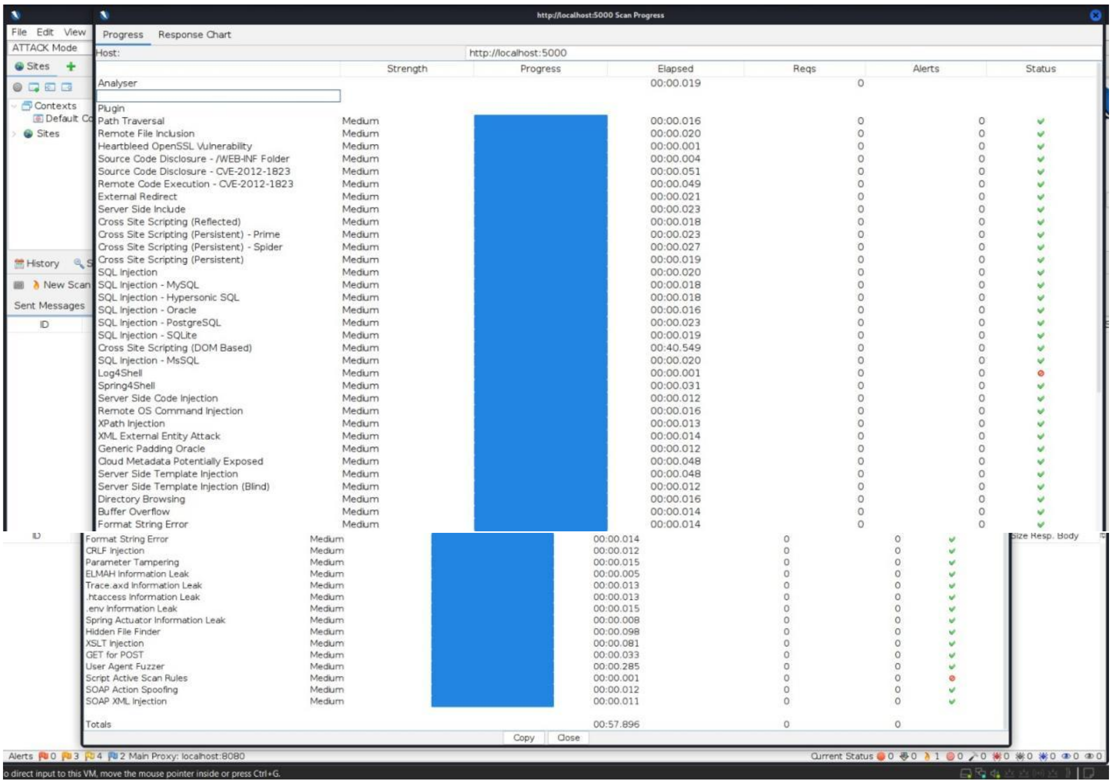
Phase A - Dynamic Testing (DAST)

A1.Automated Scan:



Discovered Endpoints:

- GET /index.html (Main application page)
- GET /register (User registration – accepts user input)
- GET /robots.txt (May expose hidden paths)
- GET /sitemap.xml (Lists application URLs)
- GET /vendor (Third-party libraries)
- GET /css/* (Static resources)
- GET /img/* (Static resources)



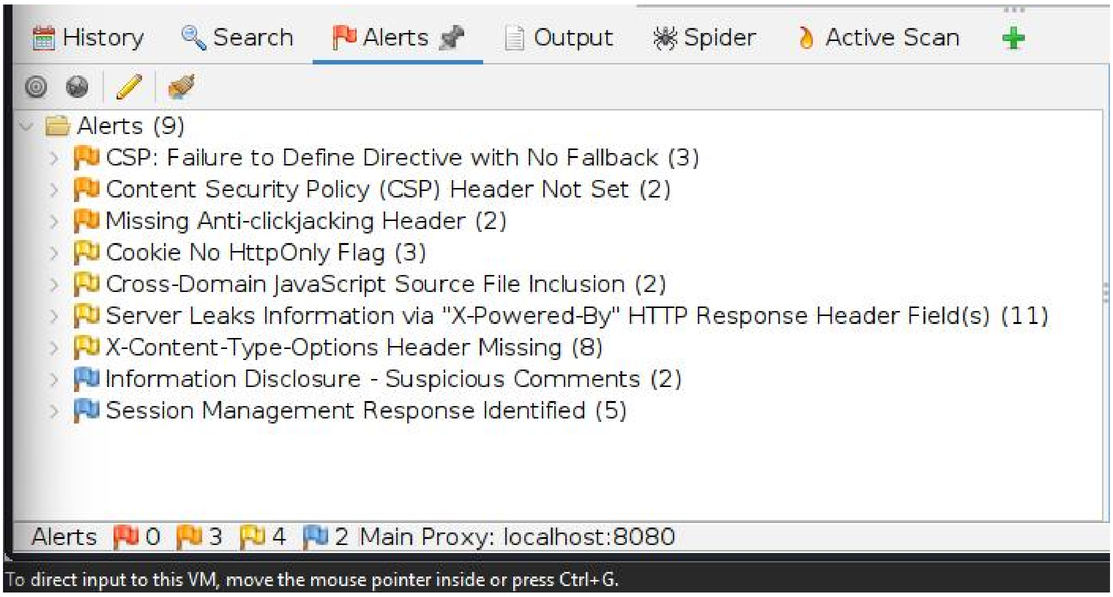
http://localhost:5000 Scan Progress

Host: http://localhost:5000

	Strength	Progress	Elapsed	Reqs	Alerts	Status
Analyser			00:00.019	0		
Plugin						
Path Traversal	Medium		00:00.016	0	0	✓
Remote File Inclusion	Medium		00:00.020	0	0	✓
Heartbleed OpenSSL Vulnerability	Medium		00:00.001	0	0	✓
Source Code Disclosure - /WEB-INF Folder	Medium		00:00.004	0	0	✓
Source Code Disclosure - CVE-2012-1823	Medium		00:00.051	0	0	✓
Remote Code Execution - CVE-2012-1823	Medium		00:00.049	0	0	✓
External Redirect	Medium		00:00.021	0	0	✓
Server Side Include	Medium		00:00.023	0	0	✓
Cross Site Scripting (Reflected)	Medium		00:00.018	0	0	✓
Cross Site Scripting (Persistent) - Prime	Medium		00:00.023	0	0	✓
Cross Site Scripting (Persistent) - Spider	Medium		00:00.027	0	0	✓
Cross Site Scripting (Persistent)	Medium		00:00.019	0	0	✓
SQL Injection	Medium		00:00.020	0	0	✓
SQL Injection - MySQL	Medium		00:00.018	0	0	✓
SQL Injection - Hypersonic SQL	Medium		00:00.018	0	0	✓
SQL Injection - Oracle	Medium		00:00.016	0	0	✓
SQL Injection - PostgreSQL	Medium		00:00.023	0	0	✓
SQL Injection - SQLite	Medium		00:00.019	0	0	✓
Cross Site Scripting (DOM Based)	Medium		00:40.549	0	0	✓
SQL Injection - MsSQL	Medium		00:00.020	0	0	✓
Log4Shell	Medium		00:00.001	0	0	●
Spring4Shell	Medium		00:00.031	0	0	✓
Server Side Code Injection	Medium		00:00.012	0	0	✓
Remote OS Command Injection	Medium		00:00.016	0	0	✓
XPath Injection	Medium		00:00.013	0	0	✓
XML External Entity Attack	Medium		00:00.014	0	0	✓
Generic Padding Oracle	Medium		00:00.012	0	0	✓
Cloud Metadata Potentially Exposed	Medium		00:00.048	0	0	✓
Server Side Template Injection	Medium		00:00.048	0	0	✓
Server Side Template Injection (Blind)	Medium		00:00.012	0	0	✓
Directory Browsing	Medium		00:00.016	0	0	✓
Buffer Overflow	Medium		00:00.014	0	0	✓
Format String Error	Medium		00:00.014	0	0	✓
Format String Error	Medium		00:00.014	0	0	✓
CRLF Injection	Medium		00:00.012	0	0	✓
Parameter Tampering	Medium		00:00.015	0	0	✓
ELMAH Information Leak	Medium		00:00.005	0	0	✓
Trace.axd Information Leak	Medium		00:00.013	0	0	✓
.htaccess Information Leak	Medium		00:00.013	0	0	✓
.env Information Leak	Medium		00:00.015	0	0	✓
Spring Actuator Information Leak	Medium		00:00.008	0	0	✓
Hidden File Finder	Medium		00:00.098	0	0	✓
XSLT Injection	Medium		00:00.081	0	0	✓
GET for POST	Medium		00:00.033	0	0	✓
User Agent Fuzzer	Medium		00:00.285	0	0	✓
Script Active Scan Rules	Medium		00:00.001	0	0	●
SOAP Action Spoofing	Medium		00:00.012	0	0	✓
SOAP XML Injection	Medium		00:00.011	0	0	✓
Totals			00:57.896	0	0	

Alerts 0 3 4 2 Main Proxy: localhost:8080 Current Status 0 0 1 0 0 0 0 0 0 0

o direct input to this VM, move the mouse pointer inside or press Ctrl+G.



- Alerts (9)
  - CSP: Failure to Define Directive with No Fallback (3)
  - Content Security Policy (CSP) Header Not Set (2)
  - Missing Anti-clickjacking Header (2)
  - Cookie No HttpOnly Flag (3)
  - Cross-Domain JavaScript Source File Inclusion (2)
  - Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s) (11)
  - X-Content-Type-Options Header Missing (8)
  - Information Disclosure - Suspicious Comments (2)
  - Session Management Response Identified (5)

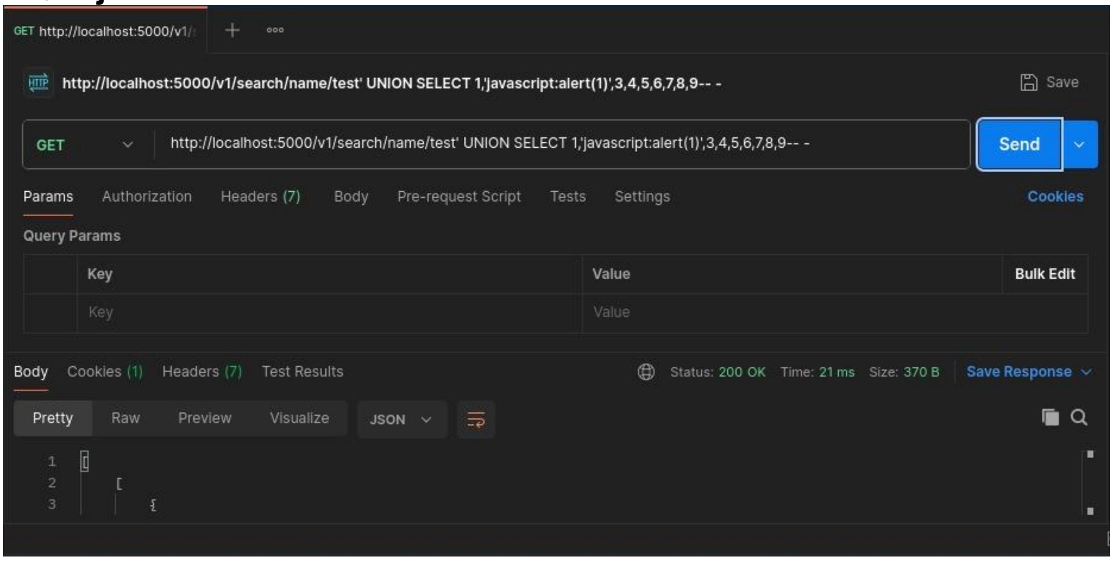
Alerts 0 3 4 2 Main Proxy: localhost:8080

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

The OWASP ZAP scan indicates potential vulnerabilities mainly related to security misconfigurations (These include missing or weak HTTP security headers such as Content Security Policy (CSP), anti-clickjacking headers, and X-Content-Type-Options, which may expose the application to attacks like reflected XSS, clickjacking, insecure cookie access, and information disclosure)

A2. Manual Testing & PoCs

SQLInjectoin



GET http://localhost:5000/v1/

http://localhost:5000/v1/search/name/test' UNION SELECT 1,'javascript:alert(1)',3,4,5,6,7,8,9-- -

GET http://localhost:5000/v1/search/name/test' UNION SELECT 1,'javascript:alert(1)',3,4,5,6,7,8,9-- -

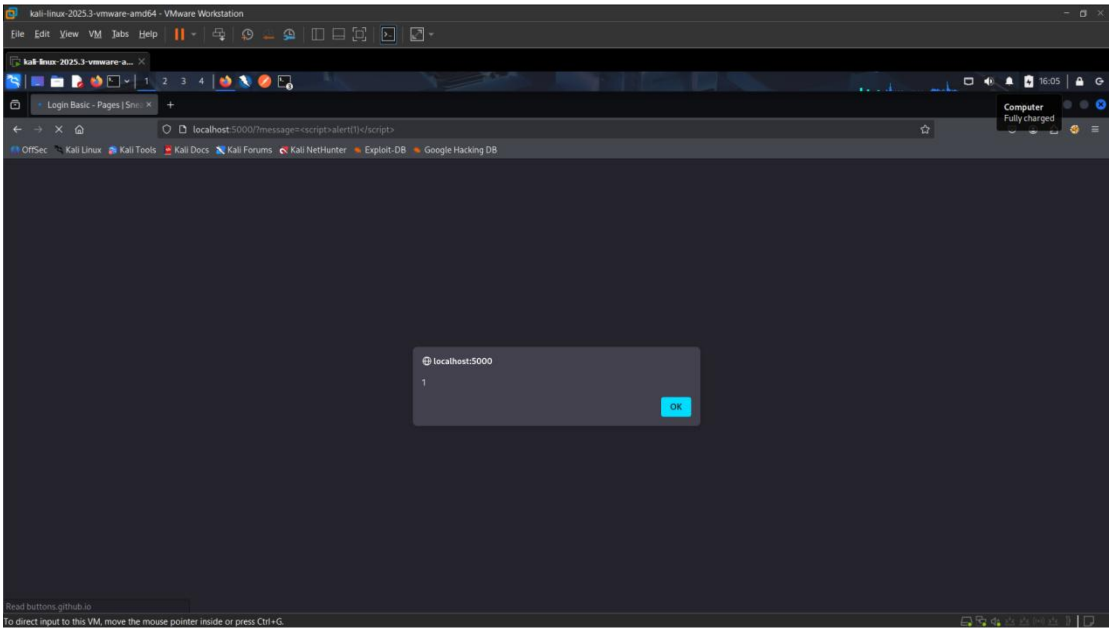
Key	Value	Bulk Edit
Key	Value	

Status: 200 OK Time: 21 ms Size: 370 B

```
[
    [
        {
```

SQL Injection occurs when an application improperly handles user input in SQL queries, allowing attackers to manipulate the database. Exploitation can lead to unauthorized data access, data modification or deletion, and in some cases full system compromise.

