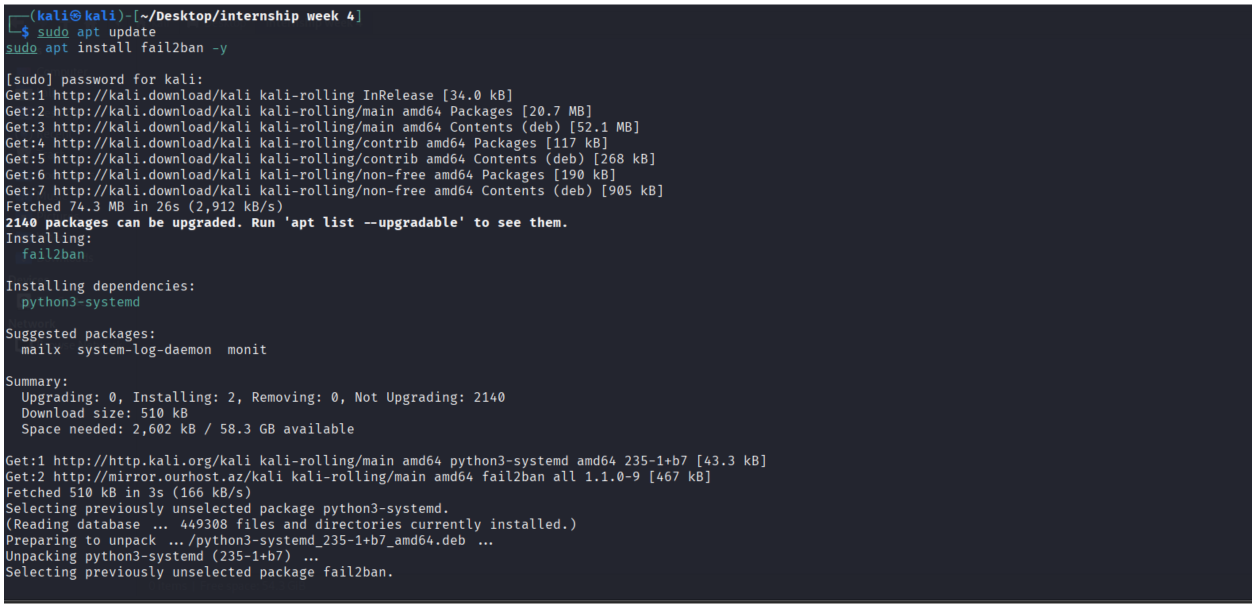
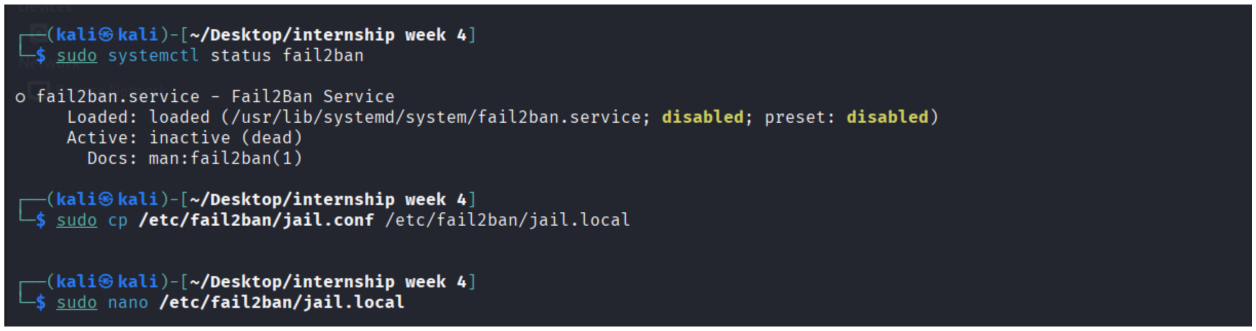
Week 4: Advanced Threat Detection & Web Security Enhancements

