XYNY / SECURITY RESEARCH PORTFOLIO

LAB / 002

SQL Injection

Retrieval of hidden catalogue data

AUTHORIZED TRAINING LAB / PORTSWIGGER WEB SECURITY ACADEMY

STATUS

SOLVED

CLASSIFICATION

CWE-89

RISK LEVEL

HIGH*

TEST DATE

21 OCT 2026

*Qualitative severity assigned for the intentionally vulnerable lab; not a formal CVSS rating.

Demonstration only. No customer infrastructure, company production systems, or third-party data were assessed.

AUTHOR / XYNY SECURITY · DEMONSTRATION CASE STUDY

01 / SCOPE AND METHOD

Engagement & methodology

AUTHORIZED / LAB SIMULATION

Scope of examination

PortSwigger Web Security Academy: SQL injection vulnerability in WHERE clause allowing retrieval of hidden data. Only the transient training lab instance supplied for this exercise was used.

Objective

Determine whether the category filter permits a crafted input to alter its server-side database selection logic and expose hidden catalogue records.

Observed workflow

01 **Baseline**

Observed product category filtering on the sample storefront.

02 **Input surface**

Identified the category query-string parameter in /filter?category=.

03 **Controlled test**

Submitted one boolean-condition payload to the authorized lab.

04 **Outcome**

Lab banner displayed Solved; catalogue expanded in the response.

02 / FINDING AND REMEDIATION

Validated SQL injection condition

CWE-89 / SQL INJECTION

SIMULATED SEVERITY: HIGH

Affected entry point

GET /filter?category=...

Test input observed on successful lab page

```
Lifestyle' OR 1=1--
```

Evidence-based result

The training application responded with catalogue entries beyond the requested category, and PortSwigger displayed the Solved status. This demonstrates a filter bypass on the lab dataset; it does not, on its own, demonstrate credential extraction or system takeover.

Impact in a production-like application

Where similarly unsafe queries exist, attackers may gain access to data outside intended views. Exact business impact depends on database permissions, application logic, and available records.

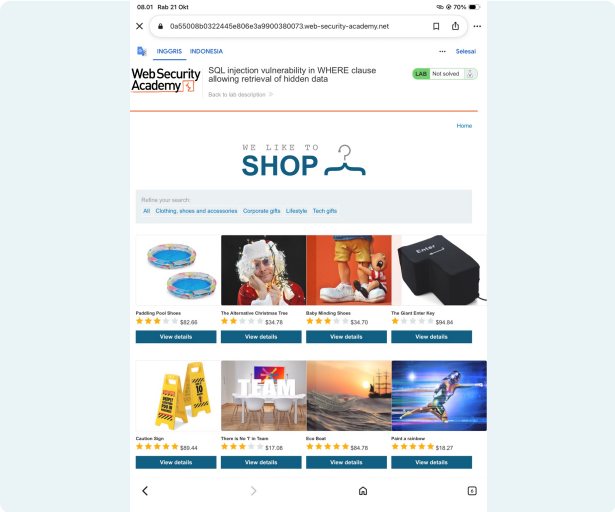
Recommendations

- Use parameterized queries / prepared statements for every user-supplied value.
- Avoid string-concatenated SQL; restrict database account privileges.
- Implement regression tests for hostile input and review error disclosures.

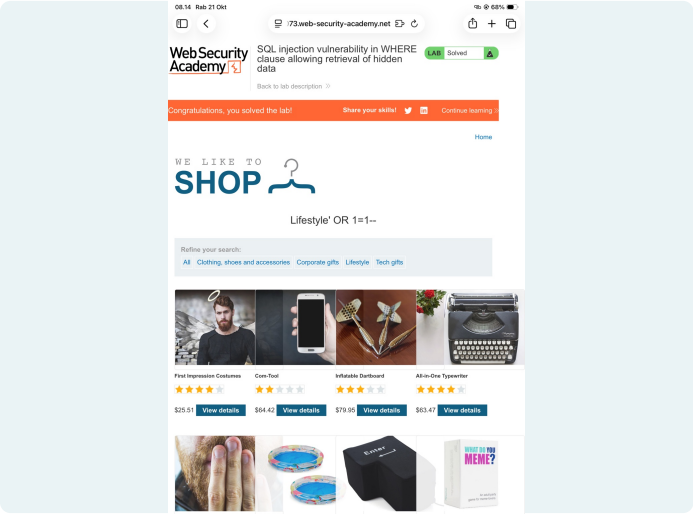
03 / SCREENSHOTS AND LIMITATIONS

Evidence & verification

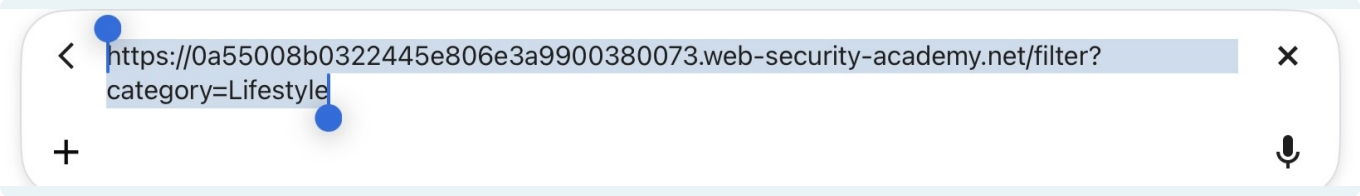
EVIDENCE 01 / INITIAL LAB VIEW



EVIDENCE 03 / LAB SOLVED



EVIDENCE 02 / OBSERVED PARAMETER



Verification and limits

Status Solved is visible in the training-lab screenshot with the chosen category expression displayed above the returned products. This report documents one intentionally vulnerable PortSwigger instance and should not be represented as independent third-party penetration testing experience.

No retest after remediation was performed because no production fix was in scope. The lab is a teaching exercise, not a customer engagement. Source: PortSwigger Web Security Academy, "SQL injection vulnerability in WHERE clause allowing retrieval of hidden data".

SOURCE / PORTSWIGGER

https://portswigger.net/web-security/sql-injection/lab-retrieve-hidden-data