

# https://sekumart.sekuripy.hr/

scope **https://sekumart.sekuripy.hr/** access **unauthenticated**

## CRITICAL

13 vulnerability families · 31 confirmed locations

59 additional items require validation

3 weaknesses tie at critical: SQL injection, NoSQL injection, XPath injection. They are listed by how many locations each one touches, which is not the same as which to fix first - that follows from what the affected code can reach.

|    |                 |                                                              |                                                                                  |                           |
|----|-----------------|--------------------------------------------------------------|----------------------------------------------------------------------------------|---------------------------|
| 1  | <b>CRITICAL</b> | <b>SQL injection 8 locations</b>                             | Parameterise database queries                                                    | <a href="#">details ↓</a> |
| 2  | <b>CRITICAL</b> | <b>NoSQL injection 1 location</b>                            | Reject non-scalar parameters                                                     | <a href="#">details ↓</a> |
| 3  | <b>CRITICAL</b> | <b>XPath injection 1 location</b>                            | Bind XPath variables                                                             | <a href="#">details ↓</a> |
| 4  | <b>HIGH</b>     | <b>Credential material returned to the client 1 location</b> | Stop returning credential material, and re-hash with a password-hashing function | <a href="#">details ↓</a> |
| 5  | <b>MEDIUM</b>   | <b>Page can be framed by any origin 18 locations</b>         | Send `Content-Security-Policy: frame-ancestors 'none'`                           | <a href="#">details ↓</a> |
| 6  | <b>MEDIUM</b>   | <b>Cross-site scripting 6 locations</b>                      | Encode on output for the context                                                 | <a href="#">details ↓</a> |
| 7  | <b>MEDIUM</b>   | <b>Stored / second-order input 2 locations</b>               | Encode at the point of output                                                    | <a href="#">details ↓</a> |
| 8  | <b>MEDIUM</b>   | <b>DOM-based scripting 1 location</b>                        | Write untrusted values with textContent                                          | <a href="#">details ↓</a> |
| 9  | <b>MEDIUM</b>   | <b>Outdated platform 1 location</b>                          | Confirm the running version, then update or backport                             | <a href="#">details ↓</a> |
| 10 | <b>LOW</b>      | <b>Transport security 24 locations</b>                       | Serve and require TLS everywhere                                                 | <a href="#">details ↓</a> |
| 11 | <b>LOW</b>      | <b>Open redirect 1 location</b>                              | Allow-list redirect destinations                                                 | <a href="#">details ↓</a> |
| 12 | <b>LOW</b>      | <b>Session handling 1 location</b>                           | Harden session cookie flags                                                      | <a href="#">details ↓</a> |
| 13 | <b>LOW</b>      | <b>Weak cookie flags 1 location</b>                          | Add HttpOnly and Secure                                                          | <a href="#">details ↓</a> |

by location: 13 critical 1 high 13 medium 4 low

Unauthenticated scan · 164 input surfaces · 14736 active probes · 6 observations · 137 clean results

### Marker values were written to 4 persistent forms

/cart.php, /challenges.php, /login.php, /reviews.php store what they are given. The values are inert strings, but they are visible to your users until removed.

**APPLICATION SECURITY POSTURE** 12 of 12 areas assessed in this scan**F**

OVERALL

|                                     |           |                             |
|-------------------------------------|-----------|-----------------------------|
| Injection and Input Handling        | <b>F</b>  | 27 findings, worst critical |
| Access Control                      | <b>B</b>  | 47 findings, worst medium   |
| Authentication and Session Security | <b>C</b>  | 4 findings, worst high      |
| Transport Security                  | <b>B+</b> | 2 findings, worst low       |
| Browser Security Controls           | <b>B</b>  | 3 findings, worst medium    |
| Cross-Origin Security               | <b>A</b>  | assessed, nothing found     |
| Client-Side Supply Chain            | <b>B+</b> | 1 finding, worst low        |
| File Handling                       | <b>A</b>  | assessed, nothing found     |
| API Surface                         | <b>B</b>  | 2 findings, worst medium    |
| Information Exposure                | <b>B</b>  | 4 findings, worst medium    |
| Business Logic                      | <b>B</b>  | 2 findings, worst medium    |
| Security Programme Hygiene          | <b>A-</b> | 1 finding, worst info       |

Grading starts at A and drops one step for the worst finding (info A-, low B+, medium B, high C, critical D). Three or more findings at that severity drop it one step further. **Not assessed** means no check for that area ran. It is not a clean bill of health.

**VULNERABILITIES** 13 weaknesses, 31 affected locations**CRITICAL****SQL injection**

8 affected locations · /, /apiv1.php, /catalog.php, /category.php, /login.php, /product.php, ...

**WHAT THIS CAN LEAD TO**

Depending on the database account's privileges and the surrounding query, SQL injection can permit reading or modifying data the application never intended to expose, and in some deployments file or command access.

**FIX, ONCE, EVERYWHERE**

Replace the string-built query with a parameterised statement (prepared statement or bound parameters) so the value can never be parsed as SQL. Validating the input's format is worth doing as well, but it is not the fix on its own.

▼ GET /category.php

query string · cat

confirmed · high confidence BL-001

**CRITICAL**

**Evidence.** Poisoning cat with quote/paren characters produced a MySQL error, so the value reaches the SQL parser.

**Why this proves it.** MySQL parsed our value and reported on it. Only input that reaches the query itself can make the engine complain about syntax.

```
or</b>: Uncaught mysqli_sql_exception: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '')(" ORDER BY id'
```

**How to fix.** Replace the string-built query with a parameterised statement (prepared statement or bound parameters) so the value can never be parsed as SQL. Validating the input's format is worth doing as well, but it is not the fix on its own.

▼ GET /  
cookie · curr

confirmed · high confidence BL-002

CRITICAL

**Evidence.** Poisoning curr with quote/paren characters produced a MySQL error, so the value reaches the SQL parser.

**Why this proves it.** MySQL parsed our value and reported on it. Only input that reaches the query itself can make the engine complain about syntax.

```
or</b>: Uncaught mysqli_sql_exception: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '")' at line 1 in
```

**How to fix.** Replace the string-built query with a parameterised statement (prepared statement or bound parameters) so the value can never be parsed as SQL. Validating the input's format is worth doing as well, but it is not the fix on its own.

▼ GET /catalog.php  
cookie · curr

confirmed · high confidence BL-003

CRITICAL

**Evidence.** Poisoning curr with quote/paren characters produced a MySQL error, so the value reaches the SQL parser.

**Why this proves it.** MySQL parsed our value and reported on it. Only input that reaches the query itself can make the engine complain about syntax.

```
or</b>: Uncaught mysqli_sql_exception: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '("' at line 1 in
```

**How to fix.** Replace the string-built query with a parameterised statement (prepared statement or bound parameters) so the value can never be parsed as SQL. Validating the input's format is worth doing as well, but it is not the fix on its own.

