DELSTRIKE
SECURITY

CHECKPOINT Secure code review

TEST · Training repository

OWASP NODEGOAT

Target	`https://github.com/OWASP/NodeGoat`
Commit	`c5cb68a7084e · master`
Scan date	**2026-11-16 · Baseline**
Security status	**Very weak**
Open findings	**20 · 2 critical, 7 high, 11 medium**
Generated	**2026-11-16**

1. EXECUTIVE SUMMARY

Baseline. This is the first scan of this code, so there is no comparison. Later scans will mark each finding as new, persistent or fixed.

Tools unavailable in this scan: osv-scanner.

SECURITY STATUS

VERY WEAK

There is at least one exploitable critical vulnerability: the code should not run in production until it is fixed.

GO-LIVE GATE

KO

Blocked while critical/high findings remain open.

Critical · 2 | High · 7 | Medium · 11 | Low · 0 | Info · 0

This report is a demonstration of the CHECKPOINT service on OWASP NodeGoat, a public training application that simulates the employee portal of a retirement plan provider and was built with deliberate flaws. All of its code was reviewed, 77 files and about 2,600 lines of JavaScript, without running the application.

The security status is VERY WEAK and the release gate is KO. There are 20 open findings: 2 critical, 7 high and 11 medium. The automated tools raised 61 leads: after manual review, 29 were confirmed and 32 were dismissed as false positives or code with no risk.

Each of the two critical findings is enough to take over the server or impersonate any user: the contributions form runs whatever the user types as code, and the secrets that sign sessions are in the code itself. The high findings let any registered employee see other customers' portfolios, change HR data reserved for administrators, use the server to read internal systems or take it down with a single request.

The good news is that almost every fix is small and well known. The framework already ships the defences, and in this code they are installed but switched off. With the six priorities in this report, the application would go from «very weak» to a level ready for a new review.

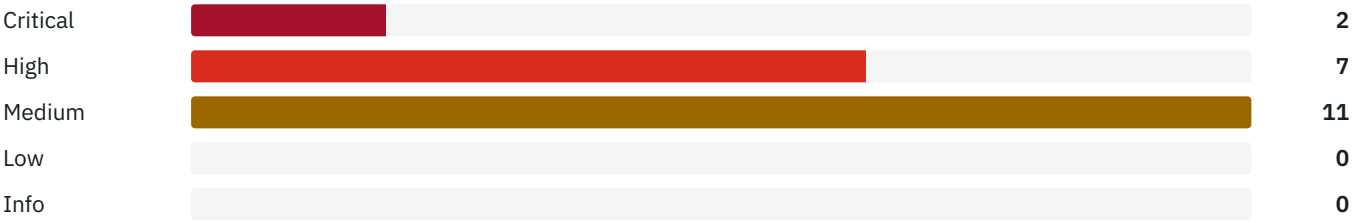
20	77	20	—	1
Open findings	Code files 2,573 LOC	Routes found 0 sensitive routes without control	Vulnerable packages unavailable	Exposed secrets

61
Automated leads
29 confirmed · 32 discarded

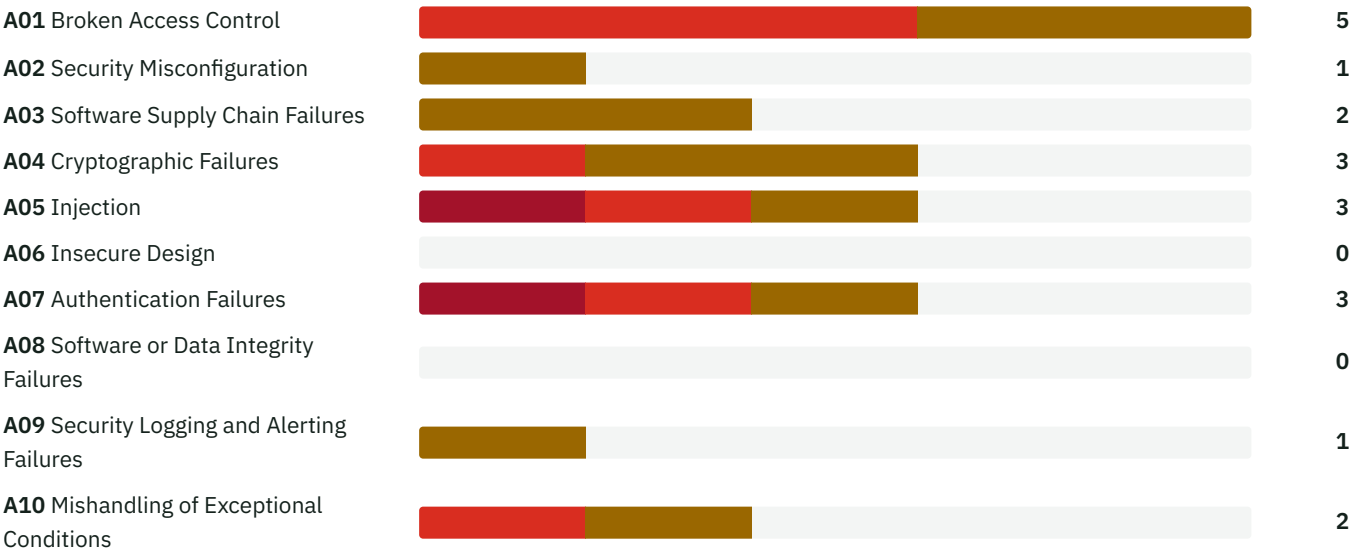
Main risks

- Server takeover: a registered user can run their own code on the server and, from there, reach the whole database and the secrets (CHK-ccd14d).
- Impersonation of any account: the secrets in the repository allow valid sessions to be forged, and the session is not renewed at login (CHK-67ab09, CHK-40e9d9).
- Leak of every customer's personal and financial data: portfolios visible by changing a number in the URL, a query that can be manipulated, and bank and social security data stored unencrypted, with a duty to notify under the GDPR (CHK-1f4cce, CHK-81bd08, CHK-f909bf).
- Abuse of administration functions: any employee can change other people's benefit dates (CHK-410234).
- Service outage: a single request with a crafted value blocks the server for minutes (CHK-847dcb, CHK-81bd08).
- Credential theft: passwords stored in clear, an unencrypted server, no attempt limit and a redirect that can be used for phishing (CHK-864901, CHK-156a89, CHK-92db72, CHK-0edfaa).

Open findings by severity



Open findings by OWASP Top 10:2025 category



2. PRIORITIES

Recommended actions, in order.

1 Remove code execution and the injectable query

Replace eval() with numeric conversion and range checks, and drop the MongoDB $where operator. These are few-line changes that close the takeover path and the mass portfolio leak. Check: a search for eval and $where returns nothing and a no-eval rule runs in CI.

Findings: `CHK-ccd14d` Critical · `CHK-81bd08` High

2 Rotate every secret and take it out of the code

Generate new session and encryption secrets, load them from environment variables or a secrets manager, remove the TLS key from the repository and clean the history. Restrict the seed script to test environments. Add Gitleaks to CI.

Findings: `CHK-67ab09` Critical

3 Enforce access control on the server

Turn on the administrator role check on /benefits and always take the user from the session, never from the URL, on /allocations. Add automated tests that check a user without permission gets 403.

Findings: `CHK-410234` High · `CHK-1f4cce` High

4 Harden authentication and sessions

argon2id hashing for passwords, session regeneration on every login, a cookie with HttpOnly, Secure and SameSite, an attempt limit, a single error message and a 12-character policy. Enable CSRF protection.

Findings: `CHK-864901` Medium · `CHK-40e9d9` High · `CHK-92db72` Medium · `CHK-e4a611` Medium · `CHK-5b2f23` Medium

5 Close SSRF, ReDoS and XSS, and serve HTTPS only

Fix the quotes provider on the server, replace the routing number regular expression with a linear anchored one, turn on template auto-escaping with a CSP, and serve over HTTPS with HSTS. Restrict the /learn redirect to an allow-list.

Findings: `CHK-e08a94` High · `CHK-847dcb` High · `CHK-3993f8` Medium · `CHK-156a89` High · `CHK-0edfaa` Medium

6 **Protect the data and the supply chain**

Encrypt the social security and bank fields, replace the abandoned dependencies (swig, bcrypt-nodejs, node-esapi), pin CI actions by SHA, show generic errors and log authentication events with a structured logger.

Findings: `CHK-f909bf` Medium · `CHK-d06a9e` Medium · `CHK-9badba` Medium · `CHK-af97d6` Medium · `CHK-6ea3ab` Medium

Conclusion

NodeGoat does its job as training material: it packs the most common flaws of a real Node.js application into a small codebase. As a production system it would not be acceptable. The most useful lesson for any team is the underlying pattern: the defences were installed and then switched off or never wired in. That is why manual review is essential: the tools spotted the symptoms, but access control, sessions and missing encryption only show up when someone reads the code. Once the priorities are applied, CHECKPOINT would run the analysis again to mark each finding as fixed and confirm the new status.

3. SCOPE AND METHODOLOGY

White-box source code review: automated static analysis plus manual data-flow review. The application was not executed and no requests were sent to any server.