XSS



kali-linux-2025.3-vmware-amd64 - VMware Workstation

localhost:5000/?message=<script>alert(1)</script>

localhost:5000

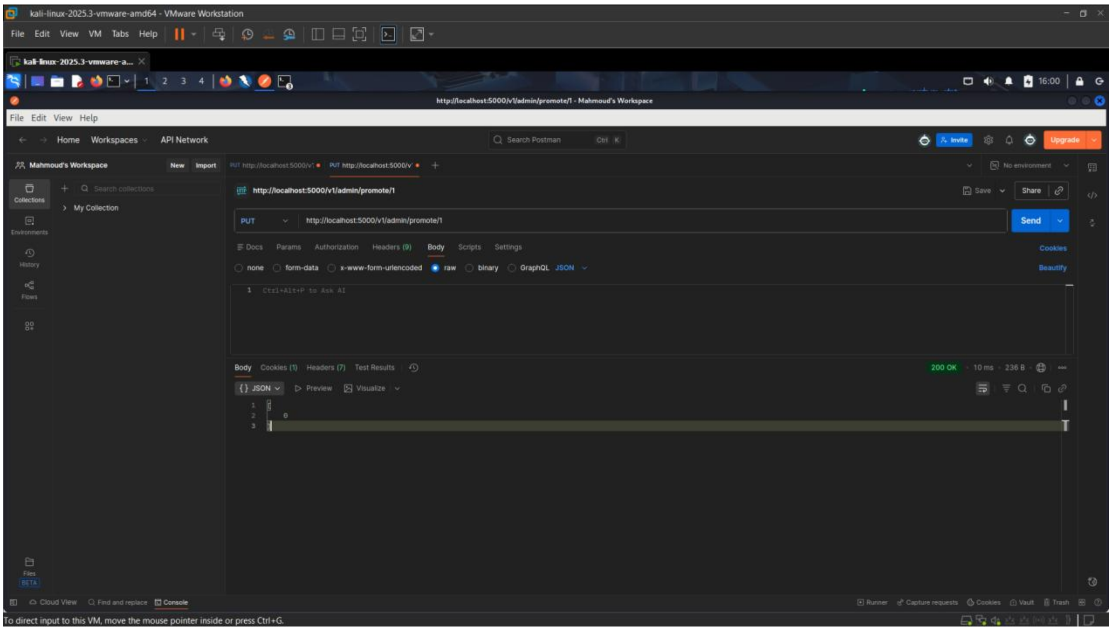
1

Read buttons.github.io

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

XSS occurs when an application improperly handles user-supplied input, allowing attackers to inject malicious scripts into web pages viewed by other users. This can lead to session hijacking, defacement, phishing attacks, or delivery of malware.

Broken auth / access control


kali-linux-2025.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
http://localhost:5000/v1/admin/promote/1 - Mahmoud's Workspace
Home Workspaces API Network
Search Postman
Invite
Upgrade
Mahmoud's Workspace
New Import
Collections
Environments
History
Flows
My Collection
No environment
Save
Share
Send
Docs Params Authorization Headers (9) Body Scripts Settings
Cookies
Beautify
Body Cookies (1) Headers (7) Test Results
Preview Visualize
Cloud View Find and replace Console
Runner Capture requests Cookies Vault Trash
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

PUT http://localhost:5000/v1/admin/promote/1

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

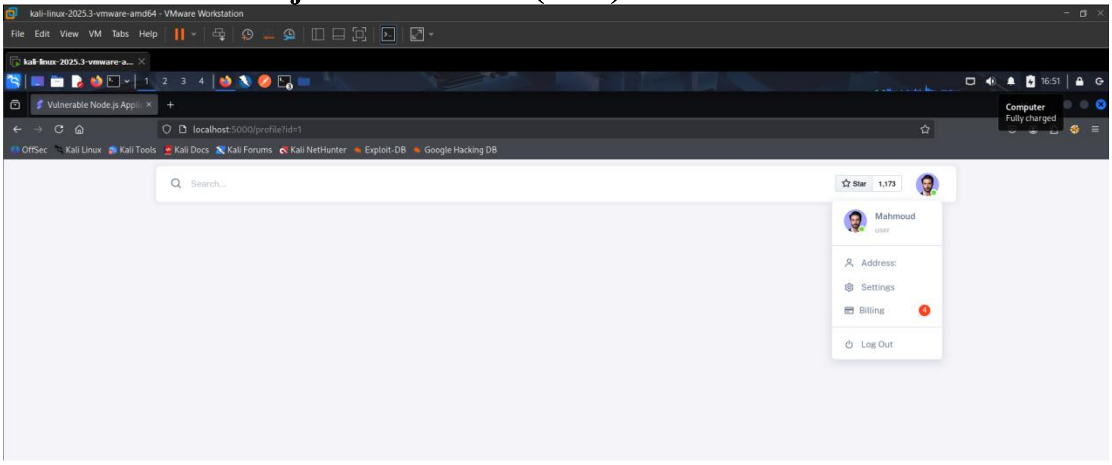
```
Ctrl+Alt+P to Ask AI
```

200 OK · 10 ms · 236 B

```
[
    0
]
```

This vulnerability occurs when an application fails to properly enforce user authentication or authorization. Attackers can exploit it to bypass login mechanisms, access restricted resources, or perform actions beyond their privileges. Common issues include weak password policies, session management flaws, predictable tokens, and misconfigured access controls. Exploitation can lead to unauthorized data access, privilege escalation, and complete account takeover.

Insecure direct object references (idor)


kali-linux-2025.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
Vulnerable Node.js Applic
Computer
Fully charged
OffSec Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB
Search...
Star 1,173
Settings
Billing
Log Out

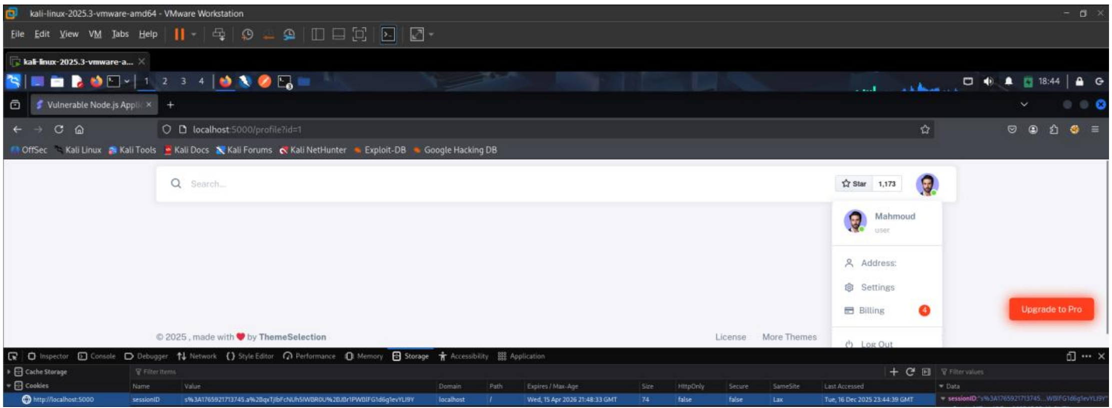
localhost:5000/profile?id=1

Mahmoud
user

Address:

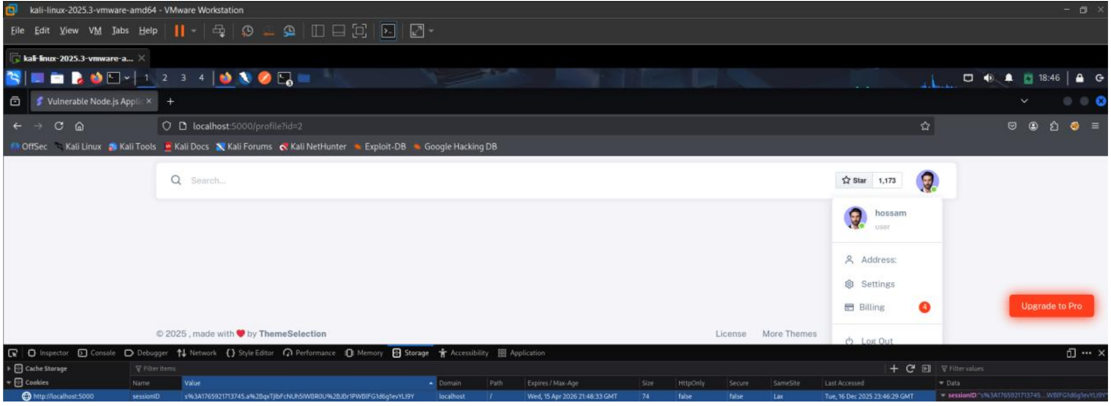
IDOR occurs when an application exposes internal objects (like files, database records, or user IDs) without proper access checks. Attackers can manipulate parameters to access data or perform actions they shouldn’t be allowed to, such as viewing other users’ accounts or modifying records. This can lead to data leakage, unauthorized modifications, or account compromise.

Insecure JWT handling (jwt)



localhost:5000/profile?id=1

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
sessionID	s%3A1765921713745.a%2BqxTjbFcNUh5IWBR0U%2BJ0r1PWBIFG1d6g1evYLI9Y	localhost	/	Wed, 15 Apr 2026 21:48:33 GMT	74	false	false	Lax	Tue, 16 Dec 2025 23:44:39 GMT

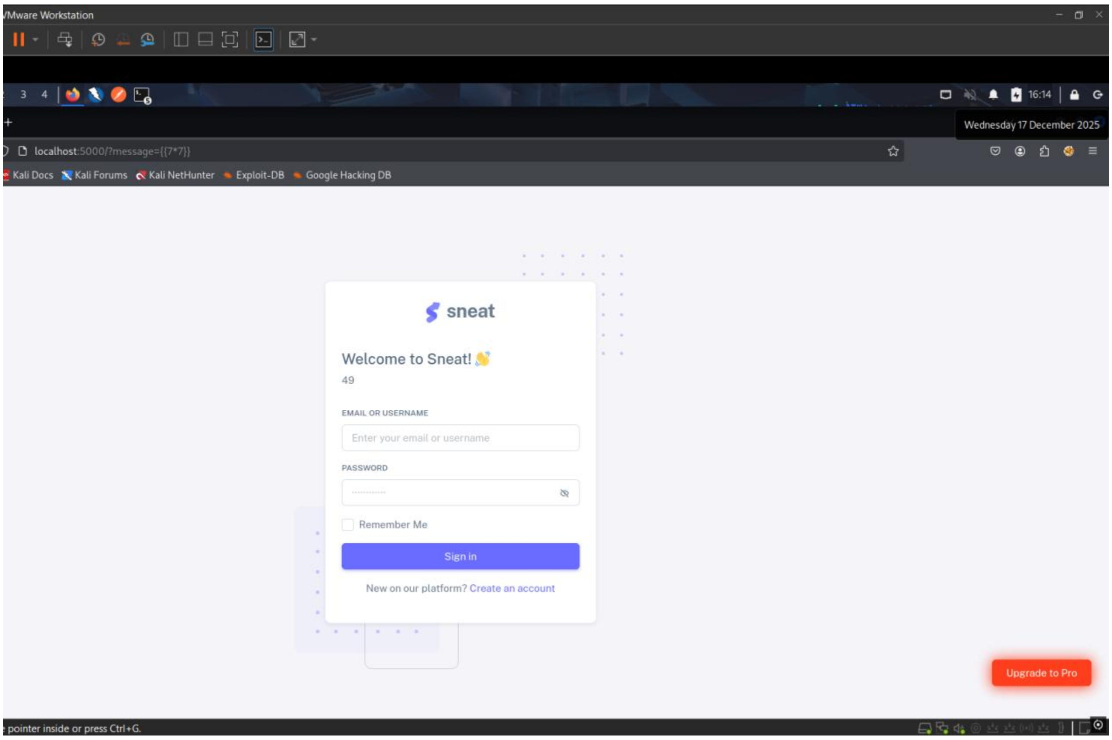


localhost:5000/profile?id=2

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
sessionID	s%3A1765921713745.a%2BqxTjbFcNUh5IWBR0U%2BJ0r1PWBIFG1d6g1evYLI9Y	localhost	/	Wed, 15 Apr 2026 21:48:33 GMT	74	false	false	Lax	Tue, 16 Dec 2025 23:46:29 GMT

This vulnerability arises when JSON Web Tokens (JWTs) are poorly implemented or validated. Attackers can exploit weak signing algorithms, expired tokens, or improper verification to bypass authentication, escalate privileges, or impersonate users, potentially leading to unauthorized access.

Injection (SSTI)



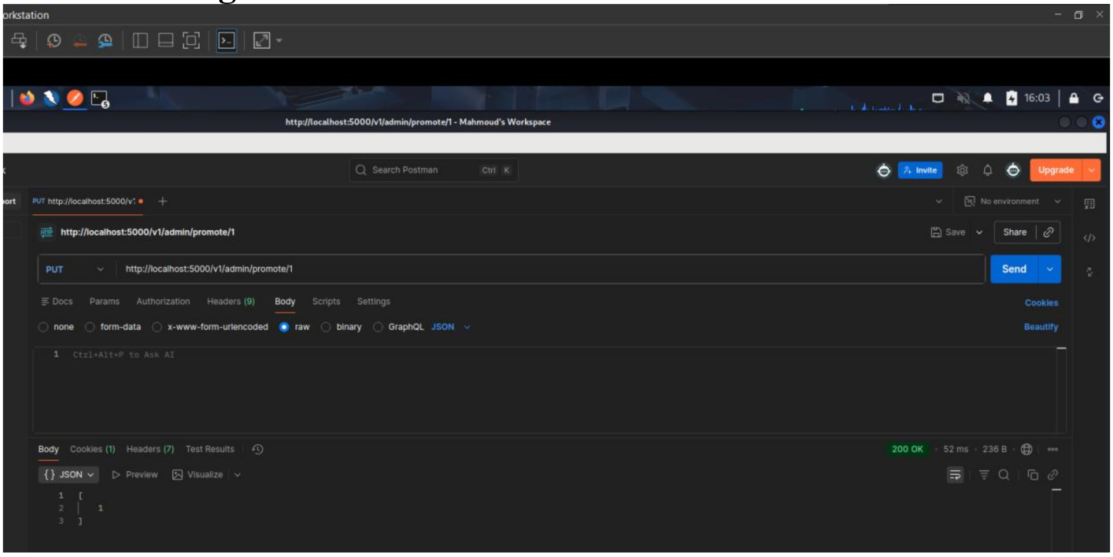
localhost:5000/?message={{7*7}}

Welcome to Sneat! 👋

49

SSTI occurs when user input is improperly handled in server-side templates, allowing attackers to inject and execute arbitrary code on the server. Exploitation can lead to data exposure, remote code execution, or full server compromise.