1. Intrusion Detection & Monitoring

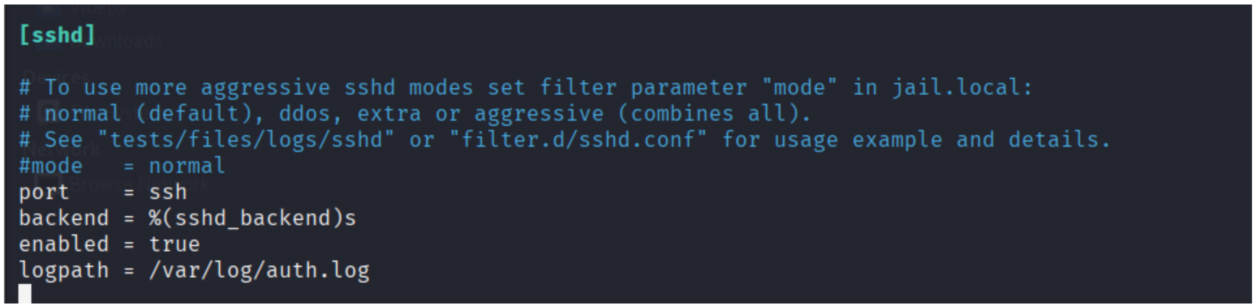
I configured Fail2Ban as a host-based intrusion prevention system. It monitors authentication logs in real time and detects repeated failed SSH login attempts. When the retry threshold is exceeded, Fail2Ban automatically bans the attacking IP using firewall rules, effectively mitigating brute-force attacks.



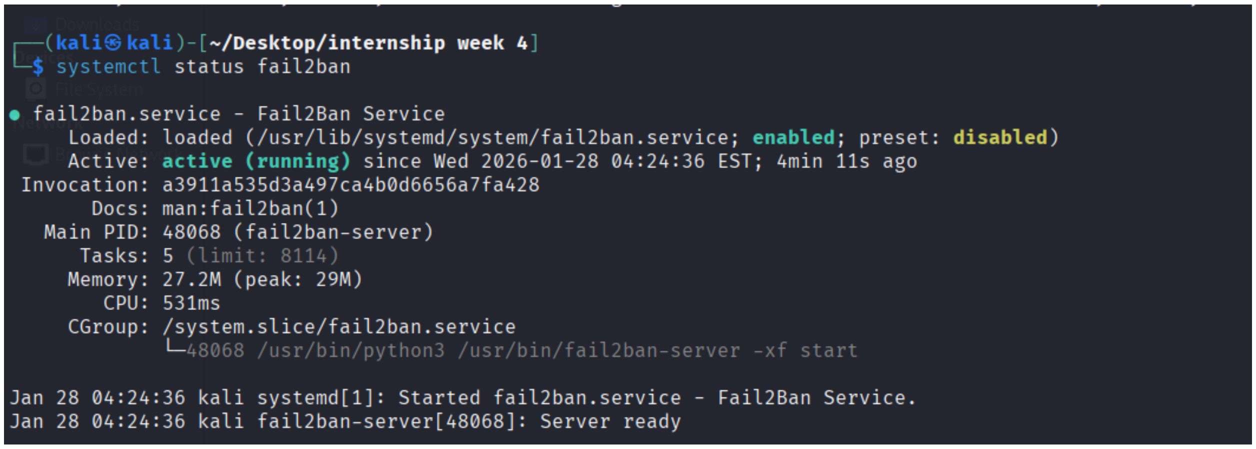
```
┌──(kali㉿kali)-[~/Desktop/internship week 4]
└─$ sudo apt update
sudo apt install fail2ban -y

[sudo] password for kali:
Get:1 http://kali.download/kali kali-rolling InRelease [34.0 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 Packages [20.7 MB]
Get:3 http://kali.download/kali kali-rolling/main amd64 Contents (deb) [52.1 MB]
Get:4 http://kali.download/kali kali-rolling/contrib amd64 Packages [117 kB]
Get:5 http://kali.download/kali kali-rolling/contrib amd64 Contents (deb) [268 kB]
Get:6 http://kali.download/kali kali-rolling/non-free amd64 Packages [190 kB]
Get:7 http://kali.download/kali kali-rolling/non-free amd64 Contents (deb) [905 kB]
Fetched 74.3 MB in 26s (2,912 kB/s)
2140 packages can be upgraded. Run 'apt list --upgradable' to see them.
Installing:
  fail2ban

Installing dependencies:
  python3-systemd

Suggested packages:
  mailx  system-log-daemon  monit

Summary:
  Upgrading: 0, Installing: 2, Removing: 0, Not Upgrading: 2140
  Download size: 510 kB
  Space needed: 2,602 kB / 58.3 GB available

Get:1 http://http.kali.org/kali kali-rolling/main amd64 python3-systemd amd64 235-1+b7 [43.3 kB]
Get:2 http://mirror.ourhost.az/kali kali-rolling/main amd64 fail2ban all 1.1.0-9 [467 kB]
Fetched 510 kB in 3s (166 kB/s)
Selecting previously unselected package python3-systemd.
(Reading database ... 449308 files and directories currently installed.)
Preparing to unpack .../python3-systemd_235-1+b7_amd64.deb ...
Unpacking python3-systemd (235-1+b7) ...
Selecting previously unselected package fail2ban.
```



```
┌──(kali㉿kali)-[~/Desktop/internship week 4]
└─$ sudo systemctl status fail2ban

○ fail2ban.service - Fail2Ban Service
     Loaded: loaded (/usr/lib/systemd/system/fail2ban.service; disabled; preset: disabled)
     Active: inactive (dead)
       Docs: man:fail2ban(1)

┌──(kali㉿kali)-[~/Desktop/internship week 4]
└─$ sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local

┌──(kali㉿kali)-[~/Desktop/internship week 4]
└─$ sudo nano /etc/fail2ban/jail.local
```