Code analysed	`https://github.com/OWASP/NodeGoat @ c5cb68a7084e (master)`
Languages	JavaScript
Excluded from analysis	Installed dependencies, build output and version control (node_modules, vendor, dist, build, .git...)
Manual review	Static review of the public OWASP NodeGoat repository, a Node.js/Express application with MongoDB built on purpose with OWASP Top 10 flaws for training. It was used as the demonstration project for the CHECKPOINT service. The startup code (server.js), configuration (config/), router, every page handler (session, profile, contributions, allocations, benefits, memos, research, errors), data access layers and views with template variables were read by hand. For each tool candidate, the data was traced from the HTTP request to the point where it is used (query, eval, redirect, outbound request, HTML output). The review also covered what tools cannot see: function- and object-level authorization, sessions, CSRF, password and personal data storage, error handling, logging and CI. The application was not run and its dependencies were not installed. OSV-Scanner could not run on the analysis machine, so dependencies were assessed by version and maintenance status, not against a vulnerability database.
Files reviewed manually	`server.js, config/config.js, config/env/all.js, package.json, app/routes/index.js, app/routes/session.js, app/routes/profile.js, app/routes/contributions.js, app/routes/allocations.js, app/routes/benefits.js, app/routes/memos.js, app/routes/research.js, app/routes/tutorial.js, app/routes/error.js, app/data/user-dao.js, app/data/profile-dao.js, app/data/allocations-dao.js, app/views/profile.html, app/views/memos.html, app/views/error-template.html, artifacts/db-reset.js, .github/workflows/e2e-test.yml, .github/workflows/lint.yml`

Tools

Tool	Purpose	Version	Status	Results
checkpoint-rules	DELSTRIKE secure-coding rules (OWASP, PHP, Java, .NET, JS, Python)	`1.1.0`	**OK**	25
checkpoint-stack	Supabase (RLS and policies), Firebase, service_role keys and deployment headers	`1.1.0`	**OK**	0
semgrep	Multi-language static analysis (SAST)	`1.185.0`	**OK**	49
bandit	Python-specific SAST	`1.9.4`	Not applicable	-
gitleaks	Secrets and credentials in files	`8.30.1`	**OK**	3
osv-scanner	Dependencies with known vulnerabilities (OSV.dev)	-	**Unavailable**	-
checkpoint-registry	Non-existent, removed or newly published packages (npm, PyPI)	`1.1.0`	**OK**	0

Scope and methodology

1. Stack inventory: languages, frameworks, dependency manifests, configuration and risky files.

2. Map of public routes extracted from the code, flagging sensitive routes without access control.
3. Automated analysis with in-house rules and SAST, secrets and dependency tools.
4. Manual triage of every critical, high or medium lead: confirmed by reading the source → sink flow, or dismissed with a reason.
5. Additional manual review of authentication, authorisation, sessions, input validation and error handling.
6. CVSS 4.0 scoring, OWASP Top 10:2025 and CWE classification, mapped to ASVS 5.0, ISO/IEC 27001:2022 and NIS2.

Severity criteria

Severity is derived from each finding's CVSS 4.0 base score:

Severity	CVSS 4.0
Critical	9.0 – 10.0
High	7.0 – 8.9
Medium	4.0 – 6.9
Low	0.1 – 3.9
Info	0.0

Limitations

- Static analysis: the application is not executed, so issues that depend on deployed configuration, infrastructure or real data may not appear.
- Dependencies are assessed from the manifest and lock files present; versions installed in production may differ.
- Secrets are searched in the current state of the files; version history is not reviewed.
- A 'probable' finding needs dynamic confirmation before committing resources.

4. STACK AND ATTACK SURFACE

Languages

Language	Files	LOC
JSON	9	27651
HTML	24	2644
JavaScript	40	2573
YAML	4	117

Frameworks and libraries

Express · Helmet

Dependency manifests

File	Ecosystem
`package-lock.json`	npm
`package.json`	npm

Configuration files

`docker-compose.yml` · `Dockerfile` · `.travis.yml` · `.github/workflows/e2e-test.yml` · `.github/workflows/lint.yml`

Risky files under version control

File	Type
`artifacts/cert/server.key`	Private key

Public routes

20 Routes found

1 Sensitive

0 Sensitive routes without control

Sensitive routes without detected access control are listed first. Detection is static; the reviewer checked the flagged ones.

ID	Method	Route	Location	Access control	Sensitive
R-0001	GET	/dashboard	app/routes/index.js:44	**Protected**	•
R-0002	GET	/	app/routes/index.js:30	**Detected**	
R-0003	GET	/	app/routes/tutorial.js:8	Legitimately public	
R-0004	GET	/${page}	app/routes/tutorial.js:31	Legitimately public	
R-0005	GET	/allocations/:userId	app/routes/index.js:63	**Detected**	
R-0006	GET	/benefits	app/routes/index.js:55	**Detected**	
R-0007	POST	/benefits	app/routes/index.js:56	**Detected**	
R-0008	GET	/contributions	app/routes/index.js:51	**Detected**	
R-0009	POST	/contributions	app/routes/index.js:52	**Detected**	

ID	Method	Route	Location	Access control	Sensitive
R-0010	GET	/learn	app/routes/index.js:70	**Detected**	
R-0011	GET	/login	app/routes/index.js:33	Legitimately public	
R-0012	POST	/login	app/routes/index.js:34	Legitimately public	
R-0013	GET	/logout	app/routes/index.js:41	Legitimately public	
R-0014	GET	/memos	app/routes/index.js:66	**Detected**	
R-0015	POST	/memos	app/routes/index.js:67	**Detected**	
R-0016	GET	/profile	app/routes/index.js:47	**Detected**	
R-0017	POST	/profile	app/routes/index.js:48	**Detected**	
R-0018	GET	/research	app/routes/index.js:76	**Detected**	
R-0019	GET	/signup	app/routes/index.js:37	Legitimately public	
R-0020	POST	/signup	app/routes/index.js:38	Legitimately public	

5. FINDINGS OVERVIEW

ID	Finding	Severity	CVSS	OWASP	CWE	Status
CHK-ccd14d	Server-side code execution through eval() on contribution form data app/routes/contributions.js:32 (+5)	Critical	9.4	A05	CWE-95	New
CHK-67ab09	Session secret, encryption key, TLS key and default accounts in the repository config/env/all.js:8 (+4)	Critical	9.1	A07	CWE-798	New
CHK-40e9d9	Session is not renewed at login (session fixation) app/routes/session.js:116	High	7.4	A07	CWE-384	New
CHK-81bd08	NoSQL injection in the allocations filter through the $where operator app/data/allocations-dao.js:78	High	7.2	A05	CWE-943	New
CHK-410234	Benefits administration functions available to any registered user app/routes/index.js:55 (+1)	High	7.1	A01	CWE-285	New
CHK-1f4cce	Other customers' allocations readable by changing the identifier in the URL (IDOR) app/routes/allocations.js:16 (+1)	High	7.1	A01	CWE-639	New
CHK-e08a94	Outbound request to a user-chosen URL on the research page (SSRF) app/routes/research.js:15	High	7.1	A01	CWE-918	New
CHK-847dcb	Regular expression with exponential backtracking on the bank routing number (ReDoS) app/routes/profile.js:59	High	7.1	A10	CWE-1333	New
CHK-156a89	Server runs over HTTP only, with no encryption in transit server.js:144	High	7.1	A04	CWE-319	New
CHK-af97d6	Error page shows internal exception details to the user app/routes/error.js:10 (+1)	Medium	6.9	A10	CWE-209	New
CHK-5b2f23	No CSRF protection on forms that change data server.js:104 (+3)	Medium	6.9	A01	CWE-352	New
CHK-6ea3ab	Unsanitized usernames written to the application log app/routes/session.js:64	Medium	6.9	A09	CWE-117	New
CHK-92db72	Weak authentication: user enumeration, short passwords and no attempt limit app/routes/session.js:60 (+2)	Medium	6.9	A07	CWE-287	New
CHK-d06a9e	Abandoned dependencies or very old versions package.json:8 (+4)	Medium	6.3	A03	CWE-1104	New
CHK-9badba	GitHub Actions referenced by mutable tag .github/workflows/e2e-test.yml:16 (+11)	Medium	6.3	A03	CWE-829	New

ID	Finding	Severity	CVSS	OWASP	CWE	Status
CHK-f909bf	Personal and banking data stored unencrypted app/data/profile-dao.js:55	Medium	5.9	A04	CWE-311	New
CHK-864901	User passwords stored and compared in plain text app/data/user-dao.js:25 (+1)	Medium	5.9	A04	CWE-256	New
CHK-e4a611	Session cookie without security attributes and no HTTP protection headers server.js:78 (+2)	Medium	5.1	A02	CWE-1004	New
CHK-3993f8	Template auto-escaping disabled: stored XSS in profile and memos server.js:135 (+3)	Medium	5.1	A05	CWE-79	New
CHK-0edfaa	Open redirect in the learning resources link app/routes/index.js:72 (+1)	Medium	4.8	A01	CWE-601	New

6. TECHNICAL DETAILS

Server-side code execution through eval() on contribution form data

CHK-ccd14d **Critical** **9.4** **New** **Confirmed**

Category	Code injection (eval)	OWASP	A05:2025 · Injection
CWE	CWE-95	ASVS 5.0	V1, V15
Vector	`CVSS:4.0/AV:N/AC:L/AT:N/PR:L/UI:N/VC:H/VI:H/VA:H/SC:H/SI:H/SA:H`		
ISO 27001	8.28	NIS2	21.2.e
Detected by	checkpoint-rules, semgrep		

LOCATIONS

- **`app/routes/contributions.js:32`** — req.body.preTax
- **`app/routes/contributions.js:33`** — req.body.afterTax
- **`app/routes/contributions.js:34`** — req.body.roth
- **`app/routes/contributions.js:32`** — same eval (Semgrep)
- +2 more locations

DESCRIPTION

The POST /contributions handler turns the three contribution percentages into numbers by passing them to eval(). eval() does not convert anything: it runs whatever text arrives in the field as JavaScript.

RISK

Any registered user can make the server run their own code with the permissions of the Node.js process. That gives access to the whole database, the configuration and the secrets, and makes it possible to take the service down or use the server to attack other systems.