Insecure Design



http://localhost:5000/v1/admin/promote/1 - Mahmoud's Workspace

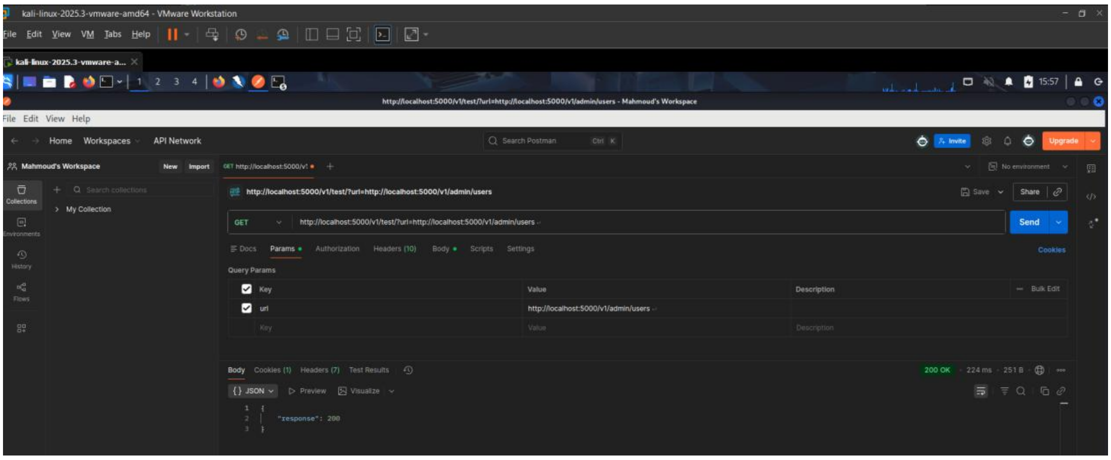
PUT http://localhost:5000/v1/admin/promote/1

200 OK · 52 ms · 236 B

```
[
  1
]
```

Insecure Design refers to flaws in the application’s architecture or logic that introduce security risks, even if the code is implemented correctly. These flaws can enable unauthorized access, data leaks, or bypass of security controls, making the system inherently vulnerable.

SERVER SIDE REQUES (SSRF)



GET http://localhost:5000/v1/test/?url=http://localhost:5000/v1/admin/users

Query Params

Key	Value	Description
☑ url	http://localhost:5000/v1/admin/users	

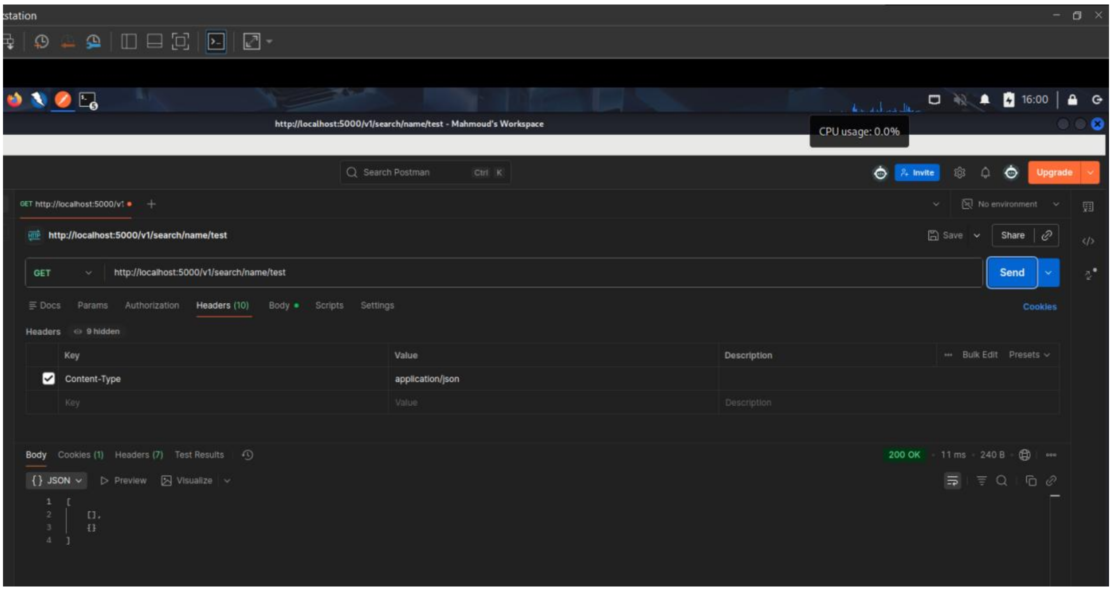
Body — 200 OK · 224 ms · 251 B

```
{
  "response": 200
}
```

kali-linux-2025.3-vmware-amd64 - VMware Workstation
http://localhost:5000/v1/test/?url=http://localhost:5000/v1/admin/users - Mahmoud's Workspace
Home
Workspaces
API Network
Search Postman
Mahmoud's Workspace
My Collection
Send
Params
Headers (10)
15:57

SSRF occurs when an application allows attackers to make arbitrary requests from the server. Exploitation can let attackers access internal systems, sensitive data, or internal APIs that are not publicly accessible, potentially leading to data theft or server compromise.

Security Misconfiguration



GET http://localhost:5000/v1/search/name/test

Headers

Key	Value	Description
☑ Content-Type	application/json	

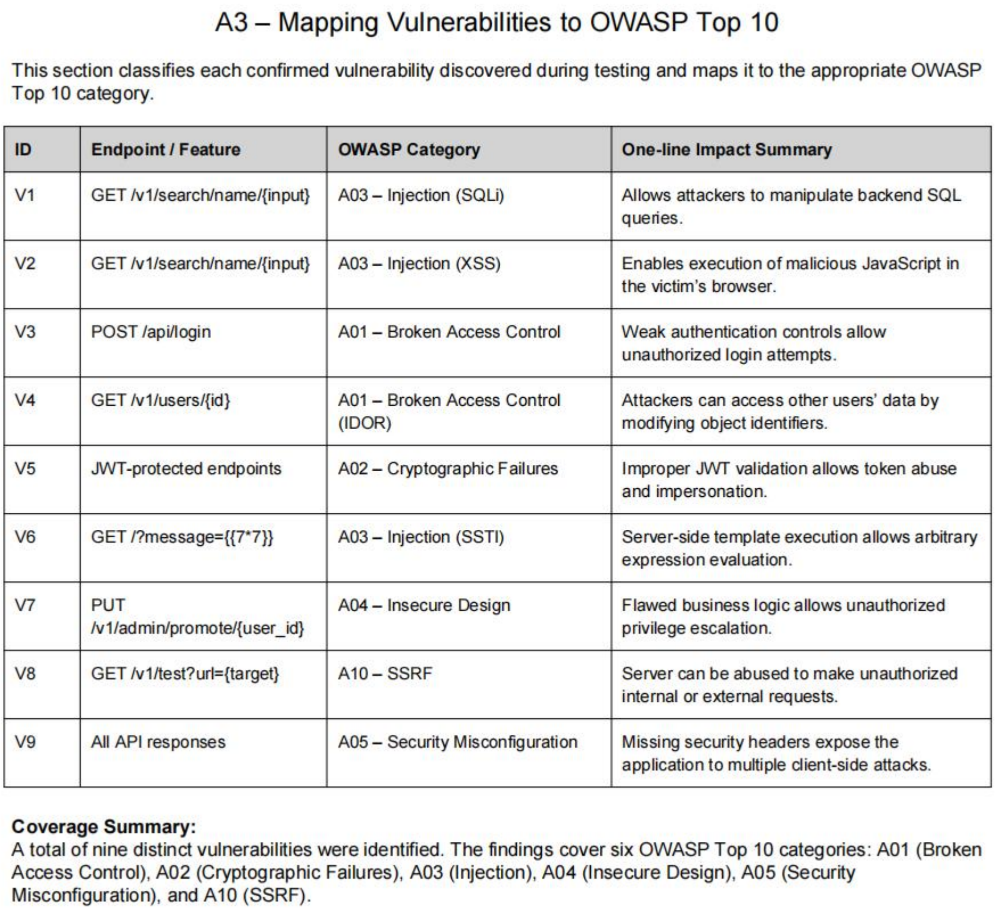
Body — 200 OK · 11 ms · 240 B

```
[
  [],
  {}
]
```

http://localhost:5000/v1/search/name/test - Mahmoud's Workspace
CPU usage: 0.0%
Search Postman
Send
Headers (10)
9 hidden
16:00

This vulnerability occurs when an application, server, or database is improperly configured, leaving it exposed to attacks. Examples include default credentials, unnecessary services, verbose error messages, or missing security headers. Exploitation can lead to information leakage, unauthorized access, or full system compromise.

A3



A3 – Mapping Vulnerabilities to OWASP Top 10

This section classifies each confirmed vulnerability discovered during testing and maps it to the appropriate OWASP Top 10 category.

ID	Endpoint / Feature	OWASP Category	One-line Impact Summary
V1	GET /v1/search/name/{input}	A03 – Injection (SQLi)	Allows attackers to manipulate backend SQL queries.
V2	GET /v1/search/name/{input}	A03 – Injection (XSS)	Enables execution of malicious JavaScript in the victim's browser.
V3	POST /api/login	A01 – Broken Access Control	Weak authentication controls allow unauthorized login attempts.
V4	GET /v1/users/{id}	A01 – Broken Access Control (IDOR)	Attackers can access other users' data by modifying object identifiers.
V5	JWT-protected endpoints	A02 – Cryptographic Failures	Improper JWT validation allows token abuse and impersonation.
V6	GET /?message={{7*7}}	A03 – Injection (SSTI)	Server-side template execution allows arbitrary expression evaluation.
V7	PUT /v1/admin/promote/{user_id}	A04 – Insecure Design	Flawed business logic allows unauthorized privilege escalation.
V8	GET /v1/test?url={target}	A10 – SSRF	Server can be abused to make unauthorized internal or external requests.
V9	All API responses	A05 – Security Misconfiguration	Missing security headers expose the application to multiple client-side attacks.

Coverage Summary:
A total of nine distinct vulnerabilities were identified. The findings cover six OWASP Top 10 categories: A01 (Broken Access Control), A02 (Cryptographic Failures), A03 (Injection), A04 (Insecure Design), A05 (Security Misconfiguration), and A10 (SSRF).

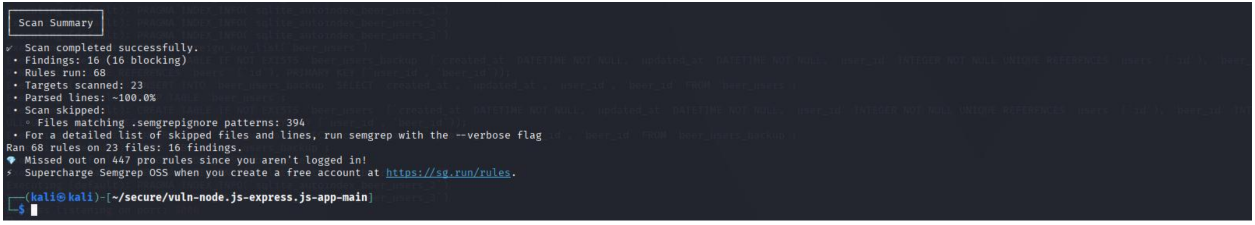
Phase B – Static Analysis with Semgrep (SAST)

B1 – Semgrep Static Analysis Summary

Semgrep was executed on the Node.js/Express application using the official JavaScript and Node.js rulesets.
The scan analyzed 23 JavaScript files and reported 16 security findings.
High-impact issues included:

- SQL Injection in Sequelize queries
- Server-Side Template Injection (SSTI) using Nunjucks
- Cross-Site Scripting (XSS) via direct response writing
- Open Redirect vulnerabilities
- Hardcoded secrets (JWT and session secrets)

Several of these findings directly correspond to vulnerabilities previously identified during DAST testing.



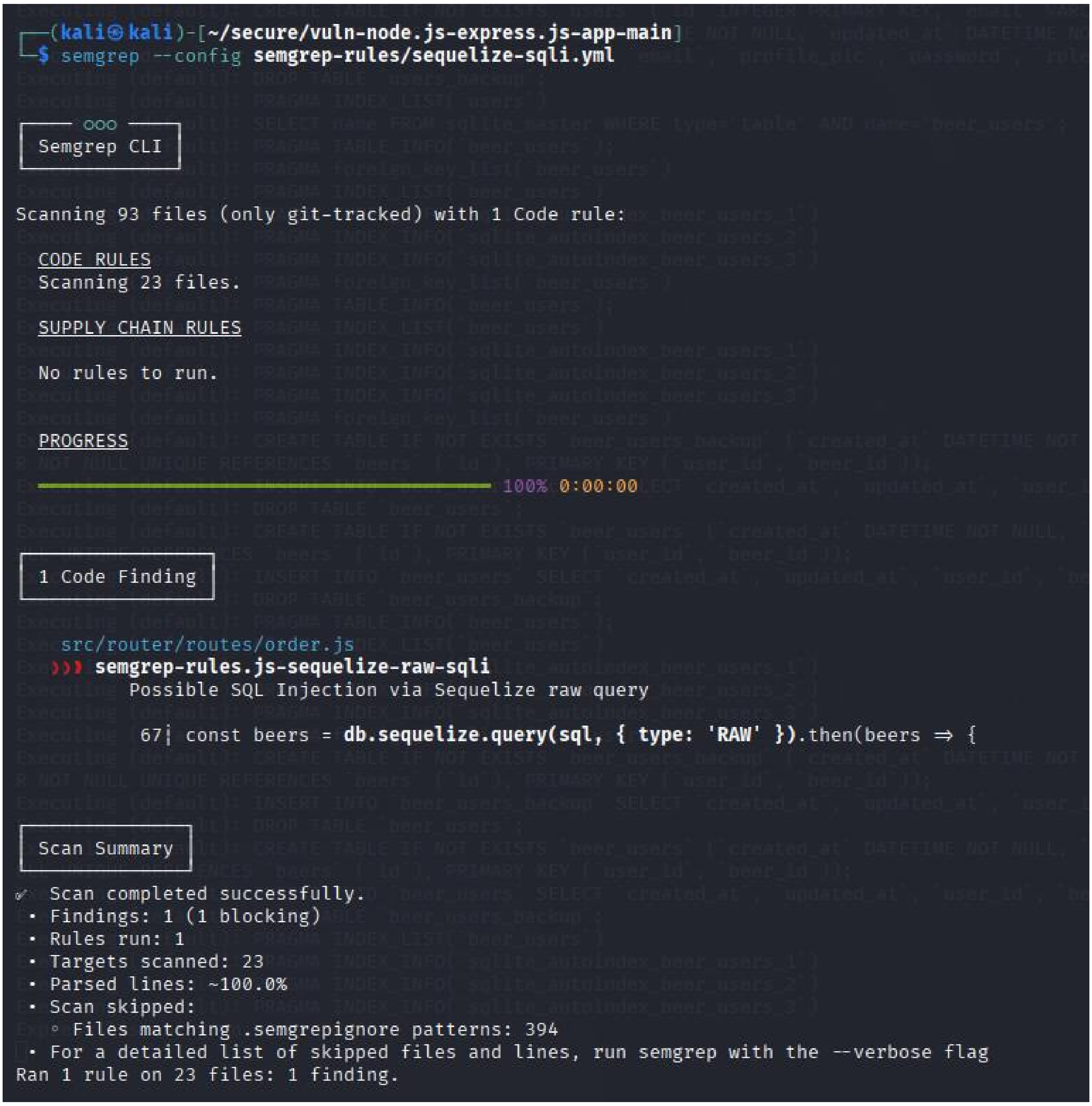
```
┌──────────────┐
│ Scan Summary │
└──────────────┘
✓  Scan completed successfully.
 • Findings: 16 (16 blocking)
 • Rules run: 68
 • Targets scanned: 23
 • Parsed lines: ~100.0%
 • Scan skipped:
   ◦ Files matching .semgrepignore patterns: 394
 • For a detailed list of skipped files and lines, run semgrep with the --verbose flag
Ran 68 rules on 23 files: 16 findings.
💎 Missed out on 447 pro rules since you aren't logged in!
⚡ Supercharge Semgrep OSS when you create a free account at https://sg.run/rules.

┌──(kali㉿kali)-[~/secure/vuln-node.js-express.js-app-main]
└─$ 
```