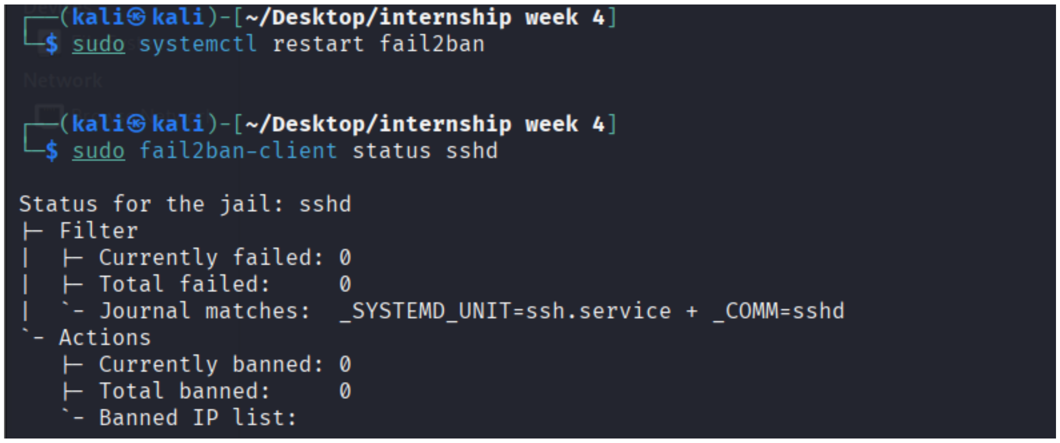


```
[sshd]

# To use more aggressive sshd modes set filter parameter "mode" in jail.local:
# normal (default), ddos, extra or aggressive (combines all).
# See "tests/files/logs/sshd" or "filter.d/sshd.conf" for usage example and details.
#mode   = normal
port    = ssh
backend = %(sshd_backend)s
enabled = true
logpath = /var/log/auth.log
```



```
┌──(kali㉿kali)-[~/Desktop/internship week 4]
└─$ systemctl status fail2ban

● fail2ban.service - Fail2Ban Service
     Loaded: loaded (/usr/lib/systemd/system/fail2ban.service; enabled; preset: disabled)
     Active: active (running) since Wed 2026-01-28 04:24:36 EST; 4min 11s ago
 Invocation: a3911a535d3a497ca4b0d6656a7fa428
       Docs: man:fail2ban(1)
   Main PID: 48068 (fail2ban-server)
      Tasks: 5 (limit: 8114)
     Memory: 27.2M (peak: 29M)
        CPU: 531ms
     CGroup: /system.slice/fail2ban.service
             └─48068 /usr/bin/python3 /usr/bin/fail2ban-server -xf start

Jan 28 04:24:36 kali systemd[1]: Started fail2ban.service - Fail2Ban Service.
Jan 28 04:24:36 kali fail2ban-server[48068]: Server ready
```