▼ GET /category.php  
cookie · curr

confirmed · high confidence BL-004

CRITICAL

**Evidence.** Poisoning curr with quote/paren characters produced a MySQL error, so the value reaches the SQL parser.

**Why this proves it.** MySQL parsed our value and reported on it. Only input that reaches the query itself can make the engine complain about syntax.

```
or</b>: Uncaught mysqli_sql_exception: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near ''alpha)'('' at li
```

**How to fix.** Replace the string-built query with a parameterised statement (prepared statement or bound parameters) so the value can never be parsed as SQL. Validating the input's format is worth doing as well, but it is not the fix on its own.

## ▼ GET /product.php

cookie · curr

confirmed · high confidence BL-005

CRITICAL

**Evidence.** Poisoning curr with quote/paren characters produced a MySQL error, so the value reaches the SQL parser.

**Why this proves it.** MySQL parsed our value and reported on it. Only input that reaches the query itself can make the engine complain about syntax.

```
or</b>: Uncaught mysqli_sql_exception: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '""(' at line 1 in
```

**How to fix.** Replace the string-built query with a parameterised statement (prepared statement or bound parameters) so the value can never be parsed as SQL. Validating the input's format is worth doing as well, but it is not the fix on its own.

## ▼ POST /apiv1.php

JSON body · id

confirmed · high confidence BL-006

CRITICAL

**Evidence.** Poisoning id with quote/paren characters produced a MySQL error, so the value reaches the SQL parser.

**Why this proves it.** MySQL parsed our value and reported on it. Only input that reaches the query itself can make the engine complain about syntax.

```
or</b>: Uncaught mysqli_sql_exception: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near ')"'(' at line 1 in
```

**How to fix.** Replace the string-built query with a parameterised statement (prepared statement or bound parameters) so the value can never be parsed as SQL. Validating the input's format is worth doing as well, but it is not the fix on its own.

## ▼ GET /product.php

query string · id

confirmed · high confidence BL-007

CRITICAL

**Evidence.** Poisoning id with quote/paren characters produced a MySQL error, so the value reaches the SQL parser.

**Why this proves it.** MySQL parsed our value and reported on it. Only input that reaches the query itself can make the engine complain about syntax.

```
or</b>: Uncaught mysqli_sql_exception: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '""')(' at line 1 in
```

**How to fix.** Replace the string-built query with a parameterised statement (prepared statement or bound parameters) so the value can never be parsed as SQL. Validating the input's format is worth doing as well, but it is not the fix on its own.

## ▼ GET /track.php

query string · order

confirmed · high confidence BL-008

CRITICAL

**Evidence.** Poisoning order with quote/paren characters produced a MySQL error, so the value reaches the SQL parser.

**Why this proves it.** MySQL parsed our value and reported on it. Only input that reaches the query itself can make the engine complain about syntax.

```
or</b>: Uncaught mysqli_sql_exception: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '""()' at line 1 in
```

**How to fix.** Replace the string-built query with a parameterised statement (prepared statement or bound parameters) so the value can never be parsed as SQL. Validating the input's format is worth doing as well, but it is not the fix on its own.

## ▼ GET /search.php

form body · q

confirmed · high confidence BL-009

CRITICAL

**Evidence.** Poisoning q with quote/paren characters produced a MySQL error, so the value reaches the SQL parser.

**Why this proves it.** MySQL parsed our value and reported on it. Only input that reaches the query itself can make the engine complain about syntax.

```
or</b>: Uncaught mysqli_sql_exception: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near ')(%'' at line 1 in /var
```

**How to fix.** Replace the string-built query with a parameterised statement (prepared statement or bound parameters) so the value can never be parsed as SQL. Validating the input's format is worth doing as well, but it is not the fix on its own.

## ▼ GET /catalog.php

query string · sort

confirmed · high confidence BL-010

CRITICAL

**Evidence.** Poisoning sort with quote/paren characters produced a MySQL error, so the value reaches the SQL parser.

**Why this proves it.** MySQL parsed our value and reported on it. Only input that reaches the query itself can make the engine complain about syntax.

```
or</b>: Uncaught mysqli_sql_exception: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near ')'('' at line 1 in
```

**How to fix.** Replace the string-built query with a parameterised statement (prepared statement or bound parameters) so the value can never be parsed as SQL. Validating the input's format is worth doing as well, but it is not the fix on its own.

▼ POST /login.php  
form body · username

confirmed · high confidence BL-011

CRITICAL

**Evidence.** Poisoning username with quote/paren characters produced a MySQL error, so the value reaches the SQL parser.

**Why this proves it.** MySQL parsed our value and reported on it. Only input that reaches the query itself can make the engine complain about syntax.

```
or</b>: Uncaught mysqli_sql_exception: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'd445b78796c6da97badbeed'
```

**How to fix.** Replace the string-built query with a parameterised statement (prepared statement or bound parameters) so the value can never be parsed as SQL. Validating the input's format is worth doing as well, but it is not the fix on its own.

CRITICAL

## NoSQL injection

1 affected location · /members.php

### FIX, ONCE, EVERYWHERE

Reject request parameters that arrive as objects or arrays where a scalar is expected, and cast to the expected type before the value reaches the query builder.

▼ GET /members.php  
query string · user

confirmed · high confidence BL-012

CRITICAL

**Evidence.** Beyond \$eq/\$ne, 1 further operator pair(s) (in\_nin) move the result set the way their own comparison semantics predict - \$in with a one-element array means exactly what the untouched value meant, and \$nin is its complement. Meanwhile a plain array and an operator no engine implements both leave the response alone. The parameter's shape is being turned into a query document, so we choose the comparison the back-end performs.

**How to fix.** Reject request parameters that arrive as objects or arrays where a scalar is expected, and cast to the expected type before the value reaches the query builder.

CRITICAL

## XPath injection

1 affected location · /directory.php

### FIX, ONCE, EVERYWHERE

Use a pre-compiled XPath expression with variable binding instead of building the expression by string concatenation.

## ▼ GET /directory.php

query string · q

confirmed · high confidence BL-013

**CRITICAL**

**Evidence.** 3 construct(s) that only an XPath/XQuery engine parses (boolean\_fn, string\_length, node\_context) each follow their own truth value, while a control built from the same characters around a nonexistent function produces no split at all. Function-call syntax on a bare boolean is not valid SQL, so the value is being compiled into an XPath expression rather than merely reaching a parser.

**Why this proves it.** 3 constructs that xpath parses fired while the control stayed silent, at the literal\_and boundary.