B2

Vuln ID	Endpoint / Feature	OWASP	File:Lines	Semgrep Rule
V1	Frontend message rendering	A03: Injection	src/router/routes/frontend.js:17–40	express-insecure-template-usage
V2	Order file read	A01: Path Traversal	src/router/routes/order.js:33	express-path-join-resolve-traversal
V2b	Order file read (Path Traversal)	A01: Path Traversal	src/router/routes/order.js:33	semgrep-rules.js-path-traversal
V3	Order SQL query	A03: Injection	src/router/routes/order.js:67	sequelize-injection-express
V4	System test response	A07: XSS	src/router/routes/system.js:18	direct-response-write
V5	Redirect endpoint	A10: Redirect	src/router/routes/system.js:37	express-open-redirect
V6	Object deserialization	A08: Software & Data Integrity	src/router/routes/system.js:64	third-party-object-deserialization
V7	JWT verification	A02: Cryptographic Failures	src/router/routes/user.js:18	hardcoded-jwt-secret
V8	Session configuration	A02: Cryptographic Failures	src/server.js:43–57	express-session-hardcoded-secret

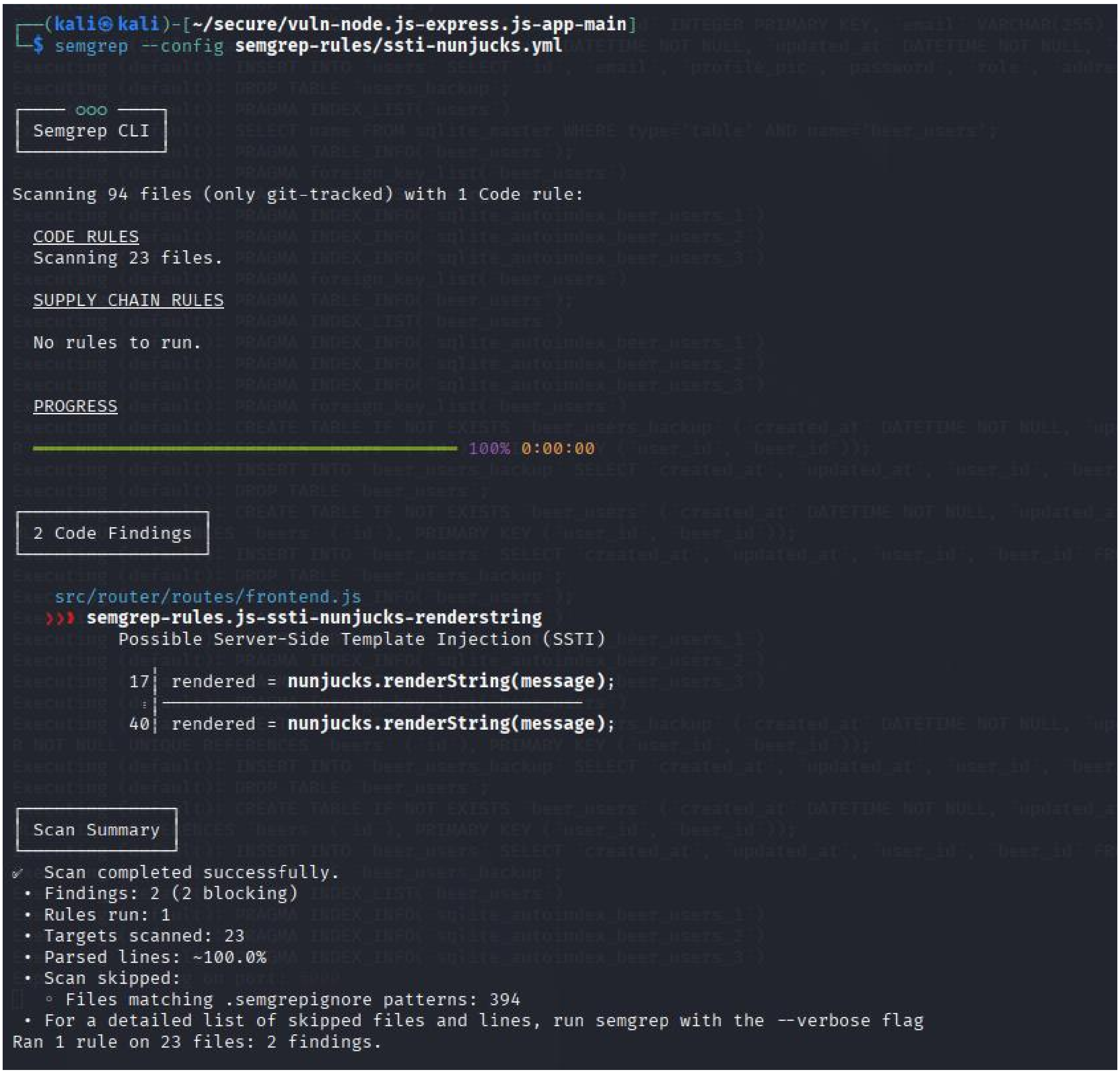
B3 –

1st rule Output



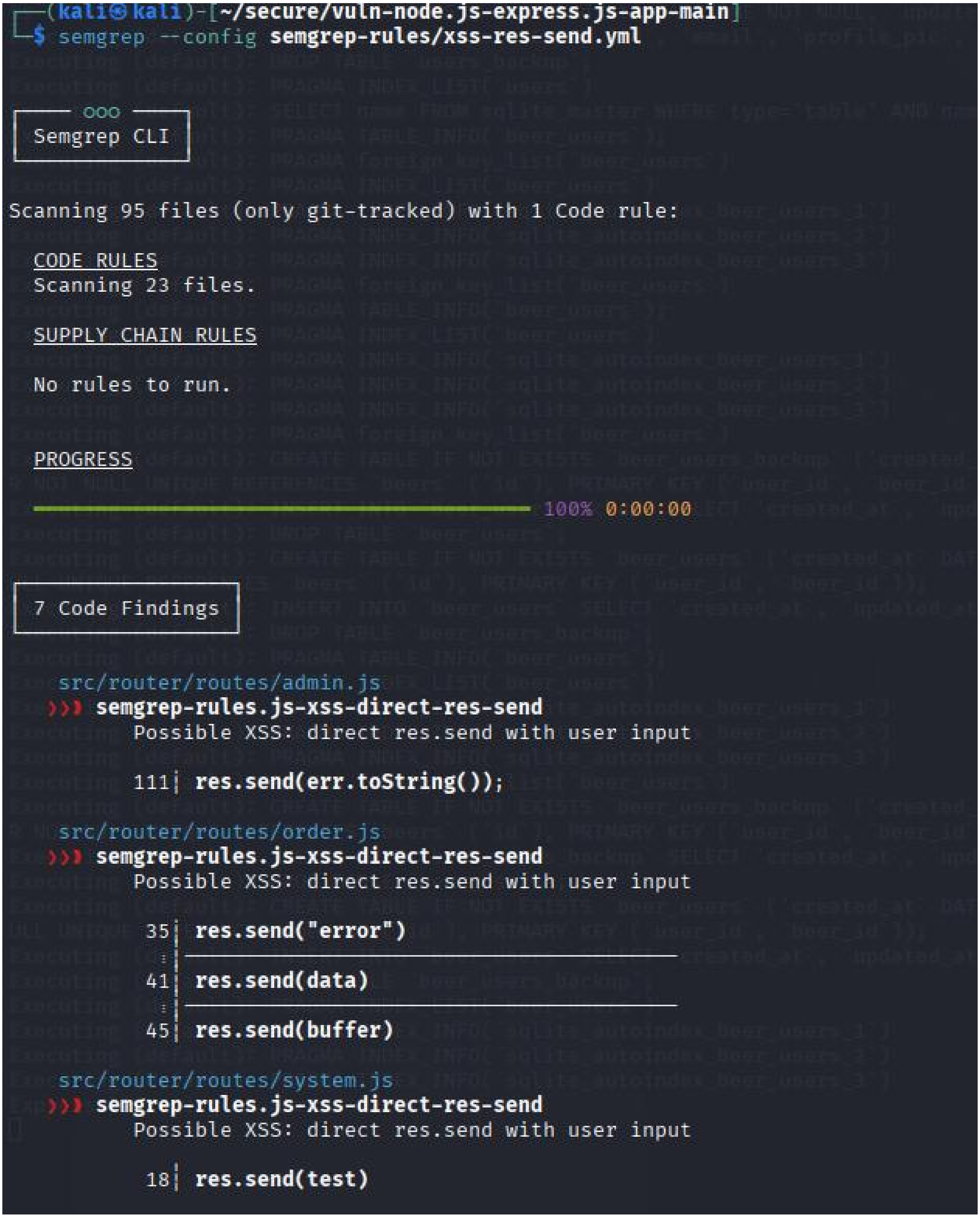
```
┌──(kali㉿kali)-[~/secure/vuln-node.js-express.js-app-main]
└─$ semgrep --config semgrep-rules/sequelize-sqli.yml


┌──── ○○○ ────┐
│ Semgrep CLI │
└─────────────┘


Scanning 93 files (only git-tracked) with 1 Code rule:

  CODE RULES
  Scanning 23 files.

  SUPPLY CHAIN RULES

  No rules to run.


  PROGRESS

  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00


┌────────────────┐
│ 1 Code Finding │
└────────────────┘

    src/router/routes/order.js
   ❯❯❱ semgrep-rules.js-sequelize-raw-sqli
          Possible SQL Injection via Sequelize raw query

           67┆ const beers = db.sequelize.query(sql, { type: 'RAW' }).then(beers ⇒ {


┌──────────────┐
│ Scan Summary │
└──────────────┘
✅ Scan completed successfully.
 • Findings: 1 (1 blocking)
 • Rules run: 1
 • Targets scanned: 23
 • Parsed lines: ~100.0%
 • Scan skipped:
   ◦ Files matching .semgrepignore patterns: 394
 • For a detailed list of skipped files and lines, run semgrep with the --verbose flag
Ran 1 rule on 23 files: 1 finding.
```

Rule 2 Output

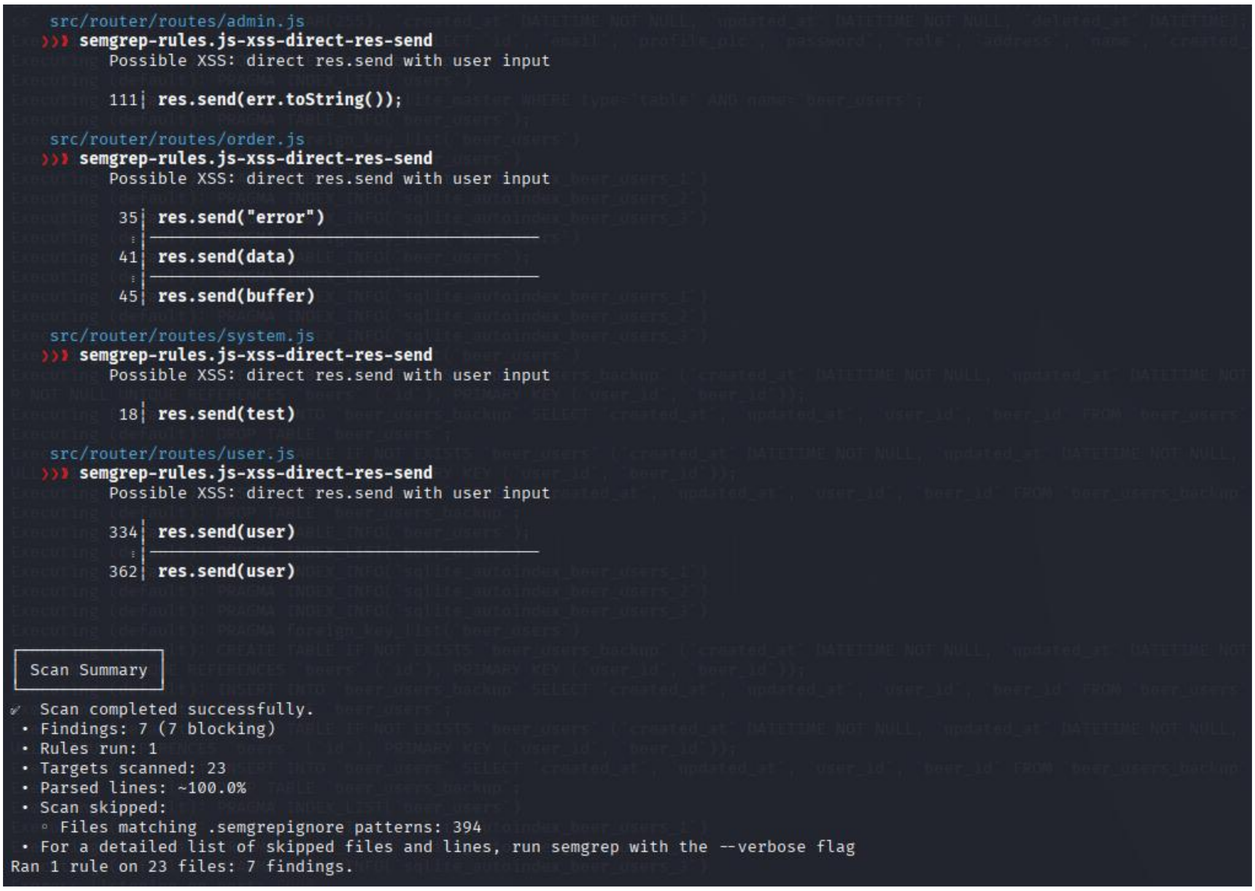


```
┌──(kali㉿kali)-[~/secure/vuln-node.js-express.js-app-main]
└─$ semgrep --config semgrep-rules/ssti-nunjucks.yml


┌──── ○○○ ────┐
│ Semgrep CLI │
└─────────────┘

Scanning 94 files (only git-tracked) with 1 Code rule:

  CODE RULES
  Scanning 23 files.

  SUPPLY CHAIN RULES

  No rules to run.


  PROGRESS

  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00


┌─────────────────┐
│ 2 Code Findings │
└─────────────────┘

    src/router/routes/frontend.js
   ❯❯❱ semgrep-rules.js-ssti-nunjucks-renderstring
          Possible Server-Side Template Injection (SSTI)

           17┆ rendered = nunjucks.renderString(message);
            ⋮┆----------------------------------------
           40┆ rendered = nunjucks.renderString(message);


┌──────────────┐
│ Scan Summary │
└──────────────┘
✅ Scan completed successfully.
 • Findings: 2 (2 blocking)
 • Rules run: 1
 • Targets scanned: 23
 • Parsed lines: ~100.0%
 • Scan skipped:
   ◦ Files matching .semgrepignore patterns: 394
 • For a detailed list of skipped files and lines, run semgrep with the --verbose flag
Ran 1 rule on 23 files: 2 findings.
```