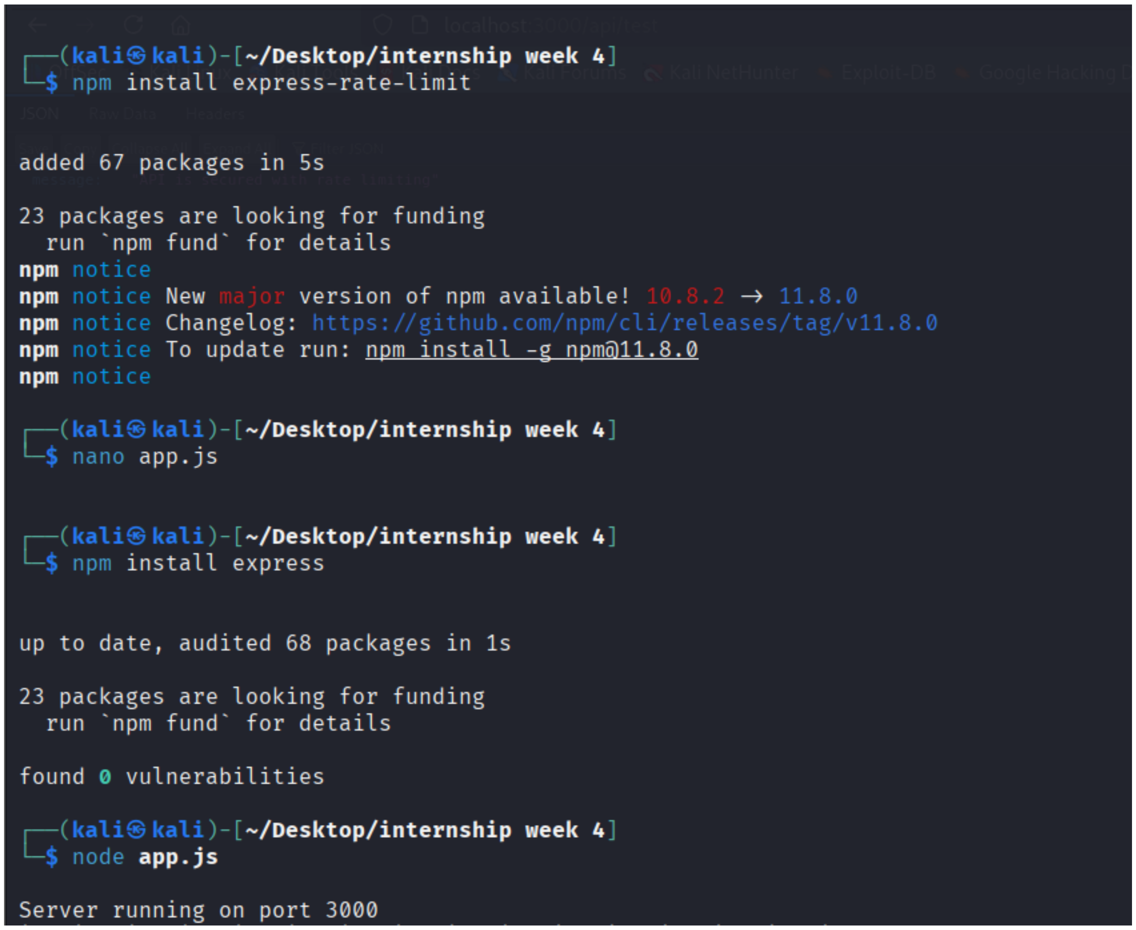
No IPs are banned yet because no brute-force attempts occurred. Fail2Ban works reactively and bans IPs only after detecting multiple failures.