**How to fix.** Use a pre-compiled XPath expression with variable binding instead of building the expression by string concatenation.

**HIGH****Credential material returned to the client**

1 affected location · /openapi.json

**FIX, ONCE, EVERYWHERE**

Remove the field from the response - no client needs it. Store passwords with argon2id, script or bcrypt at cost 10 or above, never MD5, SHA-1 or SHA-256, and migrate existing rows by re-hashing on next successful login.

## ▼ GET /openapi.json

Credential material returned to the client on /openapi.json

confirmed · confirmed confidence BL-014

**HIGH**

**Evidence.** The response carries paths./login.php.post.requestBody.content.application/x-www-form-urlencoded.schema.example.password, and its value is the password itself, in clear. Two separate failures sit on top of each other: the application can print the password, so it is not hashed at rest, and it chooses to. Anyone who reaches this response - through an access-control mistake, a log, a cache, a support screenshot - reads the credential itself, and people reuse passwords.

**How to fix.** Remove the field from the response - no client needs it. Store passwords with argon2id, script or bcrypt at cost 10 or above, never MD5, SHA-1 or SHA-256, and migrate existing rows by re-hashing on next successful login.

**MEDIUM****Page can be framed by any origin**

18 affected locations · /, /account.php, /admin.php, /cart.php, /catalog.php, /category.php, /...

**WHAT THIS CAN LEAD TO**

An attacker loads the page in a transparent frame over their own, positions their bait under its button, and the victim's click lands on YOUR control while they are logged in. Whatever that control does - a transfer, a permission grant, a deletion - is what the click does.

**FIX, ONCE, EVERYWHERE**

Set Content-Security-Policy: frame-ancestors 'none' (or an explicit allow-list) on every HTML response, and keep X-Frame-Options: DENY beside it for older clients. Note that ALLOWALL, ALLOW-FROM and any other value are not weaker policies - browsers drop them, leaving no policy at all.

▼ GET <https://sekumart.sekuripy.hr/>

confirmed · high confidence BL-015

**MEDIUM**

Page can be framed by any origin

**Evidence.** An attacker can load this page in a transparent frame over their own and have the victim click its controls. no X-Frame-Options header and no CSP frame-ancestors directive

**How to fix.** Set Content-Security-Policy: frame-ancestors 'none' (or an explicit allow-list) on every HTML response, and keep X-Frame-Options: DENY beside it for older clients. Note that ALLOWALL, ALLOW-FROM and any other value are not weaker policies - browsers drop them, leaving no policy at all.

**MEDIUM**

## Cross-site scripting

6 affected locations · /, /api.php, /help.php, /product.php, /reviews.php, /search.php

**WHAT THIS CAN LEAD TO**

Script running in the application's origin can act as the signed-in user: read the page, call the API with their session, and modify what they see.

**FIX, ONCE, EVERYWHERE**

Encode on output for the context the value lands in (HTML text, attribute, JavaScript string, URL) rather than filtering on input, and add a Content-Security-Policy that forbids inline script as a second line of defence.

## ▼ GET /product.php

query string · author

confirmed · high confidence BL-016

MEDIUM

**Evidence.** A payload placed in author was executed by the browser: loading the URL below raised a dialog containing our one-time token sksm8bhp7. Nothing about this is inferred - the code ran.

**Why this proves it.** The browser executed it and raised a dialog carrying our one-time token.

**How to fix.** Encode on output for the context the value lands in (HTML text, attribute, JavaScript string, URL) rather than filtering on input, and add a Content-Security-Policy that forbids inline script as a second line of defence.

## ▼ POST /reviews.php

form body · author

confirmed · high confidence BL-017

MEDIUM

**Evidence.** The value of author comes back inside element text, where a new tag can be opened. The characters that position needs in order to execute arrived unfiltered, so a payload placed here runs.

**Why this proves it.** The value is parsed as markup in that position, with the characters the position needs left unencoded.

```
e</i><p></p></div><div class='review'><b>emszl";<:/'`odyt</b> on <i>Quantum USB
Cable</i><p></p></div><div class='review'><b>sksemszl<<odyt</b> on <i>Quantum USB
Cable</i><p></p></div><div class='review'><b>sksemszl:/>'<";odyt</b> on <i>Quantum
USB Cable</i><p></p></div><div class='review'><b></b> on <i>Rubber Duck (Debugging
Grade)</i><p></p></div><div class='review'><b></b> on <i>Regex T-Shirt</i><p></p>
</div><div class='review'><b></b> on <i>Self-Stirring Mug</i><p></p></div><div
class='review'><b><
```

**How to fix.** Encode on output for the context the value lands in (HTML text, attribute, JavaScript string, URL) rather than filtering on input, and add a Content-Security-Policy that forbids inline script as a second line of defence.

## ▼ GET /product.php

query string · body

confirmed · high confidence BL-018

MEDIUM

**Evidence.** A payload placed in body was executed by the browser: loading the URL below raised a dialog containing our one-time token sksm2easr. Nothing about this is inferred - the code ran.

**Why this proves it.** The browser executed it and raised a dialog carrying our one-time token.

**How to fix.** Encode on output for the context the value lands in (HTML text, attribute, JavaScript string, URL) rather than filtering on input, and add a Content-Security-Policy that forbids inline script as a second line of defence.

## ▼ POST /reviews.php

form body · body

confirmed · high confidence BL-019

MEDIUM

**Evidence.** The value of body comes back inside element text, where a new tag can be opened. The characters that position needs in order to execute arrived unfiltered, so a payload placed here runs.

**Why this proves it.** The value is parsed as markup in that position, with the characters the position needs left unencoded.

```
w'><b></b> on <i>Quantum USB Cable</i><p>ilnhj">;<'msylk</p></div><div
class='review'><b></b> on <i>Quantum USB Cable</i><p>sksilnhj<>msylk</p></div><div
class='review'><b></b> on <i>Quantum USB Cable</i><p>sksilnhj:`/<>"msylk</p></div>
<div class='review'><b></b> on <i>the shop</i><p></p></div><div class='review'><b>
</b> on <i>the shop</i><p></p></div><div class='review'><b></b> on <i>the shop</i><p>
</p></div><div class='review'><b></b> on <i>the shop</i><p></p></div><div
class='review'><b></b> on <i>t
```

**How to fix.** Encode on output for the context the value lands in (HTML text, attribute, JavaScript string, URL) rather than filtering on input, and add a Content-Security-Policy that forbids inline script as a second line of defence.

## ▼ GET /api.php

query string · callback

confirmed · high confidence BL-020

MEDIUM

**Evidence.** A payload placed in callback was executed by the browser: loading the URL below raised a dialog containing our one-time token sksde7km2. Nothing about this is inferred - the code ran.