Rule 3 Output

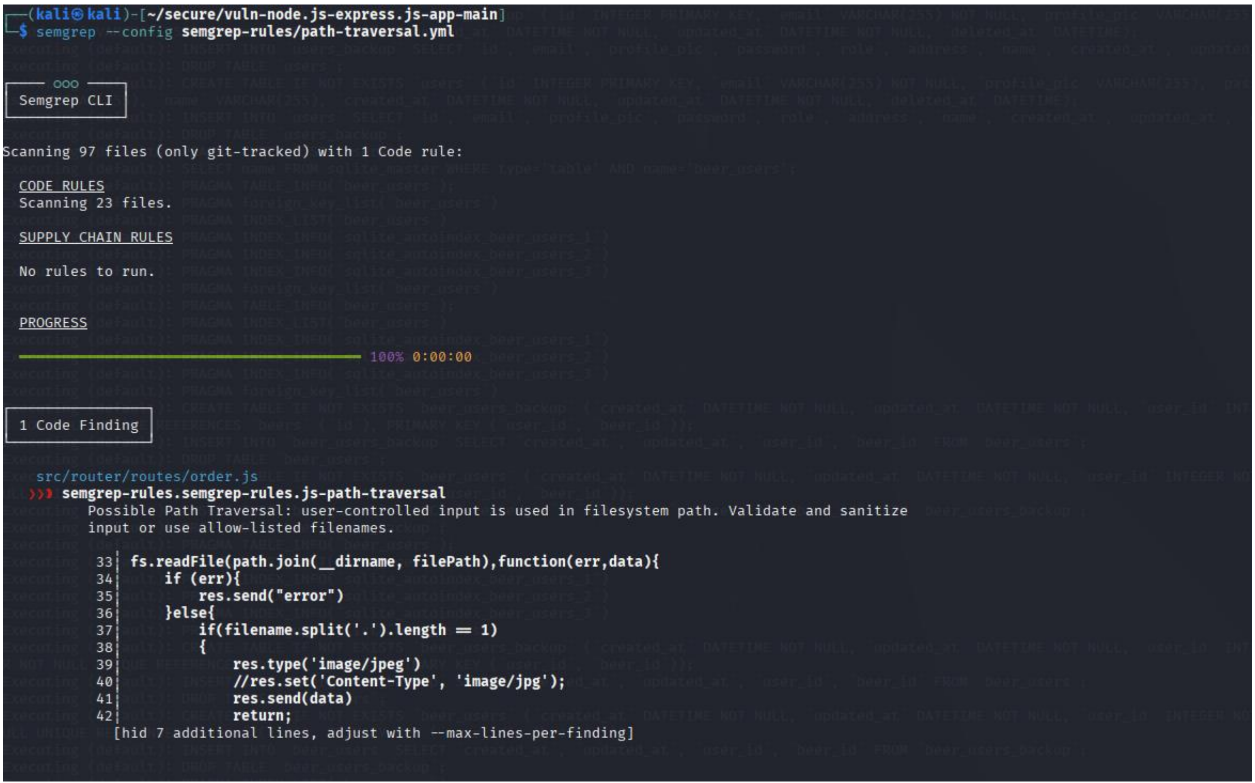


```
┌──(kali㉿kali)-[~/secure/vuln-node.js-express.js-app-main]
└─$ semgrep --config semgrep-rules/xss-res-send.yml


┌──── ○○○ ────┐
│ Semgrep CLI │
└─────────────┘


Scanning 95 files (only git-tracked) with 1 Code rule:

  CODE RULES
  Scanning 23 files.

  SUPPLY CHAIN RULES

  No rules to run.


  PROGRESS

  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00


┌──────────────────┐
│ 7 Code Findings  │
└──────────────────┘

    src/router/routes/admin.js
   ❯❯❱ semgrep-rules.js-xss-direct-res-send
          Possible XSS: direct res.send with user input

          111┆ res.send(err.toString());

    src/router/routes/order.js
   ❯❯❱ semgrep-rules.js-xss-direct-res-send
          Possible XSS: direct res.send with user input

           35┆ res.send("error")
            ⋮┆----------------------------------------
           41┆ res.send(data)
            ⋮┆----------------------------------------
           45┆ res.send(buffer)

    src/router/routes/system.js
   ❯❯❱ semgrep-rules.js-xss-direct-res-send
          Possible XSS: direct res.send with user input

           18┆ res.send(test)
```

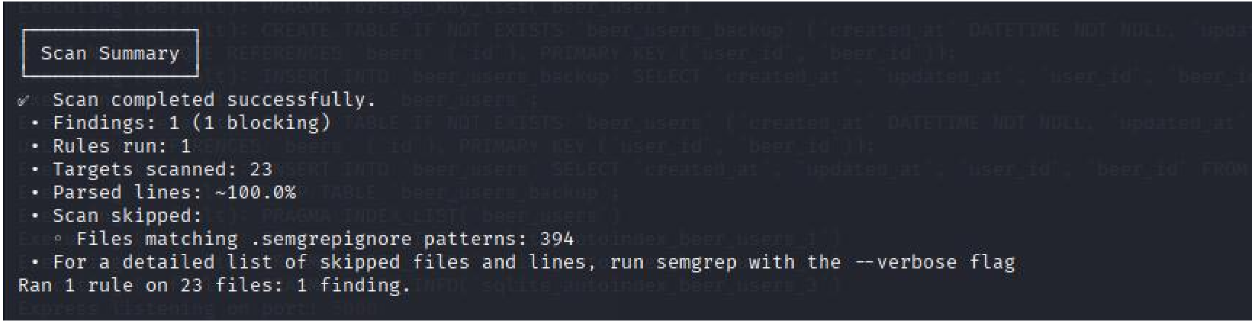


```
  src/router/routes/admin.js
   ❯❯❯ semgrep-rules.js-xss-direct-res-send
          Possible XSS: direct res.send with user input

          111┆ res.send(err.toString());
  src/router/routes/order.js
   ❯❯❯ semgrep-rules.js-xss-direct-res-send
          Possible XSS: direct res.send with user input

           35┆ res.send("error")
            ⋮┆----------------------------------------
           41┆ res.send(data)
            ⋮┆----------------------------------------
           45┆ res.send(buffer)
  src/router/routes/system.js
   ❯❯❯ semgrep-rules.js-xss-direct-res-send
          Possible XSS: direct res.send with user input

           18┆ res.send(test)
  src/router/routes/user.js
   ❯❯❯ semgrep-rules.js-xss-direct-res-send
          Possible XSS: direct res.send with user input

          334┆ res.send(user)
            ⋮┆----------------------------------------
          362┆ res.send(user)


┌──────────────┐
│ Scan Summary │
└──────────────┘
✔ Scan completed successfully.
 • Findings: 7 (7 blocking)
 • Rules run: 1
 • Targets scanned: 23
 • Parsed lines: ~100.0%
 • Scan skipped:
   ◦ Files matching .semgrepignore patterns: 394
 • For a detailed list of skipped files and lines, run semgrep with the --verbose flag
Ran 1 rule on 23 files: 7 findings.
```

Rule 4 Output

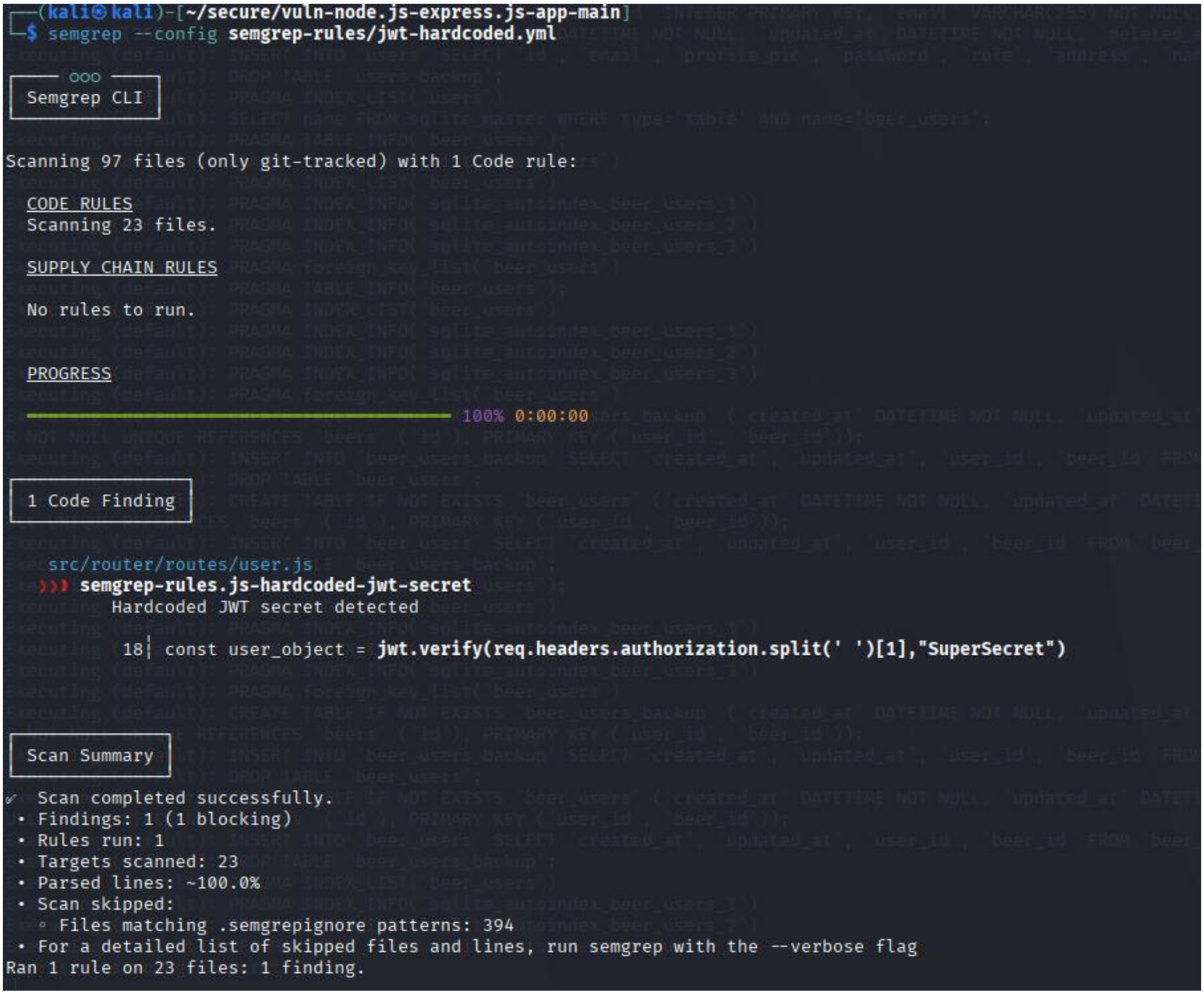


```
┌──(kali㉿kali)-[~/secure/vuln-node.js-express.js-app-main]
└─$ semgrep --config semgrep-rules/path-traversal.yml


┌──── ○○○ ────┐
│ Semgrep CLI │
└─────────────┘


Scanning 97 files (only git-tracked) with 1 Code rule:

  CODE RULES
  Scanning 23 files.

  SUPPLY CHAIN RULES

  No rules to run.


  PROGRESS

  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00


┌────────────────┐
│ 1 Code Finding │
└────────────────┘

    src/router/routes/order.js
   ❯❯❱ semgrep-rules.semgrep-rules.js-path-traversal
          Possible Path Traversal: user-controlled input is used in filesystem path. Validate and sanitize
          input or use allow-listed filenames.

            fs.readFile(path.join(__dirname, filePath),function(err,data){
                if (err){
                    res.send("error")
                }else{
                    if(filename.split('.').length == 1)
                    {
                        res.type('image/jpeg')
                        //res.set('Content-Type', 'image/jpg');
                        res.send(data)
                        return;
             [hid 7 additional lines, adjust with --max-lines-per-finding]
```