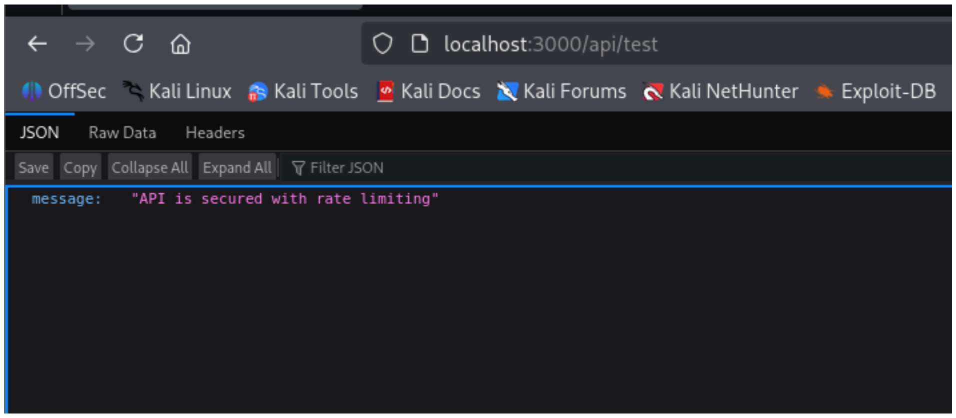
```
┌──(kali㉿kali)-[~/Desktop/internship week 4]
└─$ sudo systemctl restart fail2ban

┌──(kali㉿kali)-[~/Desktop/internship week 4]
└─$ sudo fail2ban-client status sshd

Status for the jail: sshd
|- Filter
|  |- Currently failed: 0
|  |- Total failed:     0
|  `- Journal matches:  _SYSTEMD_UNIT=ssh.service + _COMM=sshd
`- Actions
   |- Currently banned: 0
   |- Total banned:     0
   `- Banned IP list:
```

2. API Security Hardening

● Apply rate limiting using express-rate-limit to prevent brute-force attacks.



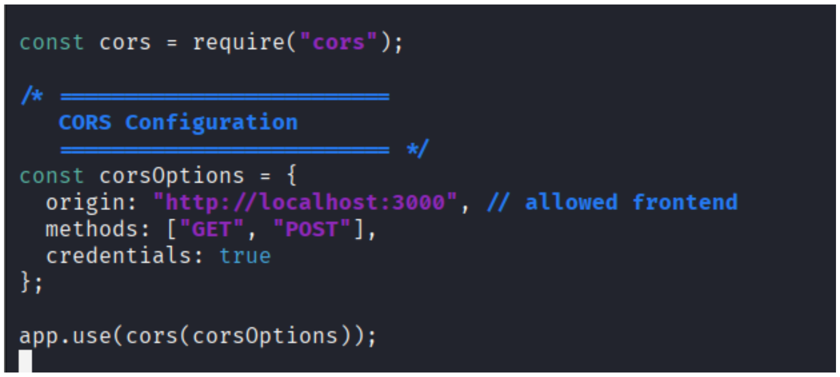
```
┌──(kali㉿kali)-[~/Desktop/internship week 4]
└─$ npm install express-rate-limit

added 67 packages in 5s

23 packages are looking for funding
  run `npm fund` for details
npm notice
npm notice New major version of npm available! 10.8.2 -> 11.8.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.8.0
npm notice To update run: npm install -g npm@11.8.0
npm notice

┌──(kali㉿kali)-[~/Desktop/internship week 4]
└─$ nano app.js


┌──(kali㉿kali)-[~/Desktop/internship week 4]
└─$ npm install express


up to date, audited 68 packages in 1s

23 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

┌──(kali㉿kali)-[~/Desktop/internship week 4]
└─$ node app.js

Server running on port 3000
```