DETAILS

Source: req.body.preTax, afterTax and roth (contribution form, only requires a logged-in session). Sink: eval() in contributions.js:32-34, before any validation. The isNaN() check on line 47 comes too late: the code has already run. There is no allow-list or prior typing.

EVIDENCE

`app/routes/contributions.js:32 · checkpoint-rules · DS-JS-001`

```
29
30          /*jslint evil: true */
31          // Insecure use of eval() to parse inputs
32          const preTax = eval(req.body.preTax);
33          const afterTax = eval(req.body.afterTax);
34          const roth = eval(req.body.roth);
35
```

`app/routes/contributions.js:33 · checkpoint-rules · DS-JS-001`

```
30          /*jslint evil: true */
31          // Insecure use of eval() to parse inputs
32          const preTax = eval(req.body.preTax);
33          const afterTax = eval(req.body.afterTax);
34          const roth = eval(req.body.roth);
35
36          /*
```

RECOMMENDATION

Remove eval() and convert with Number.parseInt(value, 10) or Number(value). Then check that each value is an integer between 0 and 30 and reject the request otherwise. Add ESLint's no-eval rule in CI so it does not come back.

FIXED CODE (PROPOSAL) · JAVASCRIPT

```
const toPct = (v) => {
  const n = Number.parseInt(v, 10);
  return Number.isInteger(n) && n >= 0 && n <= 30 ? n : NaN;
};
const preTax = toPct(req.body.preTax);
const afterTax = toPct(req.body.afterTax);
const roth = toPct(req.body.roth);
```

REFERENCES

https://cwe.mitre.org/data/definitions/95.html
https://owasp.org/Top10/2025/A05_2025-Injection/
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/eval#never_use_direct_eval!

Session secret, encryption key, TLS key and default accounts in the repository

CHK-67ab09 Critical **9.1** New Confirmed

Category	Hard-coded credentials or secrets	OWASP	A07:2025 · Authentication Failures
CWE	CWE-798	ASVS 5.0	V13, V14
Vector	`CVSS:4.0/AV:N/AC:L/AT:P/PR:N/UI:N/VC:H/VI:H/VA:N/SC:N/SI:N/SA:N`		
ISO 27001	5.17, 8.4, 8.12	NIS2	21.2.i
Detected by	checkpoint-inventory, checkpoint-rules, Manual review		

LOCATIONS

- **`config/env/all.js:8`** — literal session cookie secret and encryption key
- **`artifacts/cert/server.key:1`** — TLS private key under version control
- **`artifacts/db-reset.js:18`** — seed administrator password
- **`artifacts/db-reset.js:27`** — seed user password
- +1 more locations

DESCRIPTION

The secret that signs session cookies and the encryption key are written in the code, the TLS certificate's private key is in the repository and the seed script creates an administrator with a fixed password.

RISK

Anyone with the code (here, anyone, because it is public) can forge valid session cookies, decrypt data and intercept TLS traffic. If the seed script is used in production, there is an administrator account with a known password.

DETAILS

config/env/all.js:8-9 defines the secrets as literals with no environment-variable override. artifacts/cert/server.key is not in .gitignore. artifacts/db-reset.js:18-35 inserts users with literal passwords, including one with isAdmin: true. Values are redacted in this report.

EVIDENCE

config/env/all.js:8

```
5  module.exports = {
6      port,
7      db,
8      cookieSecret: "[REDACTADO]",
9      cryptoKey: "[REDACTADO]",
10     cryptoAlgo: "aes256",
11     hostName: "localhost",
12     environmentalScripts: []
```

artifacts/cert/server.key:1 · checkpoint-inventory · DS-FILE-clave

```
1  [REDACTADO]
2  MIICXgIBAAKBgQCfn8uP4FuHaaAPrMkcl1fNMQM5EGMT4nnNSVoaEVdiDLc6P0mC
3  AZtUO9W0OjWow+TwGk3HkqoSJOA9KRMrzK7MtEKfwNgzpsHo4m+mHaPg5DUyicnU
4  /hfUDvjGcHvTQjW8O4/chtMVl2h7P8QtPi9QDcWqxmEXCLqTB6BZXrVkjQIDAQAB
```

RECOMMENDATION

Rotate every secret. Read them from environment variables or a secrets manager and make the application refuse to start if they are missing. Remove the TLS key from the repository and clean the git history. Allow the seed script only with NODE_ENV=test and make it generate random passwords. Add Gitleaks to CI.

FIXED CODE (PROPOSAL) · JAVASCRIPT

```
1  const required = (name) => {
2    const v = process.env[name];
3    if (!v) throw new Error(`Missing variable ${name}`);
4    return v;
5  };
6  module.exports = {
7    cookieSecret: required("SESSION_SECRET"),
8    cryptoKey: Buffer.from(required("DATA_KEY_B64"), "base64"),
9  };
```

REFERENCES

https://cwe.mitre.org/data/definitions/798.html
https://cheatsheetseries.owasp.org/cheatsheets/Secrets_Management_Cheat_Sheet.html

Session is not renewed at login (session fixation)

CHK-40e9d9 High 7.4 New Confirmed

Category	Insecure session management	OWASP	A07:2025 · Authentication Failures
CWE	CWE-384	ASVS 5.0	V7
Vector	CVSS:4.0/AV:N/AC:H/AT:N/PR:N/UI:A/VC:H/VI:H/VA:N/SC:N/SI:N/SA:N		
ISO 27001	8.5	NIS2	21.2.i
Detected by	Manual review		

LOCATIONS

- **app/routes/session.js:116** — userId assigned to the existing session

DESCRIPTION

After checking the password, login assigns the user to the session that already existed instead of creating a new one. Sign-up does regenerate the session; login does not.

RISK

If someone gets the victim to use a known session identifier (for example on an unencrypted network, since the app serves plain HTTP), that session becomes authenticated after login and the attacker gets into the victim's account.

DETAILS

session.js:116 sets req.session.userId = user._id without req.session.regenerate(). Aggravating factors: saveUninitialized: true creates sessions before login and the cookie has no Secure flag.

EVIDENCE

app/routes/session.js:116

```
113             // by wrapping the below code as a function callback for the method
    req.session.regenerate()
114             // i.e:
115             // `req.session.regenerate(() => {})`
116             req.session.userId = user._id;
117             return res.redirect(user.isAdmin ? "/benefits" : "/dashboard");
118         });
119     };
```

RECOMMENDATION

Regenerate the session on every login and privilege change, and destroy it on logout (already done). Set saveUninitialized: false.

FIXED CODE (PROPOSAL) · JAVASCRIPT

```
req.session.regenerate((err) => {
  if (err) return next(err);
  req.session.userId = user._id;
  return res.redirect(user.isAdmin ? "/benefits" : "/dashboard");
});
```

REFERENCES

https://cwe.mitre.org/data/definitions/384.html
https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html

NoSQL injection in the allocations filter through the $where operator

CHK-81bd08 High 7.2 New Confirmed

Category	NoSQL injection	OWASP	A05:2025 · Injection
CWE	CWE-943	ASVS 5.0	V1
Vector	CVSS:4.0/AV:N/AC:L/AT:N/PR:L/UI:N/VC:H/VI:L/VA:H/SC:N/SI:N/SA:N		
ISO 27001	8.28	NIS2	21.2.e
Detected by	checkpoint-rules		

LOCATIONS

- `app/data/allocations-dao.js:78` — threshold URL parameter

DESCRIPTION

The allocations query builds a MongoDB $where JavaScript expression by pasting in the threshold URL parameter without converting or validating it.

RISK

A registered user can change the query logic to see every customer's investment portfolio, or send an expression that locks the database and leaves the service unresponsive.

DETAILS

Source: req.query.threshold in allocations.js:20. Sink: template string inside $where in allocations-dao.js:78, which MongoDB evaluates as JavaScript on the database server. The parameter is only wrapped in quotes, which is not a defence.

EVIDENCE

app/data/allocations-dao.js:78 · checkpoint-rules · DS-JS-008

```
75                 throw `The user supplied threshold: ${parsedThreshold} was not valid.`;
76                 */
77                 return {
78                     $where: `this.userId == ${parsedUserId} && this.stocks > '${threshold}'`
79                 };
80             }
81             return {
```

RECOMMENDATION

Do not use $where. Convert threshold to an integer, check the allowed range (0-99) and use normal query operators. Disable server-side JavaScript in MongoDB (security.javascriptEnabled: false).

FIXED CODE (PROPOSAL) · JAVASCRIPT

```
1 const t = Number.parseInt(threshold, 10);
2 if (!Number.isInteger(t) || t < 0 || t > 99) {
3   return callback(new Error("invalid threshold"), null);
4 }
5 return { userId: parsedUserId, stocks: { $gt: t } };
```

REFERENCES

https://cwe.mitre.org/data/definitions/943.html
https://www.mongodb.com/docs/manual/reference/operator/query/where/
https://cheatsheetseries.owasp.org/cheatsheets/Injection_Prevention_Cheat_Sheet.html

Benefits administration functions available to any registered user

CHK-410234 High 7.1 New Confirmed

Category	Missing or broken access control	OWASP	A01:2025 · Broken Access Control
CWE	CWE-285	ASVS 5.0	V8
Vector	CVSS:4.0/AV:N/AC:L/AT:N/PR:L/UI:N/VC:L/VI:H/VA:N/SC:N/SI:N/SA:N		
ISO 27001	5.15, 8.3	NIS2	21.2.i
Detected by	Manual review		

LOCATIONS

- `app/routes/index.js:55` — GET and POST /benefits only behind isLoggedIn
- `app/routes/benefits.js:30` — userId and benefitStartDate from the body, no role check

DESCRIPTION

The /benefits routes, meant for administrators, only check that a session exists. The isAdmin middleware exists but is not applied. The handler also takes the user to modify from the form.