**Why this proves it.** The value is parsed as markup in that position, with the characters the position needs left unencoded.

```
cb'lfeda/<;>`'':gvunz([{"id":"1","name":"Quantum USB Cable","price":"9.99"},
{"id":"2","name":"Self-Stirring Mug","price":"14.50"},{"id":"3","name":"Regex T-
Shirt","price":"19.00"},{"id":"4","name":"The Art of Off-By-One","price":"29.95"},
{"id":"5","name":"Blinkenlights Kit","price":"42.00"},{"id":"6","name":"404
Hoodie","price":"39.00"},{"id":"7","name":"Rubber Duck (Debugging
Grade)","price":"6.50"},{"id":"8","name":"Clean Code, Dirty
Deadlines","price":"24.99"}])
```

**How to fix.** Encode on output for the context the value lands in (HTML text, attribute, JavaScript string, URL) rather than filtering on input, and add a Content-Security-Policy that forbids inline script as a second line of defence.

## ▼ GET /help.php

URL fragment · lang

confirmed · high confidence BL-021

MEDIUM

**Evidence.** The payload <sksdon> supplied through lang exists as a real element in the rendered document, so attacker-controlled markup is parsed as markup.

**Why this proves it.** The value is parsed as markup in that position, with the characters the position needs left unencoded.

```
"/help.php">Help</a></nav><main><div id="langbanner" class="langbanner">Chosen
language: frienddonfi<sksdon>zmtlp</sksdon></div><h2>Help</h2><p><b>sekumart</b> is a
deliberately vulnerable practice shop. Poi
```

**How to fix.** Encode on output for the context the value lands in (HTML text, attribute, JavaScript string, URL) rather than filtering on input, and add a Content-Security-Policy that forbids inline script as a second line of defence.

## ▼ GET /

query string · q

confirmed · high confidence BL-022

MEDIUM

**Evidence.** A payload placed in q was executed by the browser: loading the URL below raised a dialog containing our one-time token sksrrhyl8. Nothing about this is inferred - the code ran.

**Why this proves it.** The browser executed it and raised a dialog carrying our one-time token.

**How to fix.** Encode on output for the context the value lands in (HTML text, attribute, JavaScript string, URL) rather than filtering on input, and add a Content-Security-Policy that forbids inline script as a second line of defence.

## ▼ GET /search.php

form body · q

confirmed · high confidence BL-023

MEDIUM

**Evidence.** The payload <sksqom> supplied through q exists as a real element in the rendered document, so attacker-controlled markup is parsed as markup.

**Why this proves it.** The value is parsed as markup in that position, with the characters the position needs left unencoded.

```
.php'>Help</a></nav><main><div id='langbanner'></div><h2>Search</h2><p>You searched
for: <b>sksqomkb<sksqom>yrdtk</b></p><div class='grid'></div><p>no matches.</p><p
class='vsrc'>👁 <a href='/viewsour
```

**How to fix.** Encode on output for the context the value lands in (HTML text, attribute, JavaScript string, URL) rather than filtering on input, and add a Content-Security-Policy that forbids inline script as a second line of defence.

MEDIUM

## Stored / second-order input

2 affected locations · /product.php, /reviews.php

### WHAT THIS CAN LEAD TO

Stored input is the case per-request filtering misses: a value that looks harmless where it enters is rendered somewhere else entirely, often to a different user and often with different escaping.

### FIX, ONCE, EVERYWHERE

Escape on output rather than on input, in the template that renders the stored value - the encoding a value needs depends on where it lands, and the endpoint that stored it cannot know that. Treat every read of persisted data as untrusted.

#### ▼ GET /product.php

Stored / second-order input on /product.php

confirmed · high confidence BL-024

MEDIUM

**Evidence.** A marker sent to author on /reviews.php came back in the response for /product.php, which never received it. The value crosses a storage boundary, so anything that parameter is not sanitising is exposed at a different URL than the one it was submitted to - which is why per-request filtering routinely misses it. The only follow-up was a revisit of the already-known render sink; no additional payload was sent there.

**Why this proves it.** The marker went to the first URL and came back from the second, which never received it.

```
b></b> on <i>Quantum USB Cable</i><p></p></div><div class="review"><b>sksi8vko</b> on
<i>Quantum USB Cable</i><p></p></div><div class="review"><b>sks</b> on <i>Quantum USB
Cable</i><p></p></div><div c
```

**How to fix.** Escape on output rather than on input, in the template that renders the stored value - the encoding a value needs depends on where it lands, and the endpoint that stored it cannot know that. Treat every read of persisted data as untrusted.

## ▼ GET /reviews.php

Stored / second-order input on /reviews.php

confirmed · high confidence BL-025

MEDIUM

**Evidence.** A marker sent to body on /catalog.php came back in the response for /reviews.php, which never received it. The value crosses a storage boundary, so anything that parameter is not sanitising is exposed at a different URL than the one it was submitted to - which is why per-request filtering routinely misses it. The only follow-up was a revisit of the already-known render sink; no additional payload was sent there.

**Why this proves it.** The marker went to the first URL and came back from the second, which never received it.

```
"><b></b> on <i>Quantum USB Cable</i><p>data://text/plain,<?php echo 'sksrfga0vy6'; ?></p></div><div class="review"><b></b> on <i>the shop</i><p></p></div><div class="review"><b></b> on <i>the shop</i>
```

**How to fix.** Escape on output rather than on input, in the template that renders the stored value - the encoding a value needs depends on where it lands, and the endpoint that stored it cannot know that. Treat every read of persisted data as untrusted.

MEDIUM

**DOM-based scripting**

1 affected location · /help.php

**FIX, ONCE, EVERYWHERE**

Write untrusted values with `textContent` rather than `innerHTML`, or sanitise with a library such as DOMPurify at the sink.

## ▼ GET /help.php

URL fragment · lang

confirmed · high confidence BL-026

MEDIUM

**Evidence.** The page reads the URL fragment and passes it to a DOM sink: a payload placed after the `#` was executed by the browser, which produced a console message containing `skswH66m3` (no dialog appeared). The fragment never reaches the server, so no server-side filter can defend this - only the page itself can.

**How to fix.** Write untrusted values with `textContent` rather than `innerHTML`, or sanitise with a library such as DOMPurify at the sink.

MEDIUM

**Outdated platform**

1 affected location · /

**FIX, ONCE, EVERYWHERE**

Check the version against the vendor's supported releases and update. Where an update is not immediately possible, confirm with the vendor or distribution whether the specific advisories are already backported, and remove the version signal (generator tag, asset query strings) only as a secondary measure - it hides the evidence rather than the exposure.