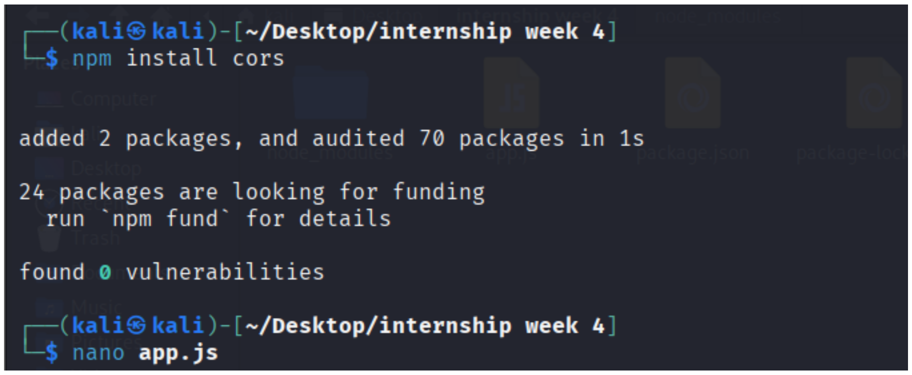
```
message:    "API is secured with rate limiting"
```

●Properly configure CORS to restrict unauthorized access.

CORS was configured to allow requests only from trusted origins. This prevents unauthorized domains from accessing the API and mitigates cross-origin attacks

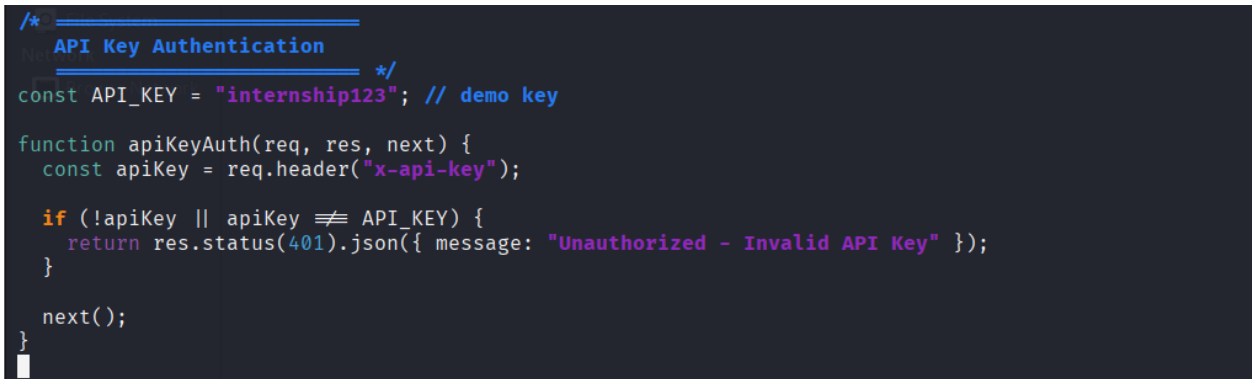


```
const cors = require("cors");

/* ========================
   CORS Configuration
   ======================== */
const corsOptions = {
  origin: "http://localhost:3000", // allowed frontend
  methods: ["GET", "POST"],
  credentials: true
};

app.use(cors(corsOptions));
```

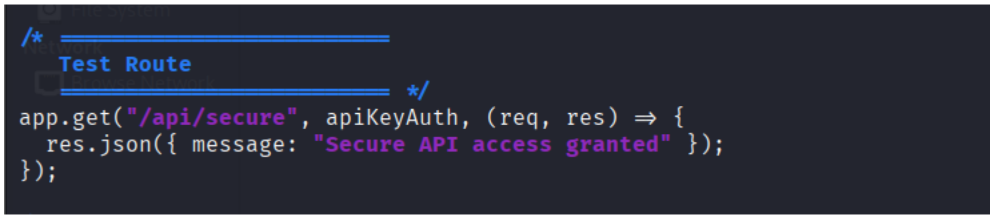


```
┌──(kali㉿kali)-[~/Desktop/internship week 4]
└─$ npm install cors

added 2 packages, and audited 70 packages in 1s

24 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

┌──(kali㉿kali)-[~/Desktop/internship week 4]
└─$ nano app.js
```

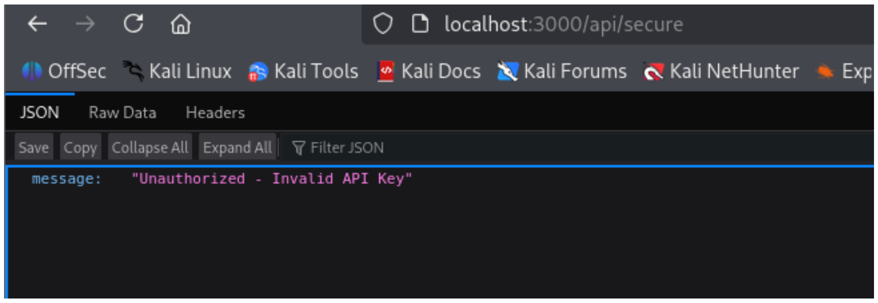
●Secure APIs using API keys or OAuth authentication.