RISK

Any registered employee can see the user list and change anyone's benefits start date. This is an integrity failure on HR data with financial and legal impact.

DETAILS

index.js:55-56 registers the routes with isLoggedIn and leaves the isAdmin version commented out. benefits.js:30-35 takes userId from the body and calls updateBenefits without checking the session user's role. The views force user.isAdmin: true, so the interface does not tell them apart either.

EVIDENCE

app/routes/index.js:55

```
52      app.post("/contributions", isLoggedIn, contributionsHandler.handleContributionsUpdate);
53
54      // Benefits Page
55      app.get("/benefits", isLoggedIn, benefitsHandler.displayBenefits);
56      app.post("/benefits", isLoggedIn, benefitsHandler.updateBenefits);
57      /* Fix for A7 - checks user role to implement  Function Level Access Control
58       app.get("/benefits", isLoggedIn, isAdmin, benefitsHandler.displayBenefits);
59       app.post("/benefits", isLoggedIn, isAdmin, benefitsHandler.updateBenefits);
```

app/routes/benefits.js:30

```
27      };
28
29      this.updateBenefits = (req, res, next) => {
30          const {
31              userId,
32              benefitStartDate
33          } = req.body;
34
35          benefitsDAO.updateBenefits(userId, benefitStartDate, (error) => {
36
37              if (error) return next(error);
38
```

RECOMMENDATION

Apply isAdmin to both routes and deny by default: create an /admin router with the role middleware mounted once. Derive the role on the server from the session, never in the view. Add an automated test that checks a user without the role gets 403.

FIXED CODE (PROPOSAL) · JAVASCRIPT

```
app.get("/benefits", isLoggedIn, isAdmin, benefitsHandler.displayBenefits);
app.post("/benefits", isLoggedIn, isAdmin, benefitsHandler.updateBenefits);
```

REFERENCES

https://cwe.mitre.org/data/definitions/285.html
https://owasp.org/Top10/2025/A01_2025-Broken_Access_Control/
https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html

Other customers' allocations readable by changing the identifier in the URL (IDOR)

CHK-1f4cce **High** **7.1** **New** **Confirmed**

Category	Insecure direct object reference (IDOR)	OWASP	A01:2025 · Broken Access Control
CWE	CWE-639	ASVS 5.0	V8
Vector	`CVSS:4.0/AV:N/AC:L/AT:N/PR:L/UI:N/VC:H/VI:N/VA:N/SC:N/SI:N/SA:N`		
ISO 27001	5.15, 8.3	NIS2	21.2.i
Detected by	Manual review		

LOCATIONS

- **app/routes/allocations.js:16** — userId taken from req.params
- **app/routes/index.js:63** — /allocations/:userId route

DESCRIPTION

The /allocations/:userId page shows the portfolio for whatever identifier arrives in the URL, without checking that it matches the session user. Identifiers are sequential numbers.

RISK

Any registered user can read the investment split and full name of every other customer. This is a leak of personal and financial data that must be notified under the GDPR.

DETAILS

Source: req.params.userId (allocations.js:16-18). Sink: allocationsDAO.getByUserIdAndThreshold, which filters only on that value. There is no ownership check between the requested object and req.session.userId.

EVIDENCE

app/routes/allocations.js:16

```
13          // Fix for A4 Insecure DOR -  take user id from session instead of from URL param
14          const { userId } = req.session;
15          */
16          const {
17              userId
18          } = req.params;
19          const {
20              threshold
21          } = req.query;
```

app/routes/index.js:63

```
60       */
61
62      // Allocations Page
63      app.get("/allocations/:userId", isLoggedIn, allocationsHandler.displayAllocations);
64
65      // Memos Page
66      app.get("/memos", isLoggedIn, memosHandler.displayMemos);
```

RECOMMENDATION

Take the identifier from the session (req.session.userId), not from the URL. If a support role needs to see other portfolios, require that role and log every access. Consider non-sequential identifiers.

FIXED CODE (PROPOSAL) · JAVASCRIPT

```
1  const { userId } = req.session;
2  allocationsDAO.getByUserIdAndThreshold(userId, threshold, (err, allocations) => { /* … */ });
```

REFERENCES

https://cwe.mitre.org/data/definitions/639.html
https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html

Outbound request to a user-chosen URL on the research page (SSRF)

CHK-e08a94 High 7.1 New Confirmed

Category	Server-side request forgery (SSRF)	OWASP	A01:2025 · Broken Access Control
CWE	CWE-918	ASVS 5.0	V1, V4
Vector	`CVSS:4.0/AV:N/AC:L/AT:N/PR:L/UI:N/VC:H/VI:L/VA:N/SC:L/SI:L/SA:N`		
ISO 27001	8.28	NIS2	21.2.e
Detected by	semgrep		

LOCATIONS

- **`app/routes/research.js:15`** — req.query.url + req.query.symbol

DESCRIPTION

The /research page uses needle to download the address built from the url and symbol request parameters, and returns the response as HTML unchanged.

RISK

The server can be used as a gateway to read internal services that are not exposed to the Internet (admin panels, databases, cloud metadata with credentials). Because the content is returned as unfiltered HTML, it can also be used to inject content into the victim's browser.

DETAILS

Source: req.query.url and req.query.symbol (requires a session). Sink: needle.get(url) in research.js:16 with no destination allow-list, and res.write(body) with Content-Type text/html on line 25.

EVIDENCE

app/routes/research.js:15 · semgrep · node_ssrf

```
12      this.displayResearch = (req, res) => {
13
14          if (req.query.symbol) {
15              const url = req.query.url + req.query.symbol;
16              return needle.get(url, (error, newResponse, body) => {
17                  if (!error && newResponse.statusCode === 200) {
18                      res.writeHead(200, {
19                          "Content-Type": "text/html"
20                      });
21                  }
22                  res.write("<h1>The following is the stock information you requested.</h1>\n\n");
23                  res.write("\n\n");
24                  if (body) {
25                      res.write(body);
26                  }
27                  return res.end();
28              });
29          }
30
```

RECOMMENDATION

Fix the quotes provider in the server configuration and accept only the symbol from the user, validated against an allow-list (for example ^[A-Z]{1,5}$). Block private and metadata addresses on outbound traffic. Return the data

as JSON or escaped, never as third-party HTML.

FIXED CODE (PROPOSAL) · JAVASCRIPT

```
const QUOTES_BASE = config.quotesBaseUrl; // fixed on the server
const { symbol } = req.query;
if (!/^[A-Z]{1,5}$/.test(symbol || "")) {
  return res.status(400).render("research", { error: "Invalid symbol" });
}
needle.get(`${QUOTES_BASE}${encodeURIComponent(symbol)}`, (err, r, body) => {
  res.render("research", { quote: body }); // template with autoescape
});
```

REFERENCES

https://cwe.mitre.org/data/definitions/918.html
https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html

Regular expression with exponential backtracking on the bank routing number (ReDoS)

CHK-847dcb **High** **7.1** **New** Confirmed

Category	ReDoS / resource exhaustion	OWASP	A10:2025 · Mishandling of Exceptional Conditions
CWE	CWE-1333	ASVS 5.0	V2, V15
Vector	`CVSS:4.0/AV:N/AC:L/AT:N/PR:L/UI:N/VC:N/VI:N/VA:H/SC:N/SI:N/SA:N`		
ISO 27001	8.28	NIS2	21.2.e
Detected by	Manual review		

LOCATIONS

- **app/routes/profile.js:59** — pattern /([0-9]+)+\#/ on bankRouting

DESCRIPTION

Bank routing number validation uses a pattern with nested quantifiers, which takes exponential time to evaluate on certain inputs.

RISK

A single request can keep the Node.js process (which serves every user on a single thread) busy for seconds or minutes, and a few in a row take the service down.

DETAILS

Source: req.body.bankRouting. Sink: regexPattern.test() in profile.js:61 with the pattern on line 59. The pattern is not anchored either, so it does not really validate the format.

EVIDENCE

app/routes/profile.js:59

```
56          // --
57          // The Fix: Instead of using greedy quantifiers the same regex will work if we omit the
    second quantifier +
58          // const regexPattern = /([0-9]+)\#/;
59          const regexPattern = /([0-9]+)+\#/;
60          // Allow only numbers with a suffix of the letter #, for example: 'XXXXXX#'
61          const testComplyWithRequirements = regexPattern.test(bankRouting);
62          // if the regex test fails we do not allow saving
63          if (testComplyWithRequirements !== true) {
64              const firstNameSafeString = firstName;
```

RECOMMENDATION

Use a linear, anchored pattern, for example /^[0-9]{6,17}#$/, and limit the field length before evaluating it. Add a ReDoS detection rule (eslint-plugin-regexp or safe-regex) in CI.

FIXED CODE (PROPOSAL) · JAVASCRIPT

```
1 const ROUTING_RE = /^[0-9]{6,17}#$/;
2 const ok = typeof bankRouting === "string" && bankRouting.length <= 18 &&
  ROUTING_RE.test(bankRouting);
```

REFERENCES

https://cwe.mitre.org/data/definitions/1333.html
https://owasp.org/www-community/attacks/Regular_expression_Denial_of_Service_-_ReDoS

Server runs over HTTP only, with no encryption in transit

CHK-156a89 High 7.1 New Probable

Category	TLS validation disabled or cleartext channel	OWASP	A04:2025 · Cryptographic Failures
CWE	CWE-319	ASVS 5.0	V12
Vector	CVSS:4.0/AV:A/AC:L/AT:N/PR:N/UI:N/VC:H/VI:L/VA:N/SC:N/SI:N/SA:N		
ISO 27001	8.24	NIS2	21.2.h
Detected by	Manual review		

LOCATIONS

- `server.js:144` — http.createServer; HTTPS commented out

DESCRIPTION

The application starts an HTTP server without TLS. The HTTPS server is written but commented out.