▼ GET /

Outdated platform on /

confirmed · high confidence BL-027

**MEDIUM**

**Evidence.** PHP 8.2.30 was identified from the header banner, and matched against the vendored NVD CPE database as php. 10 advisories cover this version, including [CVE-2025-14179](#), [CVE-2026-17543](#), [CVE-2026-6722](#), [CVE-2026-7261](#) (worst CVSS 9.8). This is an exposure claim about a \*version\*, not a demonstrated vulnerability - none of these were exercised against the application, and some may not apply to this configuration.

**How to fix.** Check the version against the vendor's supported releases and update. Where an update is not immediately possible, confirm with the vendor or distribution whether the specific advisories are already backported, and remove the version signal (generator tag, asset query strings) only as a secondary measure - it hides the evidence rather than the exposure.

**LOW****Transport security**

24 affected locations · /, /account.php, /admin.php, /api.php, /apiv1.php, /cart.php, /catalog...

**WHAT THIS CAN LEAD TO**

Anything travelling in cleartext can be read and modified by anyone on the network path. For an executable resource that is arbitrary script in your origin; for a session cookie it is account takeover.

**FIX, ONCE, EVERYWHERE**

Redirect the cleartext origin with 301 before serving any content, then add Strict-Transport-Security with a long max-age and includeSubDomains. Change subresource URLs to https:// rather than relying on upgrade-insecure-requests, which is a compatibility shim and not a fix.

▼ GET <https://sekumart.sekuripy.hr/>  
Transport security

confirmed · high confidence BL-028

LOW

**Evidence.** HTTPS response does not pin the browser to TLS.

**How to fix.** Redirect the cleartext origin with 301 before serving any content, then add Strict-Transport-Security with a long max-age and includeSubDomains. Change subresource URLs to https:// rather than relying on upgrade-insecure-requests, which is a compatibility shim and not a fix.

LOW

## Open redirect

1 affected location · /go.php

### FIX, ONCE, EVERYWHERE

Redirect only to paths on this origin, or resolve the parameter against a server-side allow-list of destinations.

▼ GET </go.php>  
query string · url

confirmed · high confidence BL-029

LOW

**Evidence.** url=https://sekuscan.invalid/sks is accepted and the application sends the browser off-site (HTTP 302 Location: https://sekuscan.invalid/sks).

**Why this proves it.** The application answered with a redirect to the host we chose, so the destination is attacker-supplied.

**How to fix.** Redirect only to paths on this origin, or resolve the parameter against a server-side allow-list of destinations.

LOW

## Session handling

1 affected location · /

▼ GET /  
Session handling on /

confirmed · high confidence BL-030

LOW

**Evidence.** A session-looking cookie is set without the flags that keep it out of reach of page scripts and off plaintext connections.

**How to fix.** Add HttpOnly and Secure to the PHPSESSID cookie; keep SameSite=Lax or make it stricter if the application's flows allow.

LOW

## Weak cookie flags

1 affected location · /

▼ GET https://sekumart.sekuripy.hr/

Weak cookie flags

confirmed · high confidence BL-031

LOW

**Evidence.** Cookie (session-looking) PHPSESSID is set with: no HttpOnly, no Secure, no \_\_Host- / \_\_Secure- name prefix.

**How to fix.** Add HttpOnly and Secure to the PHPSESSID cookie; keep SameSite=Lax or make it stricter if the application's flows allow.

**NEEDS VALIDATION** 59 observed, not proven

Each of these produced a real observation that did not reach proof, or is a hypothesis this run will not test on a live application. The evidence and the exact requests are here, so confirming or dismissing one is a minute of work.

► GET /  
API surface on /

unconfirmed · medium confidence BV-001

**MEDIUM**

► GET /image.php  
query string · file

unconfirmed · medium confidence BV-002

**MEDIUM**

► GET /account.php/sks-deception-  
z2r5t8ah.css  
Web cache poisoning / deception on  
/account.php/sks-deception-z2r5t8ah.css

unconfirmed · medium confidence BV-003

**MEDIUM**

► GET /admin.php/sks-deception-  
7p6wbn24.css  
Web cache poisoning / deception on  
/admin.php/sks-deception-7p6wbn24.css

unconfirmed · medium confidence BV-004

**MEDIUM**

► GET /cart.php  
Client-trusted business value on /cart.php

unconfirmed · low confidence BV-005

**MEDIUM**

► POST /login.php  
Session handling on /login.php

unconfirmed · low confidence BV-006

**LOW**

► GET /viewsource.php  
XPath injection on /viewsource.php

unconfirmed · low confidence BV-007

**LOW**

► GET /cart.php  
Input handling on /cart.php

unconfirmed · medium confidence BV-008

**INFO**

► GET /catalog.php  
Input handling on /catalog.php

unconfirmed · medium confidence BV-009

**INFO**

► GET /sekumart.sekuripy.hr/login.php  
Input handling on /sekumart.sekuripy.hr/login.php