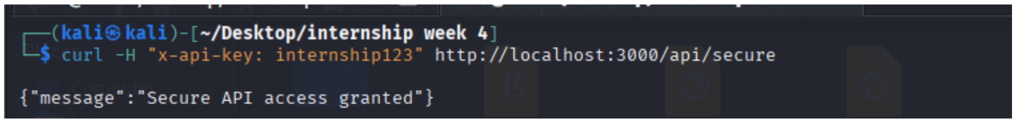
```
/* ========================
   API Key Authentication
   ======================== */
const API_KEY = "internship123"; // demo key

function apiKeyAuth(req, res, next) {
  const apiKey = req.header("x-api-key");

  if (!apiKey || apiKey !== API_KEY) {
    return res.status(401).json({ message: "Unauthorized - Invalid API Key" });
  }

  next();
}
```



```
/* ========================
   Test Route
   ======================== */
app.get("/api/secure", apiKeyAuth, (req, res) => {
  res.json({ message: "Secure API access granted" });
});
```



```
message:   "Unauthorized - Invalid API Key"
```

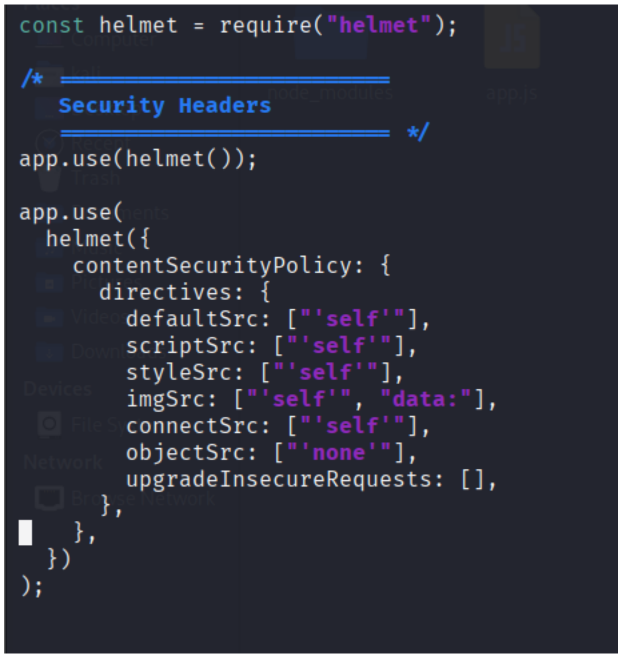


```
┌──(kali㉿kali)-[~/Desktop/internship week 4]
└─$ curl -H "x-api-key: internship123" http://localhost:3000/api/secure

{"message":"Secure API access granted"}
```

3. Security Headers & CSP Implementation

Security headers protect the application at the browser level by preventing common client-side attacks such as XSS, clickjacking, and protocol downgrade attacks.

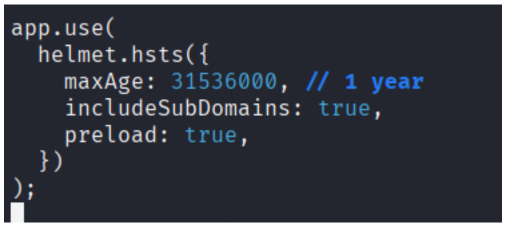
- Implement Content Security Policy (CSP) to prevent script injections.



```
const helmet = require("helmet");

/* ══════════════════════════
   Security Headers
   ══════════════════════════ */
app.use(helmet());

app.use(
  helmet({
    contentSecurityPolicy: {
      directives: {
        defaultSrc: ["'self'"],
        scriptSrc: ["'self'"],
        styleSrc: ["'self'"],
        imgSrc: ["'self'", "data:"],
        connectSrc: ["'self'"],
        objectSrc: ["'none'"],
        upgradeInsecureRequests: [],
      },
    },
  })
);
```

I implemented a strict Content Security Policy using Helmet to mitigate cross-site scripting attacks by restricting the sources from which scripts and other resources can be loaded.