RISK

Credentials, session cookies and bank details travel unencrypted and can be captured on shared networks (public Wi-Fi, corporate network).

DETAILS

server.js:144-147 uses http.createServer. It is «probable» because a real deployment might have a proxy or load balancer terminating TLS in front; that is not visible in the code and must be confirmed in the environment.

EVIDENCE

server.js:144

```
141         */
142     });
143
144     // Insecure HTTP connection
145     http.createServer(app).listen(port, () => {
146         console.log(`Express http server listening on port ${port}`);
147     });
148
149     /*
150     // Fix for A6-Sensitive Data Exposure
```

RECOMMENDATION

Always serve over HTTPS, usually with TLS terminated at the proxy or load balancer, redirect HTTP to HTTPS, enable HSTS and set app.set("trust proxy", 1) so Secure cookies work behind the proxy.

FIXED CODE (PROPOSAL) · JAVASCRIPT

```
app.set("trust proxy", 1);
app.use(helmet.hsts({ maxAge: 31536000, includeSubDomains: true }));
```

REFERENCES

https://cwe.mitre.org/data/definitions/319.html
https://cheatsheetseries.owasp.org/cheatsheets/Transport_Layer_Security_Cheat_Sheet.html

Error page shows internal exception details to the user

CHK-af97d6 **Medium** **6.9** **New** Confirmed

Category	Error handling leaks details or fails open	OWASP	A10:2025 · Mishandling of Exceptional Conditions
CWE	CWE-209	ASVS 5.0	V16
Vector	CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N		
ISO 27001	8.28	NIS2	21.2.e
Detected by	Manual review		

LOCATIONS

- `app/routes/error.js:10` — full error object passed to the view
- `app/views/error-template.html:11` — {{error}} without escaping

DESCRIPTION

The global error handler passes the whole exception object to the template, which prints it in the response.

RISK

Internal messages (queries, file paths, data structure) give an attacker hints to refine other attacks. With autoescape off, the message can also reflect request content into the HTML.

DETAILS

error.js:10-12 renders error-template with error: err. error-template.html:11 prints it. The full detail is already logged on the server with console.error, so there is no need to show it.

EVIDENCE

app/routes/error.js:10

```
    console.error(err.message);
    console.error(err.stack);
    res.status(500);
    res.render("error-template", {
        error: err
    });
};

module.exports = { errorHandler };
```

app/views/error-template.html:11

```
<body>

    Oops..
    <br>{{error}}
</body>

</html>
```

RECOMMENDATION

Show the user a generic message with an incident reference and log the detail only on the server, through a structured logger.

FIXED CODE (PROPOSAL) · JAVASCRIPT

```
const errorHandler = (err, req, res, next) => {
  const ref = crypto.randomUUID();
  logger.error({ ref, err }, "unhandled");
  res.status(500).render("error-template", { ref });
};
```

REFERENCES

https://cwe.mitre.org/data/definitions/209.html
https://cheatsheetseries.owasp.org/cheatsheets/Error_Handling_Cheat_Sheet.html

No CSRF protection on forms that change data

CHK-5b2f23 Medium 6.9 New Confirmed

Category	Cross-site request forgery (CSRF)	OWASP	A01:2025 · Broken Access Control
CWE	CWE-352	ASVS 5.0	V3, V7
Vector	`CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:A/VC:N/VI:H/VA:N/SC:N/SI:N/SA:N`		
ISO 27001	8.28	NIS2	21.2.e
Detected by	Manual review		

LOCATIONS

- **`server.js:104`** — csrf middleware commented out
- **`app/routes/index.js:48`** — POST /profile (bank details)
- **`app/routes/index.js:52`** — POST /contributions
- **`app/routes/index.js:67`** — POST /memos

DESCRIPTION

No POST request carries an anti-CSRF token. The csurf package is in the dependencies and the views include the _csrf field, but the middleware is commented out and the session cookie has no SameSite attribute.

RISK

An external website visited by a logged-in user can submit forms on their behalf: change their bank account and routing number, their contributions or post memos.

DETAILS

server.js:104-113 leaves app.use(csrf()) commented out. The POST routes in index.js do not check origin or token. The session cookie in server.js:78 uses default settings, without SameSite.

EVIDENCE

server.js:104

```
101
102     }));
103
104     /*
105     // Fix for A8 - CSRF
106     // Enable Express csrf protection
107     app.use(csrf());
108     // Make csrf token available in templates
109     app.use((req, res, next) => {
110         res.locals.csrftoken = req.csrfToken();
111         next();
112     });
113     */
114
115     // Register templating engine
116     app.engine(".html", consolidate.swig);
```

app/routes/index.js:48

```
45
46      // Profile page
47      app.get("/profile", isLoggedIn, profileHandler.displayProfile);
48      app.post("/profile", isLoggedIn, profileHandler.handleProfileUpdate);
49
50      // Contributions Page
51      app.get("/contributions", isLoggedIn, contributionsHandler.displayContributions);
```

RECOMMENDATION

Enable synchronizer-token CSRF protection (csrf-sync or csrf-csrf; csurf is deprecated) on every state-changing route and set the session cookie to SameSite=Lax or Strict. Check the Origin or Referer header on critical operations.

FIXED CODE (PROPOSAL) · JAVASCRIPT

```
1 const { csrfSynchronisedProtection, generateToken } = require("csrf-sync")();
2 app.use(csrfSynchronisedProtection);
3 app.use((req, res, next) => { res.locals.csrftoken = generateToken(req); next(); });
```

REFERENCES

https://cwe.mitre.org/data/definitions/352.html
https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html

Unsanitized usernames written to the application log

CHK-6ea3ab Medium 6.9 New Confirmed

Category	Log injection	OWASP	A09:2025 · Security Logging and Alerting Failures
CWE	CWE-117	ASVS 5.0	V16
Vector	CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:N/VI:L/VA:N/SC:N/SI:N/SA:N		
ISO 27001	8.15	NIS2	21.2.b
Detected by	Manual review		

LOCATIONS

- `app/routes/session.js:64` — userName from the login form

DESCRIPTION

Login attempts with a non-existent user are logged with the name as received, line breaks included. Successful logins and wrong-password failures are not logged.

RISK

Log lines can be forged to hide an attack or confuse an investigation. And because key authentication events are missing, a brute-force attack would go unnoticed.

DETAILS

session.js:64 concatenates userName into console.log without neutralizing CR/LF. The wrong-password and successful-login branches produce no event.

EVIDENCE

app/routes/session.js:64

```
61              const invalidPasswordErrorMessage = "[REDACTADO]";
62              if (err) {
63                  if (err.noSuchUser) {
64                      console.log("Error: attempt to login with invalid user: ", userName);
65
66                      // Fix for A1 - 3 Log Injection - encode/sanitize input for CRLF Injection
67                      // that could result in log forging:
```

RECOMMENDATION

Use a structured logger (JSON, for example pino) that escapes values. Log every authentication event (success, failure, lockout) without passwords and send them to a system with alerting.

FIXED CODE (PROPOSAL) · JAVASCRIPT

```
logger.warn({ event: "login_failed", reason: "no_such_user", user: String(userName).slice(0, 64), ip: req.ip });
```

REFERENCES

https://cwe.mitre.org/data/definitions/117.html
https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html

Weak authentication: user enumeration, short passwords and no attempt limit

CHK-92db72 Medium 6.9 New Confirmed

Category	Weak or bypassable authentication	OWASP	A07:2025 · Authentication Failures
CWE	CWE-287	ASVS 5.0	V6
Vector	CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:L/VI:L/VA:N/SC:N/SI:N/SA:N		
ISO 27001	8.5, 5.17	NIS2	21.2.i, 21.2.j
Detected by	Manual review		

LOCATIONS

- **app/routes/session.js:60** — different message for user and for password
- **app/routes/session.js:144** — policy: any 1 to 20 characters
- **app/routes/index.js:34** — POST /login with no rate limiting

DESCRIPTION

Login answers «invalid username» or «invalid password» depending on the case, the sign-up policy accepts one-character passwords and there is no attempt limit or progressive lockout. There is no MFA either.

RISK

An attacker can find out which users exist and try passwords automatically without any brake, with a good chance of getting into accounts with weak passwords.

DETAILS

session.js:60-61 and 82-98 return different messages. PASS_RE in session.js:144 is /^.{1,20}$/ (the error message says 8 to 18 characters, but it is not enforced). index.js:34 registers POST /login with no rate-limit middleware.

EVIDENCE

app/routes/session.js:60

```
57          } = req.body;
58          userDAO.validateLogin(userName, password, (err, user) => {
59              const errorMessage = "Invalid username and/or password";
60              const invalidUserNameErrorMessage = "Invalid username";
61              const invalidPasswordErrorMessage = "[REDACTADO]";
62              if (err) {
63                  if (err.noSuchUser) {
64                      console.log("Error: attempt to login with invalid user: ", userName);
```

app/routes/session.js:144

```
141         const FNAME_RE = /^.{1,100}$/;
142         const LNAME_RE = /^.{1,100}$/;
143         const EMAIL_RE = /^[\S]+@[\S]+\.[\S]+$/;
144         const PASS_RE = /^.{1,20}$/;
145         /*
146         //Fix for A2-2 - Broken Authentication -  requires stronger password
147         //(at least 8 characters with numbers and both lowercase and uppercase letters.)
```

RECOMMENDATION

Use a single message for any login failure. Require at least 12 characters and check against breached-password lists (ASVS V6). Limit attempts per account and per IP (express-rate-limit) with progressive delay. Offer MFA, mandatory for administrators.