unconfirmed · medium confidence BV-010

**INFO**

► GET /  
Client-trusted business value on /

unconfirmed · low confidence BV-011

INFO

► GET /product.php  
Input handling on /product.php

unconfirmed · low confidence BV-012

INFO

► GET /search.php  
Input handling on /search.php

unconfirmed · low confidence BV-013

INFO

► GET /track.php  
Input handling on /track.php

unconfirmed · low confidence BV-014

INFO

► GET /cart.php  
query string · content

unconfirmed · low confidence BV-015

INFO

► GET /catalog.php  
query string · content

unconfirmed · low confidence BV-016

INFO

► GET /sekumart.sekuripy.hr/login.php  
query string · content

unconfirmed · low confidence BV-017

INFO

► GET /cart.php  
query string · dir

unconfirmed · low confidence BV-018

INFO

► GET /catalog.php  
query string · dir

unconfirmed · low confidence BV-019

INFO

► GET /sekumart.sekuripy.hr/login.php  
query string · dir

unconfirmed · low confidence BV-020

INFO

► GET /cart.php  
query string · download

unconfirmed · low confidence BV-021

INFO

► GET /catalog.php  
query string · download

unconfirmed · low confidence BV-022

INFO

► GET /sekumart.sekuripy.hr/login.php  
query string · download

unconfirmed · low confidence BV-023

INFO

► GET /cart.php  
query string · fileName

unconfirmed · low confidence BV-024

INFO

► GET /catalog.php  
query string · fileName

unconfirmed · low confidence BV-025

INFO

► GET /sekumart.sekuripy.hr/login.php  
query string · fileName

unconfirmed · low confidence BV-026

INFO

► GET /cart.php  
query string · file

unconfirmed · low confidence BV-027

INFO

► GET /catalog.php  
query string · file

unconfirmed · low confidence BV-028

INFO

► GET /sekumart.sekuripy.hr/login.php  
query string · file

unconfirmed · low confidence BV-029

INFO

► GET /cart.php  
query string · filename

unconfirmed · low confidence BV-030

INFO

► GET /catalog.php  
query string · filename

unconfirmed · low confidence BV-031

INFO

► GET /sekumart.sekuripy.hr/login.php  
query string · filename

unconfirmed · low confidence BV-032

INFO

► GET /cart.php  
query string · include

unconfirmed · low confidence BV-033

INFO

► GET /catalog.php  
query string · include

unconfirmed · low confidence BV-034

INFO

► GET /sekumart.sekuripy.hr/login.php  
query string · include

unconfirmed · low confidence BV-035

INFO

► GET /cart.php  
query string · lang

unconfirmed · low confidence BV-036

INFO

► GET /catalog.php  
query string · lang

unconfirmed · low confidence BV-037

INFO

► GET /sekumart.sekuripy.hr/login.php  
query string · lang

unconfirmed · low confidence BV-038

INFO

► GET /cart.php  
query string · load

unconfirmed · low confidence BV-039

INFO

► GET /catalog.php  
query string · load

unconfirmed · low confidence BV-040

INFO

► GET /sekumart.sekuripy.hr/login.php  
query string · load

unconfirmed · low confidence BV-041

INFO

► GET /cart.php  
query string · page

unconfirmed · low confidence BV-042

INFO

► GET /catalog.php  
query string · page

unconfirmed · low confidence BV-043

INFO

► GET /sekumart.sekuripy.hr/login.php  
query string · page

unconfirmed · low confidence BV-044

INFO

► GET /cart.php  
query string · path

unconfirmed · low confidence BV-045

INFO

► GET /catalog.php  
query string · path

unconfirmed · low confidence BV-046

INFO

► GET /sekumart.sekuripy.hr/login.php  
query string · path

unconfirmed · low confidence BV-047

INFO

► GET /cart.php  
query string · src

unconfirmed · low confidence BV-048

INFO

► GET /catalog.php  
query string · src

unconfirmed · low confidence BV-049

INFO

► GET /sekumart.sekuripy.hr/login.php  
query string · src

unconfirmed · low confidence BV-050

INFO

► GET /cart.php  
query string · target

unconfirmed · low confidence BV-051

INFO

► GET /catalog.php  
query string · target

unconfirmed · low confidence BV-052

INFO

► GET /sekumart.sekuripy.hr/login.php  
query string · target

unconfirmed · low confidence BV-053

INFO

► GET /cart.php  
query string · template

unconfirmed · low confidence BV-054

INFO

► GET /catalog.php  
query string · template

unconfirmed · low confidence BV-055

INFO

► GET /sekumart.sekuripy.hr/login.php  
query string · template

unconfirmed · low confidence BV-056

INFO

► GET /cart.php  
query string · view

unconfirmed · low confidence BV-057

INFO

► GET /catalog.php  
query string · view

unconfirmed · low confidence BV-058

INFO

► GET /sekumart.sekuripy.hr/login.php  
query string · view

unconfirmed · low confidence BV-059

INFO

**SECURITY OBSERVATIONS 6** — attack surface and hardening

Not weaknesses in themselves: things worth knowing about the surface, and hardening that is absent. Each says what would make it matter.

## ▼ GET /openapi.json

API surface on /openapi.json

confirmed · high confidence B0-001

LOW

**Evidence.** /openapi.json documents 9 paths, including methods, parameters and schemas, which removes the enumeration work an attacker would otherwise have to do.

**How to fix.** Decide deliberately whether this should be public. If not, put it behind the same authentication as the API it documents.

## ▼ GET https://sekumart.sekuripy.hr/

Client-side supply chain

confirmed · high confidence B0-002

LOW

**Evidence.** https://static.cloudflareinsights.com/beacon.min.js is loaded from static.cloudflareinsights.com with no integrity hash. Whatever that host serves runs in this origin, with this origin's cookies and DOM access.

**How to fix.** Add integrity="sha384-..." \*\*and\*\* crossorigin="anonymous" to every third-party script and stylesheet - the hash is not checked without the second attribute. Where a vendor rotates the file at a stable URL, self-host it instead: a hash cannot pin a moving target.

## ▼ GET https://sekumart.sekuripy.hr/reviews.php

Content-Security-Policy

confirmed · high confidence B0-003

INFO

**Evidence.** No CSP is served: the application has no declarative defence against injected script.

**How to fix.** Remove 'unsafe-inline' and 'unsafe-eval' from script directives and adopt per-response nonces, or hashes for the handful of inline blocks that remain. Set object-src 'none' and base-uri 'self', which have no fallback from default-src. Deploy with Report-Only first, then enforce - a policy left in Report-Only enforces nothing.

## ▼ GET https://sekumart.sekuripy.hr/

Missing security headers

confirmed · high confidence B0-004

INFO

**Evidence.** Response advertises implementation details: x-powered-by

**How to fix.** Add Content-Security-Policy, X-Content-Type-Options: nosniff, Referrer-Policy and HSTS where TLS is in use, and remove version banners such as X-Powered-By.

▼ GET <https://sekumart.sekuripy.hr/>

Security programme hygiene

confirmed · high confidence B0-005

INFO

**Evidence.** No CAA record was found for `sekumart.sekuripy.hr` or its parents. Every public certificate authority may issue for this name, so a single mis-validation at any CA produces a certificate browsers accept.

**How to fix.** Serve `/.well-known/security.txt` with Contact and Expires as RFC 9116 requires, and diary the expiry. Add a CAA record naming the authorities you actually use, plus `iodef` so attempted issuance elsewhere is reported to you.

▼ GET <https://sekumart.sekuripy.hr/>

TLS configuration

confirmed · high confidence B0-006

INFO

**Evidence.** Hostname matches, chain is trusted, 73 days remaining, negotiated TLSv1.3 with TLS\_AES\_128\_GCM\_SHA256.

**How to fix.** Disable TLS 1.0 and 1.1 and every null, export, RC4, DES and 3DES suite; require TLS 1.2 with forward-secret suites as the minimum, and prefer 1.3. Automate renewal so expiry is not an operational event.

METHOD AND COVERAGE

Parameters first receive a bounded triage echelon: ordinary values get a baseline plus an alphanumeric alternate, while integer-shaped values get a baseline, a valid alternate and an arithmetic-shaped value. Only signals earn stronger proof probes: 14736 active probes went to 164 input surfaces. Detection is differential - a claim needs a database error naming the engine, a payload the browser executed, or a condition that flips the response and then reproduces with constants the application has never seen.

How to read this report

Was this run complete?

**Coverage is incomplete — read "no findings" accordingly**

This run ended as finished.