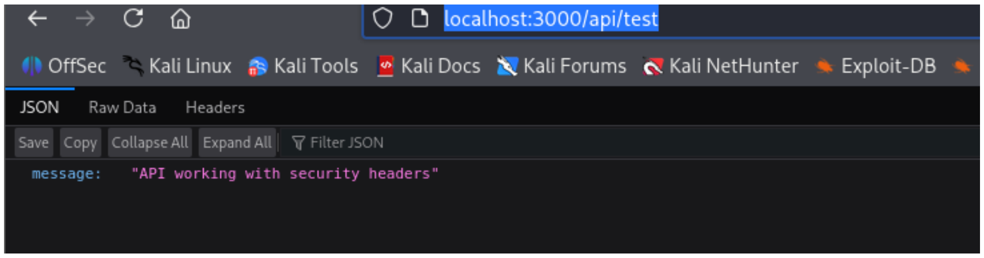
- Enforce HTTPS using Strict-Transport-Security (HSTS) headers.



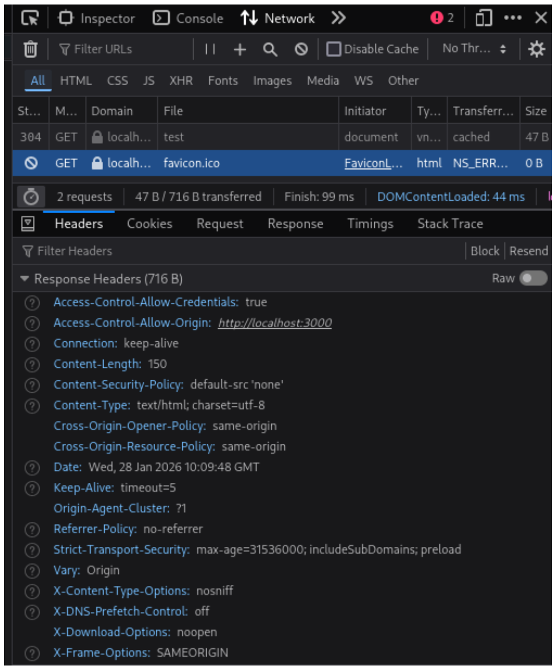
```
app.use(
  helmet.hsts({
    maxAge: 31536000, // 1 year
    includeSubDomains: true,
    preload: true,
  })
);
```

I enforced HTTPS using the Strict-Transport-Security header, which ensures that browsers communicate with the server only over secure HTTPS connections and prevents downgrade attacks.

HSTS is ignored on HTTP and localhost environments; it becomes effective when deployed over HTTPS.



```
message:   "API working with security headers"
```



St…	M…	Domain	File	Initiator	Ty…	Transferr…	Size
304	GET	localh…	test	document	vn…	cached	47 B
	GET	localh…	favicon.ico	FaviconL…	html	NS_ERR…	0 B

```
2 requests | 47 B / 716 B transferred | Finish: 99 ms | DOMContentLoaded: 44 ms

Response Headers (716 B)
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: http://localhost:3000
Connection: keep-alive
Content-Length: 150
Content-Security-Policy: default-src 'none'
Content-Type: text/html; charset=utf-8
Cross-Origin-Opener-Policy: same-origin
Cross-Origin-Resource-Policy: same-origin
Date: Wed, 28 Jan 2026 10:09:48 GMT
Keep-Alive: timeout=5
Origin-Agent-Cluster: ?1
Referrer-Policy: no-referrer
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
Vary: Origin
X-Content-Type-Options: nosniff
X-DNS-Prefetch-Control: off
X-Download-Options: noopen
X-Frame-Options: SAMEORIGIN
```