FIXED CODE (PROPOSAL) · JAVASCRIPT

```
const rateLimit = require("express-rate-limit");
const loginLimiter = rateLimit({ windowMs: 15 * 60 * 1000, limit: 10, standardHeaders: true });
app.post("/login", loginLimiter, sessionHandler.handleLoginRequest);
// session.js
const PASS_RE = /^.{12,128}$/;
const loginError = "Incorrect username or password";
```

REFERENCES

https://cwe.mitre.org/data/definitions/287.html
https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html
https://owasp.org/Top10/2025/A07_2025-Authentication_Failures/

Abandoned dependencies or very old versions

CHK-d06a9e | Medium | 6.3 | New | Probable

Category	Dependency with known vulnerabilities	OWASP	A03:2025 · Software Supply Chain Failures
CWE	CWE-1104	ASVS 5.0	V15
Vector	`CVSS:4.0/AV:N/AC:L/AT:P/PR:N/UI:N/VC:L/VI:L/VA:L/SC:N/SI:N/SA:N`		
ISO 27001	8.8, 5.21	NIS2	21.2.d, 21.2.e
Detected by	Manual review		

LOCATIONS

- **package.json:8** — bcrypt-nodejs 0.0.3, abandoned package
- **package.json:17** — marked 0.3.5, pinned 2015 version
- **package.json:19** — needle 2.2.4, old pinned version
- **package.json:20** — node-esapi 0.0.1, unmaintained
- +1 more locations

DESCRIPTION

Several production dependencies are pinned to versions that are years old or belong to abandoned projects that no longer publish security fixes (swig, bcrypt-nodejs, node-esapi). The declared mongodb (2.x) and helmet (2.x) ranges are also old.

RISK

Known flaws in these libraries will not be fixed, and some of them (templates, Markdown, HTTP client) sit directly in the path of user data.

DETAILS

Assessed by version and maintenance status. OSV-Scanner could not run on the analysis machine, so no specific CVE identifiers are listed: confirming them with npm audit or OSV-Scanner in the client's pipeline is recommended.

EVIDENCE

package.json:8

```
5   "description": "A tool to learn OWASP Top 10 for node.js developers",
6   "main": "server.js",
7   "dependencies": {
8     "bcrypt-nodejs": "0.0.3",
9     "body-parser": "^1.15.1",
10    "consolidate": "^0.14.1",
11    "csurf": "^1.8.3",
```

package.json:17

```
14    "express-session": "^1.13.0",
15    "forever": "^2.0.0",
16    "helmet": "^2.0.0",
17    "marked": "0.3.5",
18    "mongodb": "^2.1.18",
19    "needle": "2.2.4",
20    "node-esapi": "0.0.1",
```

RECOMMENDATION

Replace swig with a maintained engine (Nunjucks or Handlebars), bcrypt-nodejs with argon2 or bcrypt, and node-esapi with the engine's own escaping. Update marked, needle, mongodb and helmet to current versions. Add npm audit or OSV-Scanner and Dependabot or Renovate to CI.

REFERENCES

https://cwe.mitre.org/data/definitions/1104.html
https://owasp.org/Top10/2025/A03_2025-Software_Supply_Chain_Failures/
https://google.github.io/osv-scanner/

GitHub Actions referenced by mutable tag

CHK-9badba Medium 6.3 New Confirmed

Category	Supply chain or CI/CD risk	OWASP	A03:2025 · Software Supply Chain Failures
CWE	CWE-829	ASVS 5.0	V15
Vector	`CVSS:4.0/AV:N/AC:H/AT:P/PR:N/UI:N/VC:L/VI:L/VA:N/SC:N/SI:N/SA:N`		
ISO 27001	5.21, 8.25	NIS2	21.2.d
Detected by	checkpoint-rules, semgrep		

LOCATIONS

- `.github/workflows/e2e-test.yml:16`
- `.github/workflows/e2e-test.yml:21`
- `.github/workflows/e2e-test.yml:26`
- `.github/workflows/e2e-test.yml:58`
- +8 more locations

DESCRIPTION

The CI workflows use actions by tag (actions/checkout@v2, setup-node@v1, cache@v2, upload-artifact@v2), in old versions and not pinned to a SHA.

RISK

If the action's owner or an attacker moves the tag, the pipeline would run different code without warning, with access to the code and the repository secrets.

DETAILS

e2e-test.yml and lint.yml reference actions by major tag. It is a good sign that checkout uses persist-credentials: false.

EVIDENCE

`.github/workflows/e2e-test.yml:16 · semgrep · github-actions-mutable-action-tag`

```
13
14      steps:
15        - name: Checkout https://github.com/${{ github.repository }}@${{ github.ref }}
16          uses: actions/checkout@v2
17          with:
18            persist-credentials: false
19
```

`.github/workflows/e2e-test.yml:21 · semgrep · github-actions-mutable-action-tag`

```
18            persist-credentials: false
19
20        - name: Set up Node.js ${{ matrix.node-version }}
21          uses: actions/setup-node@v1
22          with:
23            node-version: ${{ matrix.node-version }}
24
```

RECOMMENDATION

Pin every action to a full commit SHA with a version comment, update to current major versions and let Dependabot propose the changes.

FIXED CODE (PROPOSAL) · YAML

```yaml
- uses: actions/checkout@<40-character-sha> # v4
  with:
    persist-credentials: false
```

REFERENCES

https://docs.github.com/en/actions/security-for-github-actions/security-guides/security-hardening-for-github-actions#using-third-party-actions

Personal and banking data stored unencrypted

CHK-f909bf | Medium | 5.9 | New | Confirmed

Category	Weak or obsolete cryptography	OWASP	A04:2025 · Cryptographic Failures
CWE	CWE-311	ASVS 5.0	V11
Vector	`CVSS:4.0/AV:N/AC:H/AT:P/PR:H/UI:N/VC:H/VI:N/VA:N/SC:N/SI:N/SA:N`		
ISO 27001	8.24	NIS2	21.2.h
Detected by	Manual review		

LOCATIONS

- `app/data/profile-dao.js:55` — bankAcc, bankRouting, ssn and dob in clear

DESCRIPTION

The profile stores the social security number, date of birth and bank details in clear in MongoDB. The encryption code is written but commented out.

RISK

A database leak exposes specially protected data (national identifier and bank accounts), with a duty to notify the authority and those affected, and a risk of identity fraud.

DETAILS

updateUser() in profile-dao.js:55-66 copies the form fields to the document without transformation. The commented-out encryption block also reused the IV stored in config, which would not be correct either.

EVIDENCE

app/data/profile-dao.js:55

```
52          if (address) {
53              user.address = address;
54          }
55          if (bankAcc) {
56              user.bankAcc = bankAcc;
57          }
58          if (bankRouting) {
59              user.bankRouting = bankRouting;
60          }
61          if (ssn) {
62              user.ssn = ssn;
63          }
64          if (dob) {
65              user.dob = dob;
66          }
67          /*
68          // Fix for A7 - Sensitive Data Exposure
69          // Store encrypted ssn and DOB
```

RECOMMENDATION

Encrypt at field level with AES-256-GCM and a random IV per value, storing IV and tag next to the data, with the key in a secrets manager (or use MongoDB field encryption, CSFLE/Queryable Encryption). Show only the last digits in the interface.

FIXED CODE (PROPOSAL) · JAVASCRIPT

```
1 const encrypt = (plain) => {
2   const iv = crypto.randomBytes(12);
3   const c = crypto.createCipheriv("aes-256-gcm", key, iv);
4   const data = Buffer.concat([c.update(plain, "utf8"), c.final()]);
5   return [iv, c.getAuthTag(), data].map(b => b.toString("base64")).join(".");
6 };
```

REFERENCES

https://cwe.mitre.org/data/definitions/311.html
https://cheatsheetseries.owasp.org/cheatsheets/Cryptographic_Storage_Cheat_Sheet.html
https://www.mongodb.com/docs/manual/core/csfle/

User passwords stored and compared in plain text

CHK-864901 Medium 5.9 New Confirmed

Category	Insecure password storage	OWASP	A04:2025 · Cryptographic Failures
CWE	CWE-256	ASVS 5.0	V6, V11
Vector	CVSS:4.0/AV:N/AC:H/AT:P/PR:H/UI:N/VC:H/VI:N/VA:N/SC:N/SI:N/SA:N		
ISO 27001	8.5, 8.24, 5.17	NIS2	21.2.h
Detected by	Manual review		

LOCATIONS

- `app/data/user-dao.js:25` — password inserted as received
- `app/data/user-dao.js:60` — direct comparison with ===

DESCRIPTION

On sign-up the password is stored in MongoDB without hashing. On login it is compared directly with the one received. The bcrypt library is imported but its use is commented out.

RISK

Any copy of the database (backup, insider access, another vulnerability) exposes every user's password, which people tend to reuse on other services, including corporate email.

DETAILS

addUser() in user-dao.js:25 stores the password field from the form. validateLogin() in user-dao.js:60-61 compares with ===, which confirms there is no hash. bcrypt-nodejs is also an abandoned package.

EVIDENCE

app/data/user-dao.js:25

```
22              firstName,
23              lastName,
24              benefitStartDate: this.getRandomFutureDate(),
25              password //received from request param
26              /*
27              // Fix for A2-1 - Broken Auth
28              // Stores password  in a safer way using one way encryption and salt hashing
```

app/data/user-dao.js:60

```
57      this.validateLogin = (userName, password, callback) => {
58
59          // Helper function to compare passwords
60          const comparePassword = (fromDB, fromUser) => {
61              return fromDB === fromUser;
62              /*
63              // Fix for A2-Broken Auth
64              // compares decrypted password stored in this.addUser()
```

RECOMMENDATION

Store passwords with a slow, salted key-derivation function: argon2id (argon2 package) or bcrypt (bcrypt package, cost 12 or higher). Migrate existing ones with a rehash on next login or by forcing a password change.