- check recon/exposure failed during plan (ValueError)
- 1 probe target was not admitted because a soft check allocation cap was reached

| MEASURE                                | VALUE             |
|----------------------------------------|-------------------|
| Actions used                           | 15164 (unlimited) |
| Probes still queued                    | 0                 |
| Probes deferred by action allocation   | 0                 |
| Surfaces never exercised               | 0 / 164           |
| Surfaces deferred by transport dialect | 0                 |
| Targets skipped by soft check caps     | 1                 |

Active probe allocation

Queue state only: **complete** means every group that was queued by that check finished. It is not a clean-result claim. A pending row identifies work that an action limit or an interruption left behind.

| CHECK                | GROUPS<br>DONE /<br>RUNNING /<br>QUEUED /<br>DEFERRED | PROBES<br>DONE /<br>SENT /<br>QUEUED /<br>DEFERRED | QUEUE STATE |
|----------------------|-------------------------------------------------------|----------------------------------------------------|-------------|
| access-control/authz | 5 / 0 / 0 / 0                                         | 20 / 0 / 0 / 0                                     | complete    |

|                              |                 |                  |          |
|------------------------------|-----------------|------------------|----------|
| access-control/open_redirect | 22 / 0 / 0 / 0  | 154 / 0 / 0 / 0  | complete |
| auth/jwt                     | 3 / 0 / 0 / 0   | 3 / 0 / 0 / 0    | complete |
| client/dom                   | 215 / 0 / 0 / 0 | 216 / 0 / 0 / 0  | complete |
| client/dom_pollution         | 4 / 0 / 0 / 0   | 8 / 0 / 0 / 0    | complete |
| config/cors                  | 47 / 0 / 0 / 0  | 94 / 0 / 0 / 0   | complete |
| injection/cache_poisoning    | 85 / 0 / 0 / 0  | 253 / 0 / 0 / 0  | complete |
| injection/command            | 31 / 0 / 0 / 0  | 155 / 0 / 0 / 0  | complete |
| injection/graphql            | 1 / 0 / 0 / 0   | 10 / 0 / 0 / 0   | complete |
| injection/header_injection   | 3 / 0 / 0 / 0   | 12 / 0 / 0 / 0   | complete |
| injection/log4shell          | 40 / 0 / 0 / 0  | 120 / 0 / 0 / 0  | complete |
| injection/nosqli             | 30 / 0 / 0 / 0  | 155 / 0 / 0 / 0  | complete |
| injection/sqli               | 316 / 0 / 0 / 0 | 3175 / 0 / 0 / 0 | complete |
| injection/ssrf               | 58 / 0 / 0 / 0  | 261 / 0 / 0 / 0  | complete |
| injection/ssti               | 30 / 0 / 0 / 0  | 60 / 0 / 0 / 0   | complete |
| injection/struts_ognl        | 20 / 0 / 0 / 0  | 40 / 0 / 0 / 0   | complete |
| injection/traversal          | 632 / 0 / 0 / 0 | 7263 / 0 / 0 / 0 | complete |
| injection/wp_batch_sqli      | 1 / 0 / 0 / 0   | 5 / 0 / 0 / 0    | complete |
| injection/xmlrpc_pingback    | 1 / 0 / 0 / 0   | 1 / 0 / 0 / 0    | complete |
| injection/xpath              | 28 / 0 / 0 / 0  | 566 / 0 / 0 / 0  | complete |
| injection/xslt               | 3 / 0 / 0 / 0   | 6 / 0 / 0 / 0    | complete |
| injection/xss                | 30 / 0 / 0 / 0  | 131 / 0 / 0 / 0  | complete |
| recon/iis_shortname          | 1 / 0 / 0 / 0   | 3 / 0 / 0 / 0    | complete |
| recon/param_mining           | 83 / 0 / 0 / 0  | 1196 / 0 / 0 / 0 | complete |
| recon/triage                 | 271 / 0 / 0 / 0 | 829 / 0 / 0 / 0  | complete |

### Targets skipped by soft allocation caps

These are deliberate request/precision controls, not clean results. The affected targets were not admitted to a proof group.

| CHECK                | STAGE   | PATH / PARAMETER | LIMIT | OCCURRENCES |
|----------------------|---------|------------------|-------|-------------|
| client/dom_pollution | pollute |                  | 4     | 1           |

Checks that could not run:

- business-logic/workflow
- business\_logic/race
- injection/amplification
- injection/blind\_xss
- injection/drupal\_render
- injection/gitlab\_exiftool
- injection/hql
- injection/proto\_pollution
- injection/smuggling
- injection/xxe
- upload/file\_upload
- upload/unauth\_connector

## Software inventory 1 client-side component, CycloneDX 1.6

Generated from script assets the browser loaded, not from a manifest. That catches the CDN tag nobody declared, and it cannot see packages bundled into a single artefact. The machine-readable document is the sbom object in this report's JSON.

| COMPONENT | VERSION | IDENTIFIER                                              |
|-----------|---------|---------------------------------------------------------|
| PHP       | 8.2.30  | pkg:generic/php@8.2.30<br>https://sekumart.sekuripy.hr/ |

## Standards coverage

Every control below is one this scan produced evidence about. A scanner does not grant conformance with a standard — the verb per framework says exactly what the mapping claims, and controls this scan did not reach are shown greyed rather than omitted.

| FRAMEWORK                                                                 | ASSESSED                                        | CONTROLS                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---------------------------------------------------------------------------|-------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>OWASP Top 10</b><br>2025<br>assessed at<br>category level              | <b>9 / 9</b><br>9 with findings<br>5 partial    | <div>A01:2025</div> <div>A02:2025</div> <div>A03:2025</div> <div>A04:2025</div> <div>A05:2025</div> <div>A06:2025</div> <div>A07:2025</div> <div>A08:2025</div> <div>A10:2025</div>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>OWASP API Security Top 10</b><br>2023<br>assessed at<br>category level | <b>7 / 7</b><br>3 with findings<br>1 partial    | <div>API1:2023</div> <div>API2:2023</div> <div>API5:2023</div> <div>API6:2023</div> <div>API7:2023</div> <div>API8:2023</div> <div>API9:2023</div>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>OWASP ASVS</b><br>5.0.0<br>mapped at<br>chapter level                  | <b>12 / 12</b><br>11 with findings<br>7 partial | <div>V1</div> <div>V2</div> <div>V3</div> <div>V4</div> <div>V5</div> <div>V7</div> <div>V8</div> <div>V9</div> <div>V10</div> <div>V11</div> <div>V12</div> <div>V13</div>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <b>OWASP WSTG</b><br>4.2<br>assessed at test<br>level                     | <b>32 / 33</b><br>18 with findings<br>5 partial | <div>WSTG-APIT-01</div> <div>WSTG-ATHZ-01</div> <div>WSTG-ATHZ-02</div> <div>WSTG-ATHZ-03</div> <div>WSTG-ATHZ-04</div> <div>WSTG-BUSL-09</div> <div>WSTG-CLNT-01</div> <div>WSTG-CLNT-04</div> <div>WSTG-CLNT-05</div> <div>WSTG-CLNT-07</div> <div>WSTG-CONF-02</div> <div>WSTG-CONF-04</div> <div>WSTG-CONF-05</div> <div>WSTG-CONF-07</div> <div>WSTG-CONF-12</div> <div>WSTG-CRYP-01</div> <div>WSTG-CRYP-03</div> <div>WSTG-ERRH-01</div> <div>WSTG-INFO-02</div> <div>WSTG-INFO-08</div> <div>WSTG-INPV-01</div> <div>WSTG-INPV-02</div> <div>WSTG-INPV-05</div> <div>WSTG-INPV-06</div> <div>WSTG-INPV-07</div> <div>WSTG-INPV-09</div> <div>WSTG-INPV-12</div> <div>WSTG-INPV-15</div> <div>WSTG-INPV-18</div> <div>WSTG-INPV-19</div> <div>WSTG-SESS-02</div> <div>WSTG-SESS-05</div> <div>WSTG-SESS-10</div> |

