SAMPLE REPORT

Application security assessment

This is a real report from a real audit, published so you can see how we write a finding before you decide whether to hire us. It is the same format your report will arrive in.

Application	www.technotackle.com, our own website
Assessment type	White-box source code and configuration review
Standard	OWASP Top 10:2025, CVSS v3.1, CWE
Testing period	August 2026
Remediation verified	17 September 2026
Security auditor	Naveenkumar S
AI lead	Iyyappan Palanivel, AI/ML Team Lead and Lead Software Engineer
Prepared by	Techno Tackle Software Solutions, Coimbatore, India
Document date	22 September 2026

How to read this report

We audited our own website before we sold this service to anybody, on the reasoning that a security firm should be willing to publish what was wrong with its own systems. This document contains two of the findings from that audit, written up exactly as your findings would be.

> **Two things we have deliberately held back.** First, live identifiers. Parameter names, file paths, endpoint names, table names and working exploit strings do not appear here, and they will not appear in any public document about your systems either. The code samples use generic equivalents that show the same bug and the same fix. Your own full report, delivered privately under NDA, carries the real ones. Second, this audit produced findings beyond the two below. Any finding whose remediation is still in progress stays unpublished until it is closed and verified.

Every finding in a Techno Tackle report follows the same six-part structure: what the issue was in one plain line, how we found it, what an attacker could have done with it, the classification, the time the fix took, and the before and after code. If we cannot demonstrate an issue, we do not report it.

Severity scoring

Findings are scored with CVSS v3.1 base metrics on a 0.0 to 10.0 scale across exploitability and impact, giving four tiers: Critical 9.0 to 10.0, High 7.0 to 8.9, Medium 4.0 to 6.9, and Low 0.1 to 3.9. Every finding is cross-referenced against the OWASP Top 10:2025 classification and mapped to a specific Common Weakness Enumeration (CWE) identifier.

Scope of assessment

In scope. A white-box source code and configuration review of the development branch across the full application surface: the public web portal, the administrative interfaces, the content publishing endpoints and the backend data access layer. Areas evaluated included secret management, the use of prepared statements for database queries, output sanitisation against

cross-site scripting, cross-site request forgery defences, form rate limiting, and web server routing rules.

Out of scope. Third-party hosted systems and their APIs, including mail delivery and CAPTCHA verification services. Host operating system and DNS configuration. Network-layer volumetric denial of service resilience. Physical infrastructure controls. Employee security awareness and social engineering exercises.

What we could not test

- **Static analysis limits.** This was primarily a source code review, so runtime-only behaviour such as race conditions and memory exhaustion under concurrent load could not be evaluated by inspection alone.
- **Non-destructive database testing.** Destructive payloads, schema-altering queries and table drops were deliberately excluded to protect data integrity and uptime. Checks were limited to query construction and parameter binding.
- **Environment parity.** Testing was isolated to the development branch and staging. Production-specific web server variations and any uncommitted manual server overrides could not be verified dynamically.
- **Denial of service and external fuzzing.** Volumetric resource-exhaustion tests and active fuzzing of third-party API dependencies were omitted to avoid disrupting shared hosting infrastructure.

Findings in this report

#	FINDING	SEVERITY	OWASP TOP 10:2025	STATUS
1	A pagination number went straight into a database query	Critical 9.8	A05 Injection CWE-89	Closed
2	Any website could submit our forms on a visitor's behalf	High 8.0	A01 Broken Access Control CWE-352, CWE-942	Closed

Both findings were remediated, deployed to production and retested. Neither remains open.

FINDING 1

A pagination number went straight into a database query

Critical · CVSS 9.8 | Closed

A value that arrived from the browser, controlling which page of results to show, was pasted into the text of a database query instead of being passed to it as a parameter. Anyone could replace that value with a fragment of database query of their own and have the database run it.

How we found it	No scanner flagged this. An engineer read the request handler line by line and followed every value arriving from the browser to see where it ended up. One pagination number was being concatenated into a query string a few lines away from code in the same file that handled its values correctly. A second pass confirmed the same pattern in three further query blocks and in one other page that built a filter clause the same way.
What an attacker could have done	Read the entire database with a single unauthenticated request, including every record submitted through the site's forms. A payload as short as `offset=1 UNION SELECT null, null, null--` is enough to begin mapping the schema, after which the column count can be matched and real data returned. Where the database account holds write permissions the same hole allows records to be altered, content tables to be seeded with scripts that then run in an administrator's browser, or tables to be dropped outright. No login, no session and no specialist tooling are required, and no filtering rules were in place to stop it.
Classification	Injection. OWASP Top 10:2025 A05. CWE-89, improper neutralisation of special elements used in an SQL command.
Time to fix	Eight hours. Most of that was auditing every other query in the application for the same pattern rather than repairing the one we found.