FIXED CODE (PROPOSAL) · JAVASCRIPT

```
1 const argon2 = require("argon2");
2 // sign-up
3 user.password = await argon2.hash(password, { type: argon2.argon2id });
4 // login
5 const ok = await argon2.verify(user.password, password);
```

REFERENCES

https://cwe.mitre.org/data/definitions/256.html
https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html

Session cookie without security attributes and no HTTP protection headers

CHK-e4a611 Medium 5.1 New Confirmed

Category	Missing security headers or cookie flags	OWASP	A02:2025 · Security Misconfiguration
CWE	CWE-1004	ASVS 5.0	V3, V13
Vector	`CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:A/VC:L/VI:L/VA:N/SC:N/SI:L/SA:N`		
ISO 27001	8.9	NIS2	21.2.e
Detected by	Manual review, semgrep		

LOCATIONS

- **`server.js:78`** — no SameSite, Secure or expiry
- **`server.js:78`** — default name and no HttpOnly
- **`server.js:38`** — helmet and nosniff commented out; x-powered-by active

DESCRIPTION

The session cookie uses default settings: no HttpOnly, Secure, SameSite or expiry, and the standard connect.sid name. Helmet is installed but disabled, so no CSP, HSTS, X-Frame-Options or X-Content-Type-Options are sent, and the X-Powered-By header reveals Express.

RISK

Without these defences any XSS steals the session, the page can be embedded in another site to trick the user (clickjacking) and the browser does not help stop attacks it would normally block.

DETAILS

server.js:78-102 calls session() with no cookie object. server.js:38-65 has helmet.frameguard, contentSecurityPolicy, hsts and nosniff commented out. The declared helmet version (2.x) is also very old.

EVIDENCE

`server.js:78 · semgrep · cookie_session_no_samesite`

```
75      }));
76
77      // Enable session management using express middleware
78      app.use(session({
79          // genid: (req) => {
80          //    return genuuid() // use UUIDs for session IDs
81          //},
82          secret: [REDACTADO]
83          // Both mandatory in Express v4
84          saveUninitialized: true,
85          resave: true
86          /*
87          // Fix for A5 - Security MisConfig
88          // Use generic cookie name
89          key: "sessionId",
90          */
91
92          /*
93          // Fix for A3 - XSS
```

server.js:78 · semgrep · express-cookie-session-default-name

```
75      }));
76
77      // Enable session management using express middleware
78      app.use(session({
79          // genid: (req) => {
80          //    return genuuid() // use UUIDs for session IDs
81          //},
82          secret: [REDACTADO]
83          // Both mandatory in Express v4
84          saveUninitialized: true,
85          resave: true
86          /*
87          // Fix for A5 - Security MisConfig
88          // Use generic cookie name
89          key: "sessionId",
90          */
91
92          /*
93          // Fix for A3 - XSS
```

RECOMMENDATION

Configure the cookie with httpOnly, secure, sameSite: "lax", maxAge and a generic name. Update helmet to the current version and enable it with a strict CSP. Disable x-powered-by.

FIXED CODE (PROPOSAL) · JAVASCRIPT

```javascript
app.disable("x-powered-by");
app.use(helmet());
app.use(session({
  name: "sid",
  secret: config.cookieSecret,
  resave: false,
  saveUninitialized: false,
  cookie: { httpOnly: true, secure: true, sameSite: "lax", maxAge: 30 * 60 * 1000 }
}));
```

REFERENCES

https://cwe.mitre.org/data/definitions/1004.html
https://expressjs.com/en/advanced/best-practice-security.html
https://helmetjs.github.io/

Template auto-escaping disabled: stored XSS in profile and memos

CHK-3993f8 **Medium** **5.1** **New** Confirmed

Category	Cross-site scripting (XSS)	OWASP	A05:2025 · Injection
CWE	CWE-79	ASVS 5.0	V1, V3
Vector	`CVSS:4.0/AV:N/AC:L/AT:N/PR:L/UI:P/VC:N/VI:N/VA:N/SC:L/SI:L/SA:N`		
ISO 27001	8.28	NIS2	21.2.e
Detected by	Manual review, semgrep		

LOCATIONS

- **`server.js:135`** — swig autoescape: false for every view
- **`app/views/profile.html:78`** — firstNameSafeString in an href without URL encoding
- **`app/routes/profile.js:28`** — HTML encoding applied to a value used as a URL
- **`app/views/memos.html:31`** — user memo rendered with marked 0.3.5

DESCRIPTION

The Swig template engine is configured with autoescape: false, so everything a user saves (first name, last name, address, memos) is written into HTML without escaping. On the profile page, HTML encoding is also used for a value that goes inside a link, which needs URL encoding.

RISK

A user can store content that runs in other users' or the administrator's browser when they view the profile or the memo board. That can be used to steal sessions (the cookie has no HttpOnly flag), act on their behalf or show fake forms.

DETAILS

Source: profile fields (POST /profile) and memo text (POST /memos), stored in MongoDB. Sink: {{...}} variables in the views, which are inserted as HTML with autoescape off. Memos go through marked 0.3.5 with the sanitize option, an old version whose sanitizing is not considered reliable.

EVIDENCE

server.js:135

```
132     routes(app, db);
133
134     // Template system setup
135     swig.setDefaults({
136         // Autoescape disabled
137         autoescape: false
138         /*
139         // Fix for A3 - XSS, enable auto escaping
140         autoescape: true // default value
141         */
142     });
143
144     // Insecure HTTP connection
145     http.createServer(app).listen(port, () => {
```

app/views/profile.html:78 · semgrep · var-in-href

```
75
76                     <!-- @FIXME use a properly escaped variable that matches the URL context, for example
77                     refer to a firstNameSafeURLString field on the doc object set by the controller for this template -->
78                     <a href="{{firstNameSafeString}}">Google search this profile by name</a>
79                 </form>
80             </div>
81         </div>
```

RECOMMENDATION

Set autoescape: true in Swig (the default) or, better, move to a maintained engine with context-aware escaping (Nunjucks, Handlebars). URL-encode values used in href and check they start with https://. Update marked and sanitize its output with DOMPurify. Add a Content-Security-Policy that blocks inline scripts.

FIXED CODE (PROPOSAL) · JAVASCRIPT

```
swig.setDefaults({ autoescape: true });

// profile.js: value used in a link
const url = String(doc.website || "");
doc.websiteHref = /^https:\/\//i.test(url) ? encodeURI(url) : "#";
```

REFERENCES

https://cwe.mitre.org/data/definitions/79.html
https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html
https://owasp.org/Top10/2025/A05_2025-Injection/

Open redirect in the learning resources link

CHK-0edfaa **Medium** **4.8** **New** Confirmed

Category	Open redirect	OWASP	A01:2025 · Broken Access Control
CWE	CWE-601	ASVS 5.0	V3, V15
Vector	`CVSS:4.0/AV:N/AC:L/AT:N/PR:L/UI:A/VC:N/VI:N/VA:N/SC:L/SI:L/SA:N`		
ISO 27001	8.28	NIS2	21.2.e
Detected by	checkpoint-rules, semgrep		

LOCATIONS

- **app/routes/index.js:72** — req.query.url
- **app/routes/index.js:72** — same point (Semgrep)

DESCRIPTION

The /learn route redirects to the address in the url parameter without checking the destination.

RISK

A link that starts with the legitimate domain can lead to a phishing site imitating the login, which makes the scam more credible.

DETAILS

Source: req.query.url. Sink: res.redirect() in index.js:72 with no allow-list. It requires a session, which narrows the reach but does not remove it.

EVIDENCE

app/routes/index.js:72 · checkpoint-rules · DS-JS-009

```
69      // Handle redirect for learning resources link
70      app.get("/learn", isLoggedIn, (req, res) => {
71          // Insecure way to handle redirects by taking redirect url from query string
72          return res.redirect(req.query.url);
73      });
74
75      // Research Page
```

app/routes/index.js:72 · semgrep · express_open_redirect

```
69      // Handle redirect for learning resources link
70      app.get("/learn", isLoggedIn, (req, res) => {
71          // Insecure way to handle redirects by taking redirect url from query string
72          return res.redirect(req.query.url);
73      });
74
75      // Research Page
```

RECOMMENDATION

Redirect only to allow-listed destinations identified by a key rather than a full URL, or show an interstitial page warning that the user is leaving the site.

FIXED CODE (PROPOSAL) · JAVASCRIPT

```
1 const LEARN = { owasp: "https://owasp.org/www-project-top-ten/", node:
  "https://nodejs.org/en/learn" };
2 app.get("/learn", isLoggedIn, (req, res) => {
3   const dest = LEARN[req.query.id];
4   return dest ? res.redirect(dest) : res.status(400).send("Destination not allowed");
5 });
```

REFERENCES

https://cwe.mitre.org/data/definitions/601.html
https://cheatsheetseries.owasp.org/cheatsheets/Unvalidated_Redirects_and_Forwards_Cheat_Sheet.html

7. VULNERABLE DEPENDENCIES

No dependencies with known vulnerabilities were found in the analysed manifests.

8. CODE QUALITY AND PRACTICES

Assessment of how the code is written from a security standpoint (OWASP Top 10 Proactive Controls).