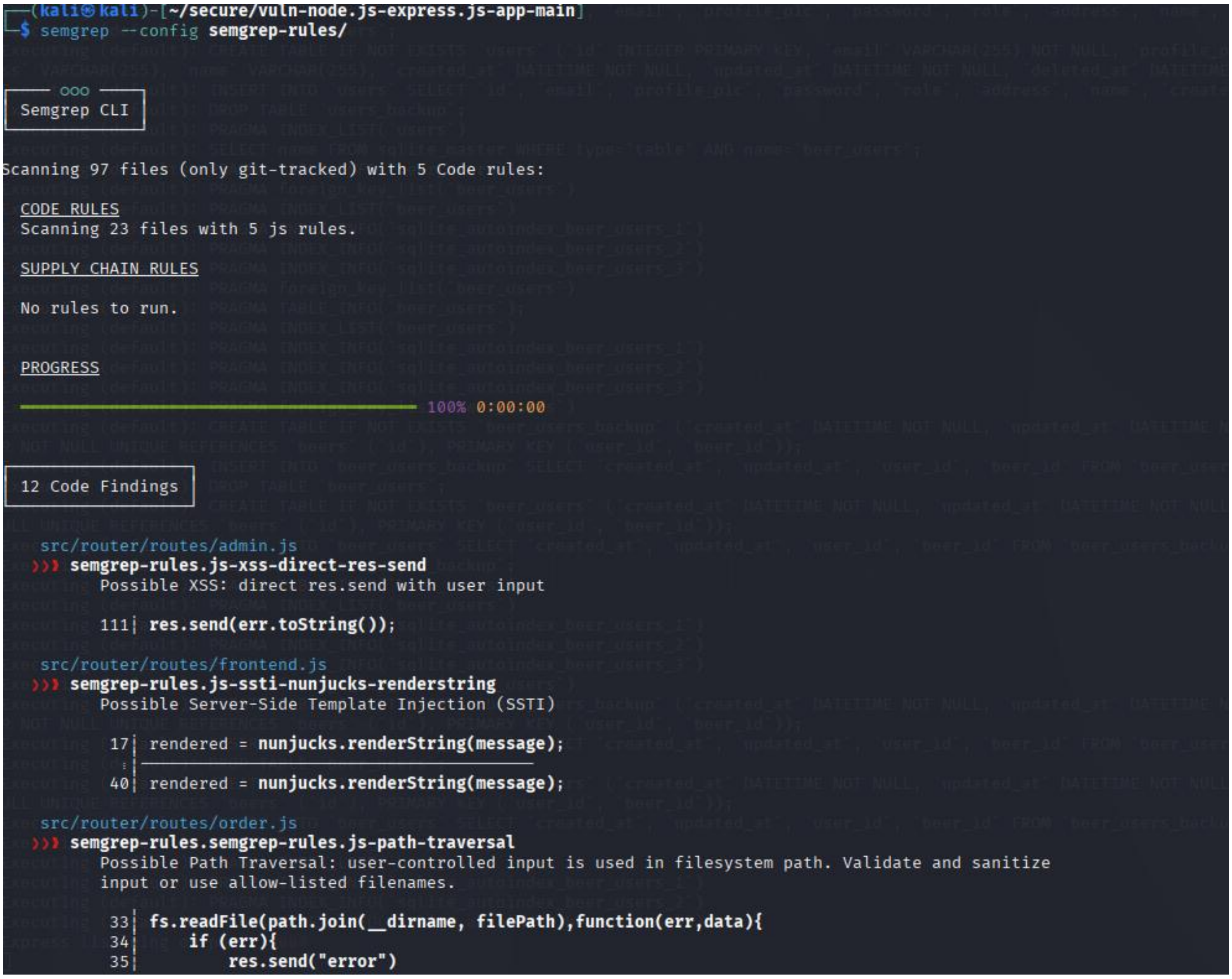
```
┌──────────────┐
│ Scan Summary │
└──────────────┘
✅ Scan completed successfully.
 • Findings: 1 (1 blocking)
 • Rules run: 1
 • Targets scanned: 23
 • Parsed lines: ~100.0%
 • Scan skipped:
   ◦ Files matching .semgrepignore patterns: 394
 • For a detailed list of skipped files and lines, run semgrep with the --verbose flag
Ran 1 rule on 23 files: 1 finding.
```

Rule 5 Output

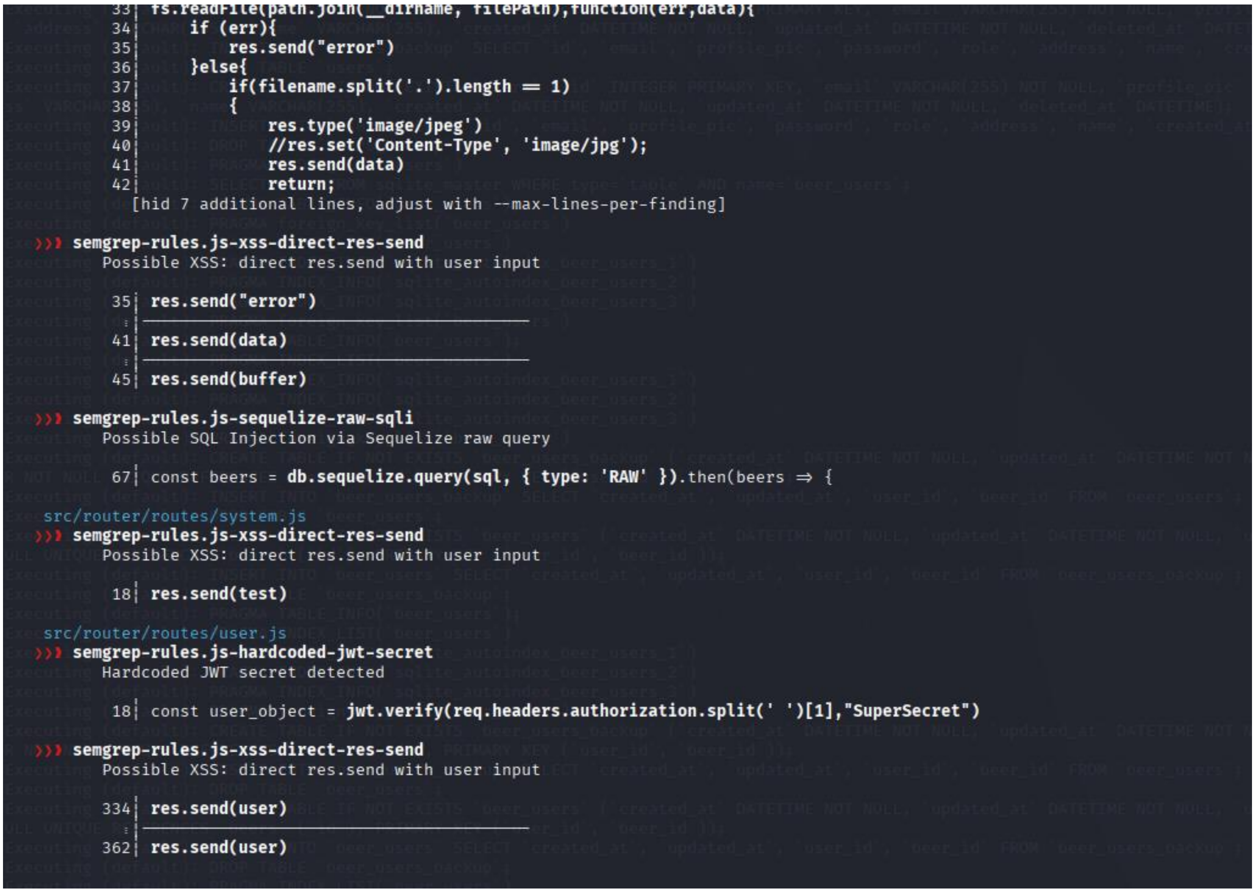


```
┌──(kali㉿kali)-[~/secure/vuln-node.js-express.js-app-main]
└─$ semgrep --config semgrep-rules/jwt-hardcoded.yml

┌──── ○○○ ────┐
│ Semgrep CLI │
└─────────────┘

Scanning 97 files (only git-tracked) with 1 Code rule:

  CODE RULES
  Scanning 23 files.

  SUPPLY CHAIN RULES

  No rules to run.


  PROGRESS

  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00


┌────────────────┐
│ 1 Code Finding │
└────────────────┘

    src/router/routes/user.js
   ❯❯❱ semgrep-rules.js-hardcoded-jwt-secret
          Hardcoded JWT secret detected

           18┆ const user_object = jwt.verify(req.headers.authorization.split(' ')[1],"SuperSecret")


┌──────────────┐
│ Scan Summary │
└──────────────┘
✔ Scan completed successfully.
 • Findings: 1 (1 blocking)
 • Rules run: 1
 • Targets scanned: 23
 • Parsed lines: ~100.0%
 • Scan skipped:
   ◦ Files matching .semgrepignore patterns: 394
 • For a detailed list of skipped files and lines, run semgrep with the --verbose flag
Ran 1 rule on 23 files: 1 finding.
```

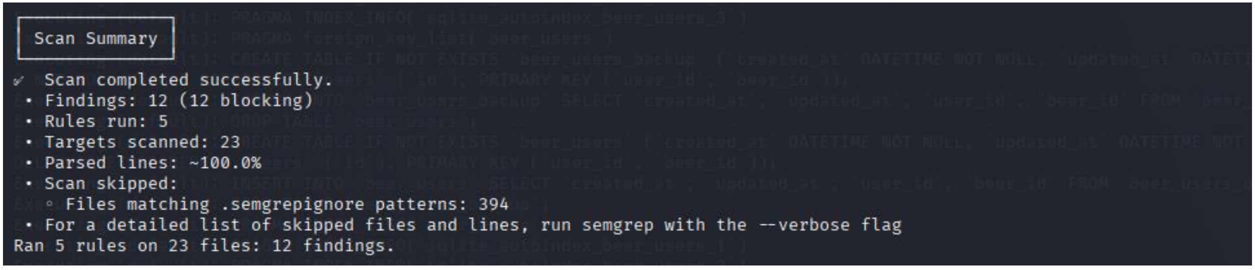
All the rules together



```
┌──(kali㉿kali)-[~/secure/vuln-node.js-express.js-app-main]
└─$ semgrep --config semgrep-rules/


┌──── ○○○ ────┐
│ Semgrep CLI │
└─────────────┘


Scanning 97 files (only git-tracked) with 5 Code rules:

  CODE RULES
  Scanning 23 files with 5 js rules.

  SUPPLY CHAIN RULES

  No rules to run.


  PROGRESS

  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00


┌──────────────────┐
│ 12 Code Findings │
└──────────────────┘

    src/router/routes/admin.js
   ❯❯❱ semgrep-rules.js-xss-direct-res-send
          Possible XSS: direct res.send with user input

          111┆ res.send(err.toString());

    src/router/routes/frontend.js
   ❯❯❱ semgrep-rules.js-ssti-nunjucks-renderstring
          Possible Server-Side Template Injection (SSTI)

           17┆ rendered = nunjucks.renderString(message);
            ⋮┆----------------------------------------
           40┆ rendered = nunjucks.renderString(message);

    src/router/routes/order.js
   ❯❯❱ semgrep-rules.semgrep-rules.js-path-traversal
          Possible Path Traversal: user-controlled input is used in filesystem path. Validate and sanitize
          input or use allow-listed filenames.

           33┆ fs.readFile(path.join(__dirname, filePath),function(err,data){
           34┆     if (err){
           35┆         res.send("error")
```



```
         33┆ fs.readFile(path.join(__dirname, filePath),function(err,data){
         34┆     if (err){
         35┆         res.send("error")
         36┆     }else{
         37┆         if(filename.split('.').length == 1)
         38┆         {
         39┆             res.type('image/jpeg')
         40┆             //res.set('Content-Type', 'image/jpg');
         41┆             res.send(data)
         42┆             return;
          [hid 7 additional lines, adjust with --max-lines-per-finding]

   ❯❯❯ semgrep-rules.js-xss-direct-res-send
          Possible XSS: direct res.send with user input

         35┆ res.send("error")
          ⋮┆----------------------------------------
         41┆ res.send(data)
          ⋮┆----------------------------------------
         45┆ res.send(buffer)

   ❯❯❯ semgrep-rules.js-sequelize-raw-sqli
          Possible SQL Injection via Sequelize raw query

         67┆ const beers = db.sequelize.query(sql, { type: 'RAW' }).then(beers => {

  src/router/routes/system.js
   ❯❯❯ semgrep-rules.js-xss-direct-res-send
          Possible XSS: direct res.send with user input

         18┆ res.send(test)

  src/router/routes/user.js
   ❯❯❯ semgrep-rules.js-hardcoded-jwt-secret
          Hardcoded JWT secret detected

         18┆ const user_object = jwt.verify(req.headers.authorization.split(' ')[1],"SuperSecret")

   ❯❯❯ semgrep-rules.js-xss-direct-res-send
          Possible XSS: direct res.send with user input

        334┆ res.send(user)
          ⋮┆----------------------------------------
        362┆ res.send(user)
```

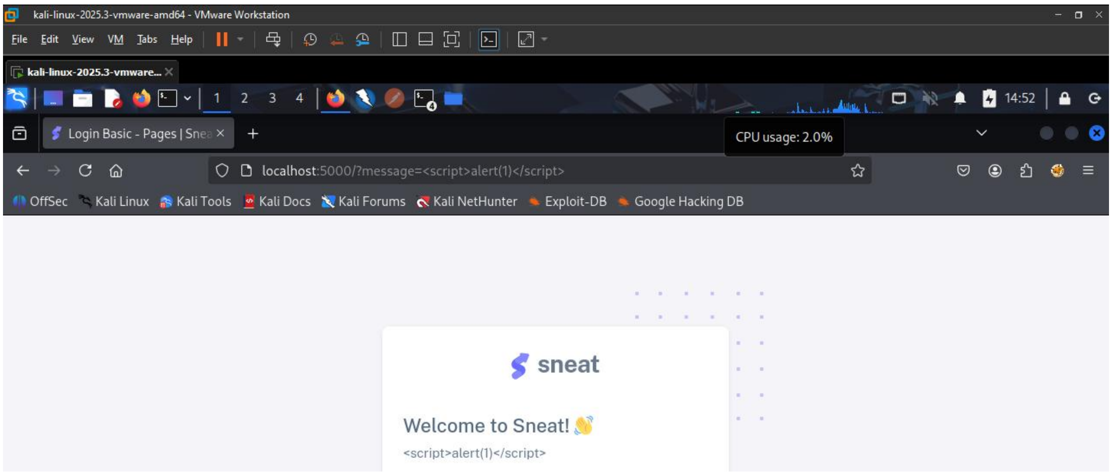


```
┌──────────────┐
│ Scan Summary │
└──────────────┘
✅ Scan completed successfully.
 • Findings: 12 (12 blocking)
 • Rules run: 5
 • Targets scanned: 23
 • Parsed lines: ~100.0%
 • Scan skipped:
   ◦ Files matching .semgrepignore patterns: 394
 • For a detailed list of skipped files and lines, run semgrep with the --verbose flag
Ran 5 rules on 23 files: 12 findings.
```

Phase c

C1 :

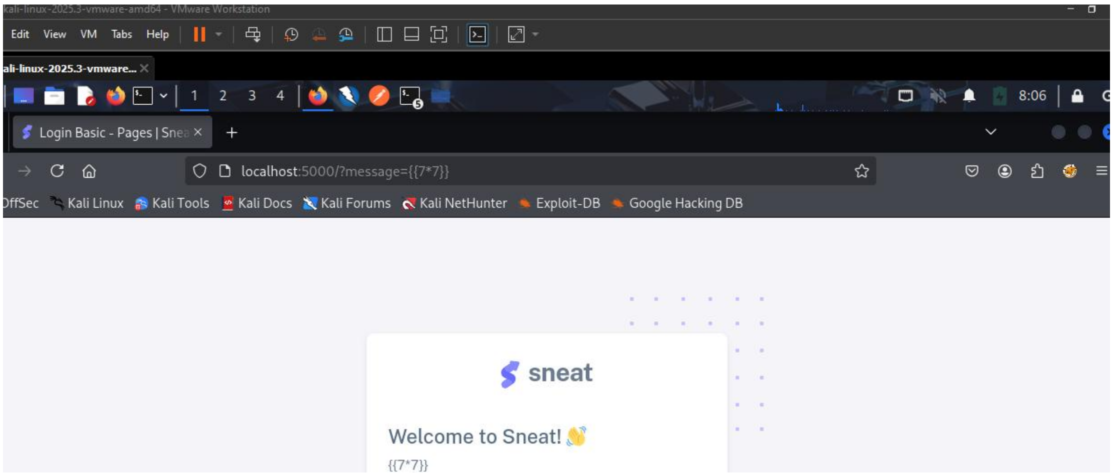
V1-2 – SSTI & Reflected XSS



kali-linux-2025.3-vmware-amd64 - VMware Workstation
Login Basic - Pages | Sneat
CPU usage: 2.0%
localhost:5000/?message=<script>alert(1)</script>
sneat