Fixed and verified	Deployed to production 17 September 2026 and retested after deployment.

BEFORE

```
$offset = $_POST['offset'];
$sql   .= " ORDER BY id DESC LIMIT $offset, $limit";
$result = $conn->query($sql);
```

AFTER

```
$offset = isset($_POST['offset']) ? intval($_POST['offset']) : 0;
$sql   .= " WHERE is_active = 1 AND category_id = ?";
$sql   .= " ORDER BY id DESC LIMIT ?, ?";
$stmt   = $conn->prepare($sql);
$stmt->bind_param("iii", $category, $offset, $limit);
```

Why this fix and not a filter

A common response to this class of bug is to strip dangerous-looking characters from the input. That approach fails as soon as someone finds an encoding the filter does not recognise. The fix above removes the possibility rather than the symptom: the pagination value is forced to an integer, and every remaining value reaches the database as a bound parameter, which the database treats as data and never as instructions. A regression test covers the endpoint so the pattern cannot return unnoticed.

FINDING 2

Any website could submit our forms on a visitor's behalf

High · CVSS 8.0 Closed

No form on the site checked that a submission had actually come from the site, and the server told every browser that requests from any website on the internet were acceptable. Together those two gaps meant a page belonging to someone else could submit our forms using a visitor's browser, without the visitor noticing.

How we found it	An architectural review of the form handling pipeline. We read every POST endpoint in the central request handler and every form template in the application, and confirmed there were no anti-forgery tokens, no nonces and no origin validation anywhere: nothing generated a token, no form carried one, and no handler checked for one. We then examined the server's response headers and found a wildcard cross-origin policy being sent on every response. Either issue alone is serious. Together they remove the browser's own protection against this attack for every form on the site.
What an attacker could have done	Publish a page that silently submits our forms whenever anybody visits it. The consequences run from flooding the database and the sales and recruitment inboxes with junk, through to using the file upload path on the careers form to place an executable file on the server, which can lead to the attacker running commands there. Offensive or defamatory content could be pushed through the contact form and would arrive looking like a genuine enquiry. With no rate limiting in place, submissions can be automated at volume until the server slows for real visitors. Because the requests are made by the visitor's own browser and carry their address, tracing the source is harder than it looks.

Classification	Broken access control. OWASP Top 10:2025 A01. CWE-352, cross-site request forgery, and CWE-942, overly permissive cross-domain whitelist.
Time to fix	Seven hours. Six to implement token generation, embedding and verification across every form, one to replace the wildcard cross-origin policy.
Fixed and verified	Deployed to production 17 September 2026 and retested after deployment.

BEFORE

```
// every response, any origin accepted
header('Access-Control-Allow-Origin: *');
// form handler, no origin or token check
if (isset($_POST['form_submit'])) {
    $data['name'] = $_POST['name'];   // processed immediately
```

AFTER

```
// only our own origins are answered
$allowed = ['https://example.com', 'https://www.example.com'];
if (!empty($origin) && in_array($origin, $allowed, true)) {
    header('Access-Control-Allow-Origin: ' . $origin);
    header('Vary: Origin');
}
// token checked before any form is processed
function validateRequestCsrf() {
    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        if (!verifyCsrfToken($_POST['csrf_token'] ?? '')) {
            http_response_code(403);
            exit(json_encode(['status' => false, 'message' => 'Invalid CSRF
token.']));
        }
    }
}
validateRequestCsrf();
```

Why both halves were needed

Adding tokens without fixing the cross-origin policy leaves the server advertising that it will answer anyone, which weakens other defences later. Fixing the policy without adding tokens leaves the forms open, because a form submitted the ordinary way does not depend on that policy at all. The two changes shipped together, and the token check runs before any form handler, so a new form added in future is covered by default rather than by somebody remembering.

Sign-off

Security auditor	Naveenkumar S
AI lead	Iyyappan Palanivel, AI/ML Team Lead and Lead Software Engineer
Application	www.technotackle.com
Testing period	August 2026
Remediation verified	17 September 2026
Standards applied	OWASP Top 10:2025, CVSS v3.1, CWE

What your own report would contain

The same structure, with the real identifiers in place: the actual parameter, file, endpoint and table names, the exact requests that triggered each finding, and screenshots or logs as evidence. It arrives under NDA, along with a walkthrough call with the engineer who did the testing and an attestation letter you can forward to your own customers.

Findings are retested until they are verifiably closed, at no extra charge, and it makes no difference whether your developers wrote the fix or ours did.

Techno Tackle Software Solutions · Site No 15, Krishna Havisha Building, SA Garden, Muthuswamy Colony, Selvapuram, Coimbatore 641026, Tamil Nadu, India
+91 86376 46693 · hello@technotackle.com · www.technotackle.com
ISO/IEC 27001:2022 certified. We are not CERT-In empanelled, and we say so before you ask.