|                                                                         |                                                 |                                                                                                             |
|-------------------------------------------------------------------------|-------------------------------------------------|-------------------------------------------------------------------------------------------------------------|
| <b>ISO/IEC 27001</b><br>2022<br>supports at<br>Annex A control<br>level | <b>12 / 13</b><br>11 with findings<br>8 partial | <div>A.5.15 A.5.21 A.5.24 A.8.2 A.8.3 A.8.5 A.8.8 A.8.9 A.8.20</div> <div>A.8.24 A.8.26 A.8.28 A.8.29</div> |
| <b>NIST CSF 2.0</b><br>supports at<br>subcategory<br>level              | <b>6 / 6</b><br>6 with findings<br>3 partial    | <div>ID.AM-02 ID.RA-01 PR.AA-05 PR.DS-02 PR.PS-01 PR.PS-02</div>                                            |
| <b>PCI DSS 4.0.1</b><br>supports at<br>requirement<br>level             | <b>8 / 9</b><br>8 with findings<br>4 partial    | <div>4.2.1 6.2.4 6.3.1 6.3.3 6.4.1 6.4.3 7.2.1 8.3.1 11.3.2</div>                                           |

Identifiers reviewed 2026-08-22. The mapped subset is validated by tools/refresh\_standards.py against the official machine-readable ASVS 5.0.0 and Top 10:2025 sources. Unmapped official controls are not assessed here.

### Requests, by what they were for

| TRAFFIC                    | REQUESTS     | WHAT IT IS                                                                                                                   |
|----------------------------|--------------|------------------------------------------------------------------------------------------------------------------------------|
| Active security probes     | <b>14736</b> | Requests this scanner composed to test a parameter. Issued outside the page, so they do not appear in the browser log below. |
| Page loads                 | <b>737</b>   | Documents the browser navigated to while mapping the application.                                                            |
| In-scope assets            | <b>1267</b>  | Scripts, stylesheets and images the pages themselves pulled from this origin.                                                |
| Third-party requests       | <b>1265</b>  | Traffic your own pages send elsewhere - analytics, telemetry, CDNs. Not initiated by this assessment.                        |
| <b>Total HTTP requests</b> | <b>18005</b> | Everything above: 3269 from the browser plus 14736 probes.                                                                   |

### What was reached

|                    |                                                                                                                                  |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------|
| pages rendered     | 373 page states over 43 distinct paths                                                                                           |
| input surfaces     | 164 tested (164 exercised, 110 produced a signal)                                                                                |
| forms              | 10 seen, 9 submitted                                                                                                             |
| application errors | 8 paths returned an interpreter error in the response body. Those arrive with HTTP 200, so they are not in the 0 HTTP 5xx count. |
| stopped because    | agent-finished                                                                                                                   |

**How severity was assigned** and which taxonomy version

**Scope, exclusions and scan metadata** what was in bounds, and the run's own numbers

## APPENDIX

**Tested, nothing observed** 137 results that were checked and came back clean

**Attack surface** 164 surfaces · 164 exercised · 110 flagged

**Forms and inputs** 10 forms

**Endpoints observed** 52 endpoints · 1 returning JSON

**Pages reached** 373 states · 672 transitions

**Checks applied** 78 modules · 57 send probes

**Browser console** 124 events

**Every raw signal** 536 before consolidation

**Run detail** identifiers, versions, markers

## REFERENCES

Every catalogue and standard this report cites, and nothing else. A reference list longer than the document's own citations is decoration.

## Published vulnerabilities

- [CVE-2025-14179](#) National Vulnerability Database
- [CVE-2026-14355](#) National Vulnerability Database
- [CVE-2026-17543](#) National Vulnerability Database
- [CVE-2026-6722](#) National Vulnerability Database
- [CVE-2026-6735](#) National Vulnerability Database
- [CVE-2026-7259](#) National Vulnerability Database
- [CVE-2026-7260](#) National Vulnerability Database
- [CVE-2026-7261](#) National Vulnerability Database
- [CVE-2026-7262](#) National Vulnerability Database
- [CVE-2026-7568](#) National Vulnerability Database

## Weakness types

- [CWE-22](#) traversal
- [CWE-79](#) dom, second\_order, xss
- [CWE-89](#) sql
- [CWE-200](#) graphql, rest
- [CWE-209](#) errors
- [CWE-256](#) credential\_storage
- [CWE-319](#) transport
- [CWE-326](#) tls
- [CWE-352](#) session
- [CWE-384](#) session
- [CWE-441](#) cache\_poisoning
- [CWE-601](#) open\_redirect
- [CWE-643](#) xpath
- [CWE-693](#) csp, headers
- [CWE-829](#) supplychain
- [CWE-840](#) workflow
- [CWE-943](#) nosqli
- [CWE-1004](#) cookies
- [CWE-1021](#) clickjacking
- [CWE-1104](#) platform

## Standards and frameworks

- [OWASP Top 10](#) version 2025 — mapped at category level
- [OWASP API Security Top 10](#) version 2023 — mapped at category level
- [OWASP ASVS](#) version 5.0.0 — mapped at chapter level
- [OWASP WSTG](#) version 4.2 — mapped at test level
- [ISO/IEC 27001](#) version 2022 — mapped at Annex A control level
- [NIST CSF](#) version 2.0 — mapped at subcategory level
- [PCI DSS](#) version 4.0.1 — mapped at requirement level

A mapping says which control an observation is relevant to. It is not a statement of compliance, which needs scope, evidence and an assessor this report does not stand in for.