Welcome to Sneat! 👋

<script>alert(1)</script>



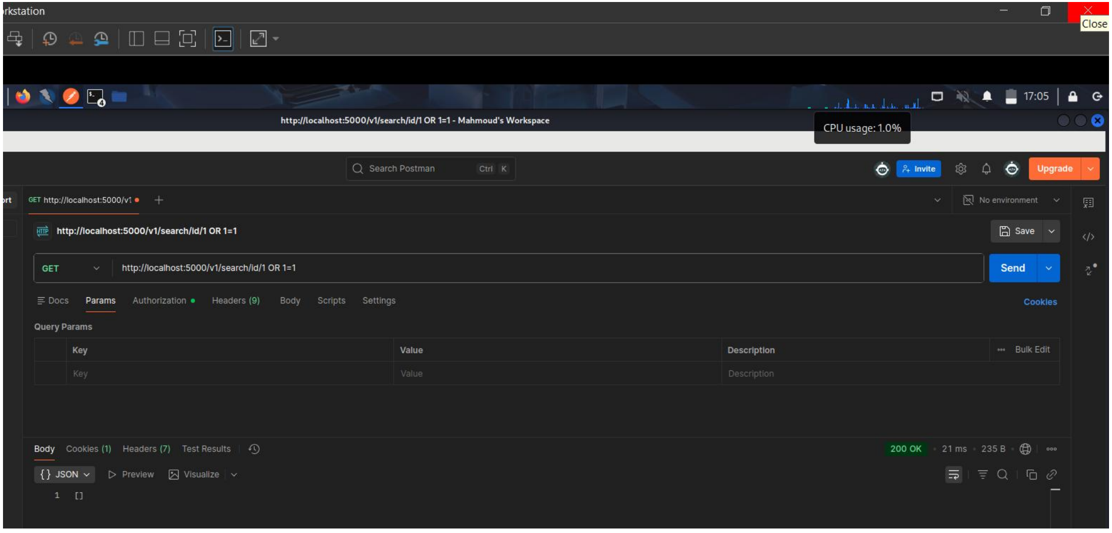
kali-linux-2025.3-vmware-amd64 - VMware Workstation
Login Basic - Pages | Sneat
localhost:5000/?message={{7*7}}
sneat

Welcome to Sneat! 👋

{{7*7}}

V3-4

Sql injection



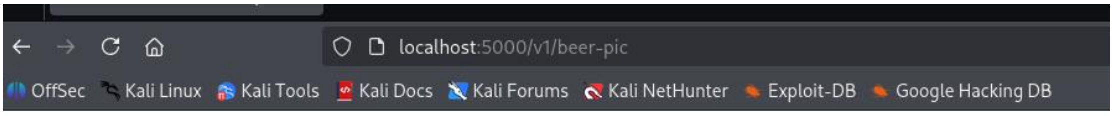
http://localhost:5000/v1/search/id/1 OR 1=1 - Mahmoud's Workspace

GET http://localhost:5000/v1/search/id/1 OR 1=1

200 OK · 21 ms · 235 B

```
1  []
```

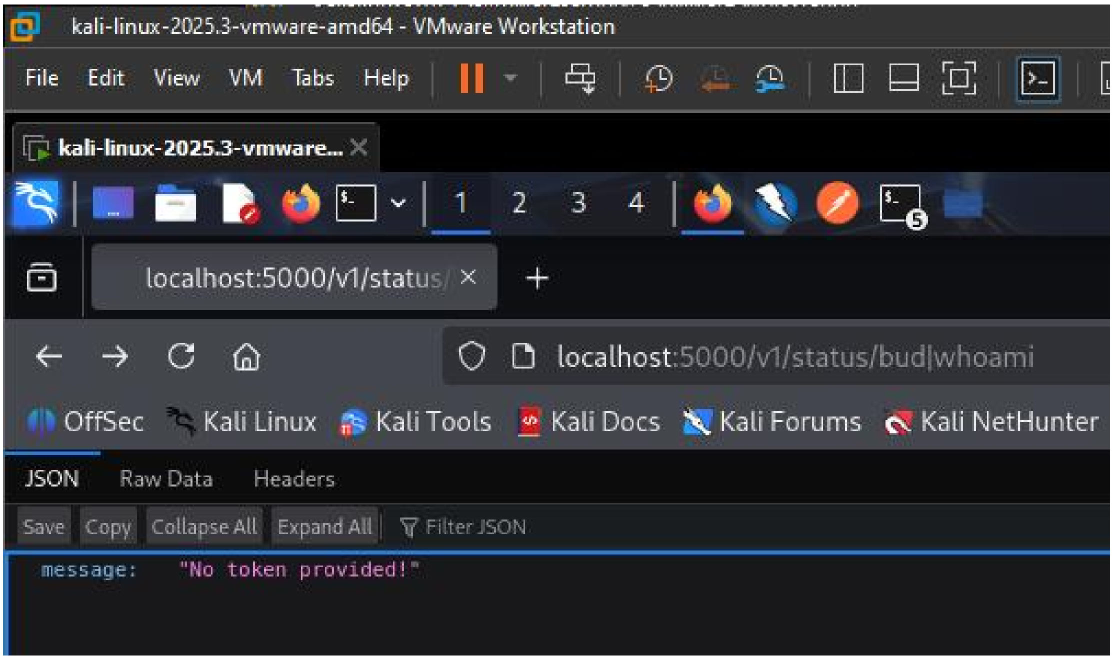
Path traversal



localhost:5000/v1/beer-pic

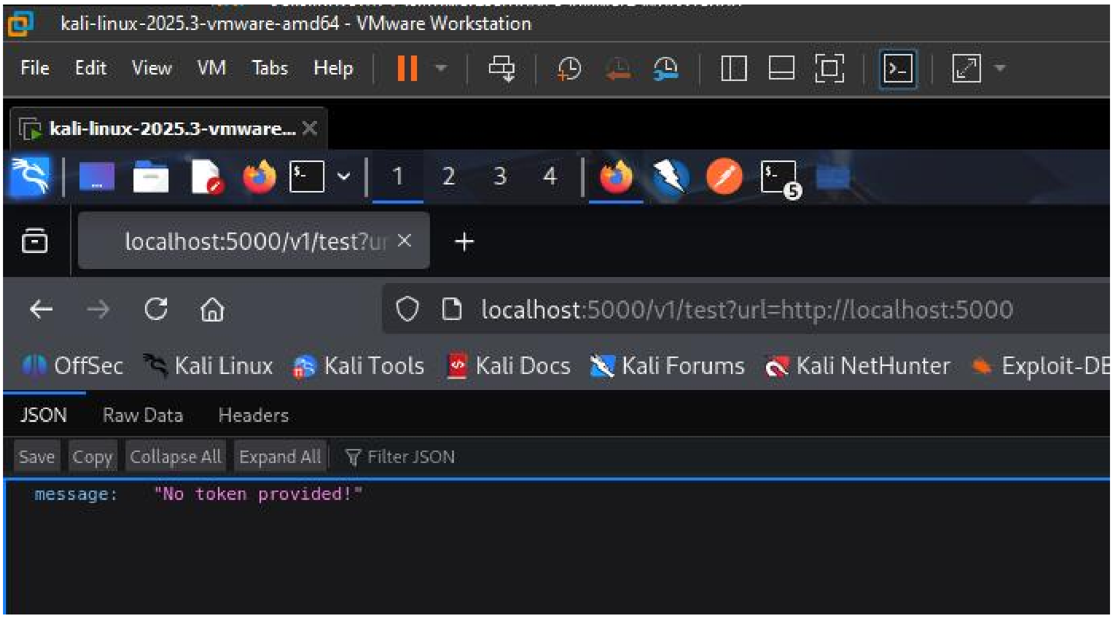
Invalid file name

V5RCE


kali-linux-2025.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
kali-linux-2025.3-vmware...
1 2 3 4
localhost:5000/v1/status/
localhost:5000/v1/status/bud|whoami
OffSec Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter
JSON Raw Data Headers
Save Copy Collapse All Expand All Filter JSON

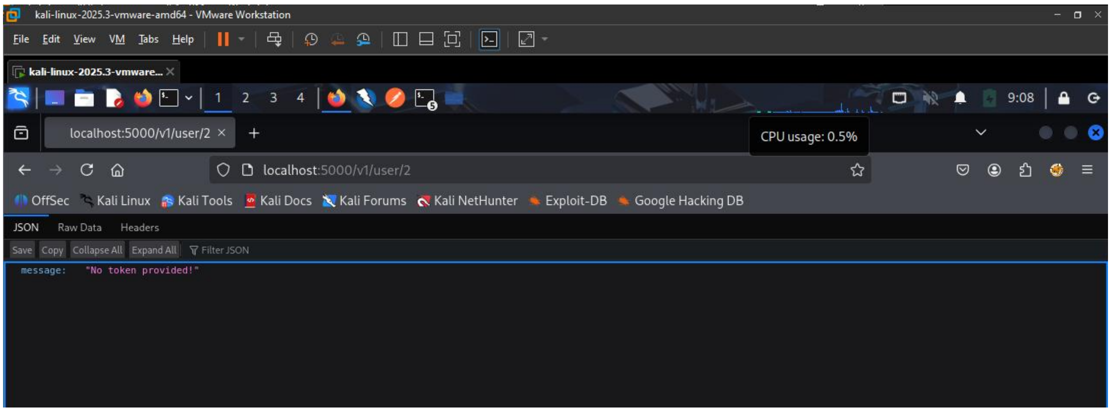
```
message:    "No token provided!"
```

V6 SSRF


kali-linux-2025.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
kali-linux-2025.3-vmware...
1 2 3 4
localhost:5000/v1/test?ur
localhost:5000/v1/test?url=http://localhost:5000
OffSec Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB
JSON Raw Data Headers
Save Copy Collapse All Expand All Filter JSON

```
message:    "No token provided!"
```

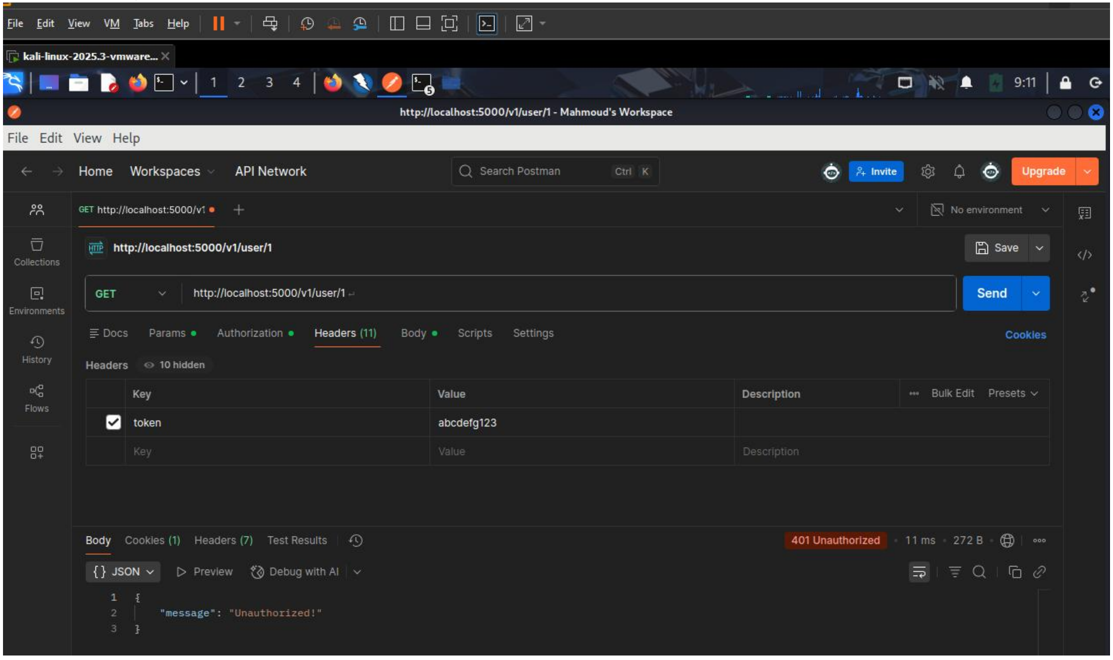
V7 Broken Authentication


kali-linux-2025.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
kali-linux-2025.3-vmware...
9:08
CPU usage: 0.5%
OffSec Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB
JSON Raw Data Headers
Save Copy Collapse All Expand All Filter JSON

localhost:5000/v1/user/2

```
message:   "No token provided!"
```

V8 insecure jwt


File Edit View VM Tabs Help
kali-linux-2025.3-vmware...
9:11
http://localhost:5000/v1/user/1 - Mahmoud's Workspace
File Edit View Help
Home Workspaces API Network
Search Postman Ctrl K
Invite Upgrade
Collections
Environments
History
Flows
No environment
Save
Send
Docs Params Authorization Headers (11) Body Scripts Settings
Cookies
Headers 10 hidden
Bulk Edit Presets
Body Cookies (1) Headers (7) Test Results
{ } JSON Preview Debug with AI