Aspect	Assessment	Note
Use of framework defences (C2)	Poor	Helmet, csurf and bcrypt are in the dependencies but disabled or commented out (server.js:38-65 and 104-113, user-dao.js:26-30). Template auto-escaping is switched off on purpose.
Secure data access (C3)	Needs work	Most queries use filter objects with parseInt, which is right, but allocations-dao.js builds a $where from user text. There is no shared input validation layer.
Output encoding and escaping (C4)	Poor	Global autoescape: false; manual encoding with node-esapi applied in the wrong context (profile.js:28); third-party HTML forwarded unchanged (research.js:25).
Validate all inputs (C5)	Poor	eval() used as a number parser, unanchored or ReDoS-prone regular expressions and a password policy that does not match its own message. No validation schema (zod, joi).
Digital identity and sessions (C6)	Poor	Plain-text passwords, messages that enumerate users, no attempt limit, no MFA and no session regeneration at login.
Access control (C7)	Poor	Authorization is applied route by route and forgotten on /benefits; /allocations trusts the id in the URL. A centralized deny-by-default approach is missing.
Data protection (C8)	Poor	Unencrypted personal and banking data, secrets in the code and an HTTP-only server.
Logging and monitoring (C9)	Poor	Scattered console.log calls, incomplete authentication events, user data not neutralized and no central destination.
Errors and exceptions (C10)	Needs work	There is a global handler (good), but it shows the full error to the user. Some callbacks can be called twice (allocations-dao.js:49-53).
Dependencies and CI	Needs work	There is a lockfile, lint and end-to-end tests in CI (good), but with abandoned dependencies, actions not pinned by SHA and no security analysis (SAST, secrets, dependencies) in the pipeline.

9. COMPLIANCE AND FRAMEWORKS

OWASP Top 10:2025

Category	Critical	High	Medium	Low	Info	Related findings
A01 Broken Access Control		3	2			`CHK-410234`, `CHK-1f4cce`, `CHK-e08a94`, `CHK-5b2f23`, `CHK-0edfaa`
A02 Security Misconfiguration			1			`CHK-e4a611`
A03 Software Supply Chain Failures			2			`CHK-d06a9e`, `CHK-9badba`
A04 Cryptographic Failures		1	2			`CHK-156a89`, `CHK-f909bf`, `CHK-864901`
A05 Injection	1	1	1			`CHK-ccd14d`, `CHK-81bd08`, `CHK-3993f8`
A06 Insecure Design						
A07 Authentication Failures	1	1	1			`CHK-67ab09`, `CHK-40e9d9`, `CHK-92db72`
A08 Software or Data Integrity Failures						
A09 Security Logging and Alerting Failures			1			`CHK-6ea3ab`
A10 Mishandling of Exceptional Conditions		1	1			`CHK-847dcb`, `CHK-af97d6`

OWASP ASVS 5.0 by chapter

Chapter	Status	Note	Related findings
V1 Encoding and Sanitization	Does not meet	eval() and $where with user data; output escaping disabled.	`CHK-ccd14d`, `CHK-81bd08`, `CHK-e08a94`, `CHK-3993f8`
V2 Validation and Business Logic	Does not meet	Weak or missing validation (percentages, threshold, ReDoS-prone routing number).	`CHK-847dcb`
V3 Web Frontend Security	Does not meet	No CSP or X-Frame-Options; stored XSS; cookie without SameSite.	`CHK-5b2f23`, `CHK-e4a611`, `CHK-3993f8`, `CHK-0edfaa`
V4 API and Web Service	Partial	Server-rendered app without a public API; the research page makes unrestricted outbound requests (SSRF).	`CHK-e08a94`
V5 File Handling	Not applicable	The application does not handle file upload or download.	
V6 Authentication	Does not meet	Plain-text passwords, 1-character minimum policy, user enumeration and no attempt limit or MFA.	`CHK-92db72`, `CHK-864901`

Chapter	Status	Note	Related findings
V7 Session Management	**Does not meet**	No regeneration at login; cookie without HttpOnly/Secure/SameSite or expiry.	`CHK-40e9d9`, `CHK-5b2f23`
V8 Authorization	**Does not meet**	Administration functions without role check and IDOR on allocations.	`CHK-410234`, `CHK-1f4cce`
V9 Self-contained Tokens	Not applicable	No self-contained tokens (JWT); server-side session.	
V10 OAuth and OIDC	Not applicable	Does not use OAuth or OIDC.	
V11 Cryptography	**Does not meet**	No password hashing or encryption of sensitive data; literal encryption key.	`CHK-f909bf`, `CHK-864901`
V12 Secure Communication	**Does not meet**	HTTP server without TLS or HSTS (to be confirmed if a TLS proxy sits in front).	`CHK-156a89`
V13 Configuration	**Does not meet**	Secrets and TLS key in the repository; security configuration disabled.	`CHK-67ab09`, `CHK-e4a611`
V14 Data Protection	**Does not meet**	SSN and bank details in clear in the database.	`CHK-67ab09`
V15 Secure Coding and Architecture	**Does not meet**	Abandoned and old dependencies; CI actions not pinned.	`CHK-ccd14d`, `CHK-847dcb`, `CHK-d06a9e`, `CHK-9badba` …
V16 Security Logging and Error Handling	**Does not meet**	Internal errors shown to the user and unstructured logging without authentication events.	`CHK-af97d6`, `CHK-6ea3ab`
V17 WebRTC	Not applicable	Does not use WebRTC.	

ISO/IEC 27001:2022 (Annex A) and NIS2 (Art. 21.2)

Control	Related findings
ISO 27001 A.5.15 Access control	`CHK-410234`, `CHK-1f4cce`
ISO 27001 A.5.17 Authentication information	`CHK-67ab09`, `CHK-92db72`, `CHK-864901`
ISO 27001 A.5.21 ICT supply chain security	`CHK-d06a9e`, `CHK-9badba`
ISO 27001 A.8.3 Information access restriction	`CHK-410234`, `CHK-1f4cce`
ISO 27001 A.8.4 Access to source code	`CHK-67ab09`
ISO 27001 A.8.5 Secure authentication	`CHK-40e9d9`, `CHK-92db72`, `CHK-864901`
ISO 27001 A.8.8 Management of technical vulnerabilities	`CHK-d06a9e`
ISO 27001 A.8.9 Configuration management	`CHK-e4a611`
ISO 27001 A.8.12 Data leakage prevention	`CHK-67ab09`
ISO 27001 A.8.15 Logging	`CHK-6ea3ab`
ISO 27001 A.8.24 Use of cryptography	`CHK-156a89`, `CHK-f909bf`, `CHK-864901`

Control	Related findings
ISO 27001 A.8.25 Secure development life cycle	`CHK-9badba`
ISO 27001 A.8.28 Secure coding	`CHK-ccd14d, CHK-81bd08, CHK-e08a94, CHK-847dcb, CHK-af97d6, CHK-5b2f23, CHK-3993f8, CHK-0edfaa`
ISO 27001 A.8.29 Security testing in development and acceptance	`Checkpoint`
NIS2 Art. 21.2.d) Supply chain security	`CHK-d06a9e, CHK-9badba`
NIS2 Art. 21.2.e) Security in acquisition, development and maintenance, including vulnerability handling	`CHK-ccd14d, CHK-81bd08, CHK-e08a94, CHK-847dcb, CHK-af97d6, CHK-5b2f23, CHK-d06a9e, CHK-e4a611 …`
NIS2 Art. 21.2.h) Cryptography and encryption	`CHK-156a89, CHK-f909bf, CHK-864901`
NIS2 Art. 21.2.i) Access control and asset management	`CHK-67ab09, CHK-40e9d9, CHK-410234, CHK-1f4cce, CHK-92db72`
NIS2 Art. 21.2.j) Multi-factor authentication	`CHK-92db72`
NIS2 Art. 21.2.b) Incident handling	`CHK-6ea3ab`

10. ANNEXES

Low or informational automated leads not reviewed individually

Listed by rule for the development team to review. They are not counted as findings.

None

Leads dismissed during triage

Rule	Findings	Reason
`semgrep · node_insecure_random_generator`	7	Math.random() is used to generate demo data (dates and portfolio percentages), not tokens, session identifiers or anything with security value.
`checkpoint-rules · DS-JS-015`	7	Math.random() is used to generate demo data (dates and portfolio percentages), not tokens, session identifiers or anything with security value.
`semgrep · plaintext-http-link`	5	http:// links to external documentation on the tutorial pages. They carry no application or user data.
`semgrep · node_username`	3	False positives: these are interface error messages containing the words «password» or «user name», not credentials.
`semgrep · node_password`	2	False positives: these are interface error messages containing the words «password» or «user name», not credentials.
`semgrep · express_lfr_warning`	2	render() receives a fixed view name or one from a closed list defined in code (pages), not from the request.
`checkpoint-rules · DS-GEN-004`	2	TODO comments inside commented-out configuration blocks; their content is already covered by the headers and cookies finding.
`checkpoint-rules · DS-JS-008`	1	Line inside a commented-out block (the project's own proposed fix, using parseInt and a range). It is not active code.
`checkpoint-rules · DS-GEN-002`	1	Credential of a test user in the end-to-end tests. Not part of the deployed application.
`semgrep · regex_dos`	1	The pattern /^.{1,20}$/ is anchored and linear, with no nested quantifiers: it cannot cause exponential backtracking.
`semgrep · node_timing_attack`	1	Compares the password with its confirmation in the same sign-up form. There is no stored secret to leak through response timing.

Glossary

SAST	Static application security testing: analysing source code without running it.
Source → sink	Path of user-controlled data (source) to a dangerous operation (sink) without validation or encoding.
CVSS 4.0	Common Vulnerability Scoring System (FIRST), version 4.0.
CWE	Catalogue of software weakness types (MITRE).
OWASP Top 10	The ten most critical web application risks (2025 edition).
ASVS	OWASP Application Security Verification Standard (version 5.0).
Triage	Manual review of each automated lead to confirm it or dismiss it with a reason.