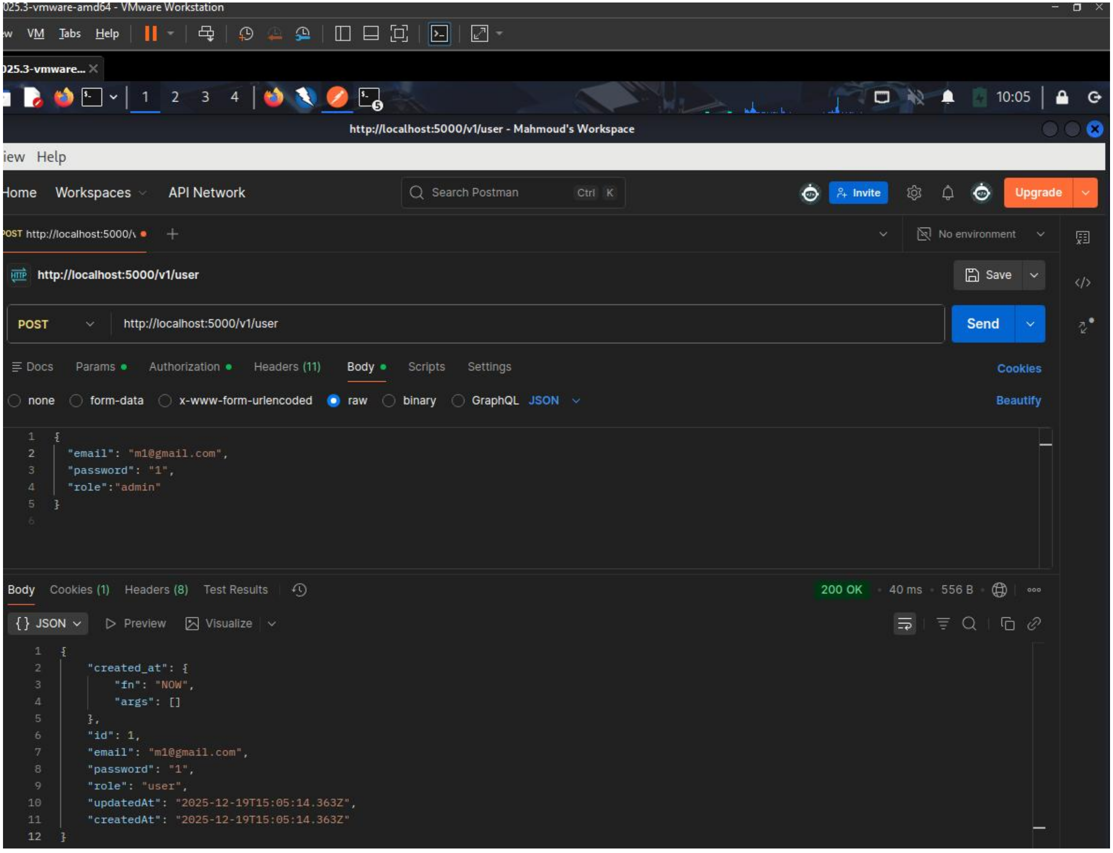
GET http://localhost:5000/v1/user/1

	Key	Value	Description
☑	token	abcdefg123	
	Key	Value	Description

401 Unauthorized · 11 ms · 272 B

```
{
    "message": "Unauthorized!"
}
```

V9 Insecure Design



POST http://localhost:5000/v1/user

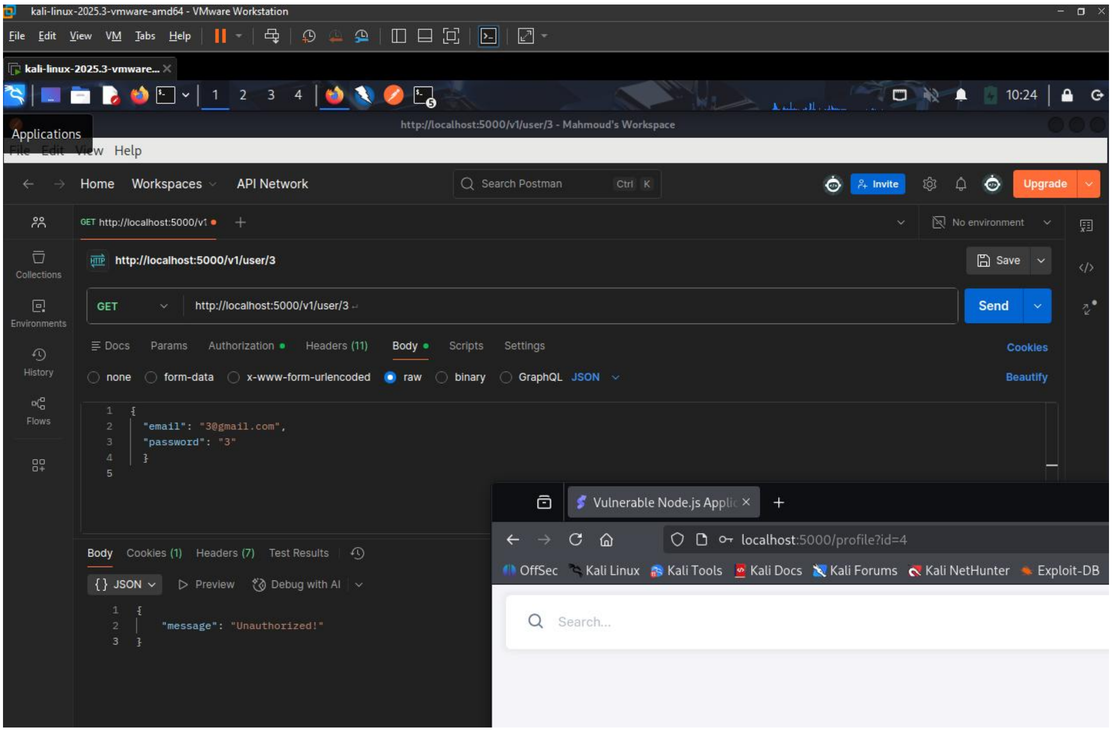
○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
  "email": "m1@gmail.com",
  "password": "1",
  "role":"admin"
}
```

Body 200 OK · 40 ms · 556 B

```
{
    "created_at": {
        "fn": "NOW",
        "args": []
    },
    "id": 1,
    "email": "m1@gmail.com",
    "password": "1",
    "role": "user",
    "updatedAt": "2025-12-19T15:05:14.363Z",
    "createdAt": "2025-12-19T15:05:14.363Z"
}
```

V10 idor



GET http://localhost:5000/v1/user/3

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

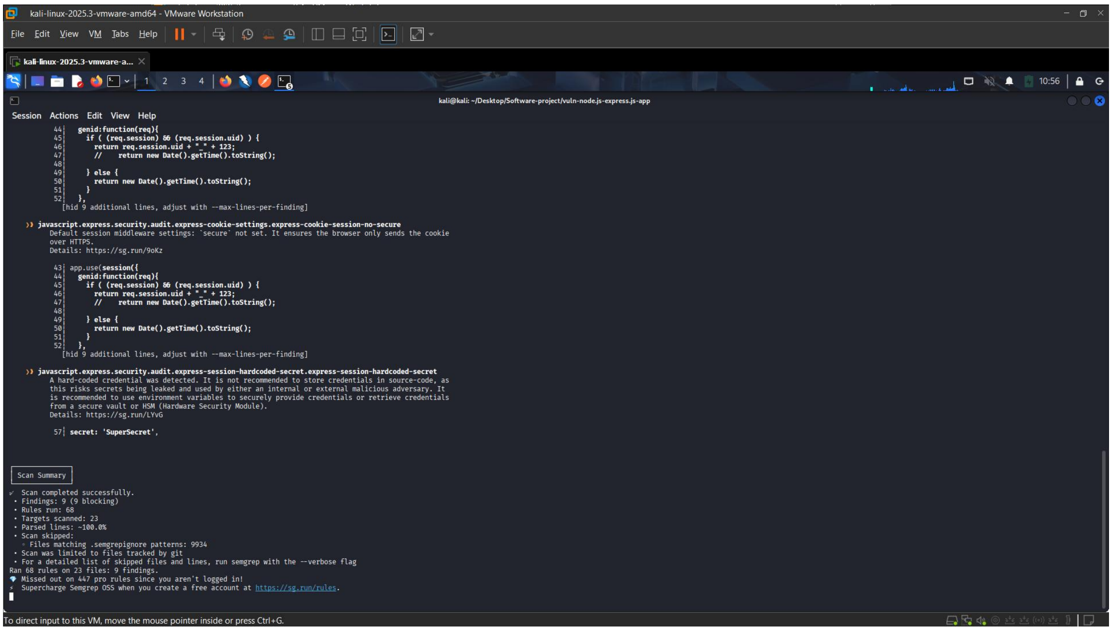
```
{
  "email": "3@gmail.com",
  "password": "3"
}
```

Body

```
{
    "message": "Unauthorized!"
}
```

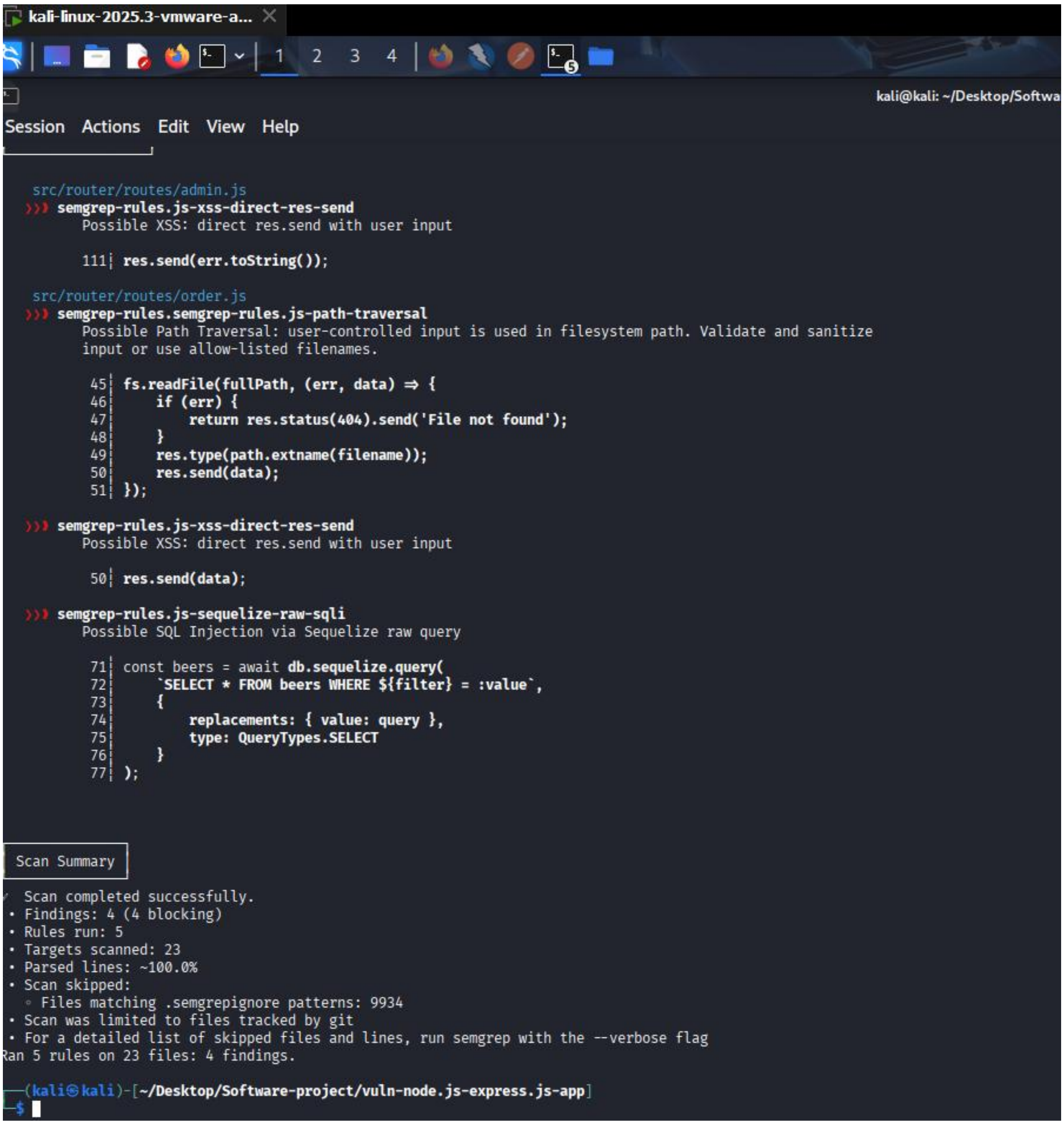
localhost:5000/profile?id=4

C3


kali-linux-2025.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
kali-linux-2025.3-vmware-a...
1 2 3 4
10:56
kali@kali: ~/Desktop/Software-project/vuln-node.js-express.js-app
Session Actions Edit View Help

```
        44|   genid:function(req){
        45|     if ( (req.session) && (req.session.uid) ) {
        46|       return req.session.uid + "_" + 123;
        47|       //    return new Date().getTime().toString();
        48|
        49|     } else {
        50|       return new Date().getTime().toString();
        51|     }
        52|   },
          [hid 9 additional lines, adjust with --max-lines-per-finding]

   ❯❯ javascript.express.security.audit.express-cookie-settings.express-cookie-session-no-secure
        Default session middleware settings: `secure` not set. It ensures the browser only sends the cookie
        over HTTPS.
        Details: https://sg.run/9oKz

        43| app.use(session({
        44|   genid:function(req){
        45|     if ( (req.session) && (req.session.uid) ) {
        46|       return req.session.uid + "_" + 123;
        47|       //    return new Date().getTime().toString();
        48|
        49|     } else {
        50|       return new Date().getTime().toString();
        51|     }
        52|   },
          [hid 9 additional lines, adjust with --max-lines-per-finding]

   ❯❯ javascript.express.security.audit.express-session-hardcoded-secret.express-session-hardcoded-secret
        A hard-coded credential was detected. It is not recommended to store credentials in source-code, as
        this risks secrets being leaked and used by either an internal or external malicious adversary. It
        is recommended to use environment variables to securely provide credentials or retrieve credentials
        from a secure vault or HSM (Hardware Security Module).
        Details: https://sg.run/LYvG

        57| secret: 'SuperSecret',


┌──────────────┐
│ Scan Summary │
└──────────────┘
✓ Scan completed successfully.
 • Findings: 9 (9 blocking)
 • Rules run: 68
 • Targets scanned: 23
 • Parsed lines: ~100.0%
 • Scan skipped:
   ◦ Files matching .semgrepignore patterns: 9934
 • Scan was limited to files tracked by git
 • For a detailed list of skipped files and lines, run semgrep with the --verbose flag
Ran 68 rules on 23 files: 9 findings.
💎 Missed out on 447 pro rules since you aren't logged in!
⚡ Supercharge Semgrep OSS when you create a free account at https://sg.run/rules.
█
```

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.



```
  src/router/routes/admin.js
 ❯❯❯ semgrep-rules.js-xss-direct-res-send
        Possible XSS: direct res.send with user input

        111┆ res.send(err.toString());

  src/router/routes/order.js
 ❯❯❯ semgrep-rules.semgrep-rules.js-path-traversal
        Possible Path Traversal: user-controlled input is used in filesystem path. Validate and sanitize
        input or use allow-listed filenames.

         45┆ fs.readFile(fullPath, (err, data) ⇒ {
         46┆     if (err) {
         47┆         return res.status(404).send('File not found');
         48┆     }
         49┆     res.type(path.extname(filename));
         50┆     res.send(data);
         51┆ });

 ❯❯❯ semgrep-rules.js-xss-direct-res-send
        Possible XSS: direct res.send with user input

         50┆ res.send(data);

 ❯❯❯ semgrep-rules.js-sequelize-raw-sqli
        Possible SQL Injection via Sequelize raw query

         71┆ const beers = await db.sequelize.query(
         72┆     `SELECT * FROM beers WHERE ${filter} = :value`,
         73┆     {
         74┆         replacements: { value: query },
         75┆         type: QueryTypes.SELECT
         76┆     }
         77┆ );


 Scan Summary

✓ Scan completed successfully.
 • Findings: 4 (4 blocking)
 • Rules run: 5
 • Targets scanned: 23
 • Parsed lines: ~100.0%
 • Scan skipped:
   ◦ Files matching .semgrepignore patterns: 9934
 • Scan was limited to files tracked by git
 • For a detailed list of skipped files and lines, run semgrep with the --verbose flag
Ran 5 rules on 23 files: 4 findings.

┌──(kali㉿kali)-[~/Desktop/Software-project/vuln-node.js-express.js-app]
└─$ 
```