AKSHAT PARIKH

CYBERSECURITY CONSULTANT

Web Application Penetration Test Report

For [Company Name]

Prepared For: [Company Name]

Company Website: http://store.company.com/

Prepared By: Akshat Parikh

Akshat's Website: https://akshatsecurity.com

Date: September 23, 2025

Version: 1.0

Table of Contents

1.0 Introduction and Scope

1.1 Confidentiality Statement

This document contains confidential information about the security posture of [Company Name]. The content of this report is intended solely for the use of [Company Name] and its designated representatives. Unauthorized disclosure or distribution of this report is strictly prohibited.

1.2 Contact Information

Name	Title	Contact Information
Assessment Team		
Akshat Parikh	Lead Penetration Tester	Email: contact@gmail.com Phone: [Phone Number]
Client Team		
[Client Name]	[Client Role]	Email: [Client Email] Phone: [Client Phone]

1.3 Assessment Overview

Akshat Parikh was engaged by [Company Name] to perform a web application penetration test of their e-commerce platform. The assessment covered the primary storefront hosted at https://store.company.com, as well as associated subdomains and key application directories, to provide a comprehensive evaluation of the platform's security. The assessment was conducted from 28/09/2025 to 05/10/2025. The goal was to identify security vulnerabilities that could be exploited by a malicious actor to compromise customer data, disrupt business operations, and to provide actionable recommendations for remediation.

1.4 Scope and Exclusions

Scope

The scope of this assessment was limited to the following assets, accessible via the public internet:

- The primary e-commerce application: https://store.company.com
- Associated subdomains, including but not limited to: api.store.company.com and blog.store.company.com.
- Key application directories and functionality, such as the customer account portal (/account/*) and the administrative login interface (/admin)

Scope Exclusions

Per client request, Akshat Parikh did not perform any of the following attacks during testing:

- Denial of Service (DoS) attacks
- Social Engineering or Phishing campaigns
- Physical security assessments

Any internal systems, corporate networks, or third-party services not explicitly listed above were also considered out of scope.

Client Allowances

[Company Name] did not provide any credentials, source code, or special access to assist the testing. The assessment was performed from an unauthenticated, black-box perspective.

2.0 Executive Summary

2.1 Assessment Summary

This report outlines the findings of the web application penetration test conducted against the https://store.company.com e-commerce platform. The assessment simulated the actions of a motivated attacker and identified several high-impact vulnerabilities that present a direct and immediate threat to [Company Name]'s business operations and customer data. The most significant of these is a Critical SQL Injection vulnerability that could, if exploited, result in a complete compromise of the customer database. Additionally, several High severity flaws were discovered that could lead to customer account takeovers and the leakage of private order information.

2.2 Key Findings and Recommendations

Severity	Vulnerability	Business Impact
Critical	SQL Injection	Complete compromise of customer database, leading to massive data breach, loss of customer trust, and potential regulatory fines.
High	Cross-Site Scripting (XSS)	Attackers can hijack customer sessions, steal personal information, and perform fraudulent activities, leading to reputational damage.
High	Insecure Direct Object References (IDOR)	Customer privacy violation by allowing unauthorized access to other users' order details and personal addresses.
Medium	Weak Password Policy	Increases the likelihood of customer account takeovers through simple password guessing, which could lead to fraudulent orders and chargebacks.
Low	Software Version Disclosure	Leaks technical details that help attackers plan more targeted and effective attacks against the website in the future.

2.3 Overall Risk Rating

The overall risk to the store.company.com platform is rated as CRITICAL. The presence of a vulnerability that allows for a full database compromise presents an immediate and severe threat to the business. Swift remediation is essential to protect customer data and the company's reputation.

2.4 Strategic Recommendation

To mitigate the identified risks and improve the overall security posture, we strongly recommend that [Company Name] take the following actions:

1. **Immediate Remediation:** Prioritize the remediation of all Critical and High severity findings within the next 7-14 days to address the most immediate threats.
2. **Secure Coding Training:** Invest in secure coding training for the development team, with a focus on preventing the OWASP Top 10 vulnerabilities.
3. **Implement a Vulnerability Management Program:** Establish a formal process for regularly identifying, assessing, and remediating security vulnerabilities. This should include commissioning annual, independent penetration tests.

3.0 Methodology

The penetration test was conducted using a systematic and phased approach aligned with industry-standard frameworks, including the Open Web Application Security Project (OWASP) Web Security Testing Guide (WSTG).

3.1 Planning and Reconnaissance

This initial phase involved gathering information about the target application to understand its functionality and potential attack surface. This included:

- **Public Information Gathering:** Reviewing publicly available information about [Company Name] and their web application.
- **Application Discovery:** Mapping the application's structure, identifying pages, and understanding the technologies in use.

3.2 Vulnerability Analysis

In this phase, both automated and manual techniques were used to identify potential vulnerabilities. This included:

- **Automated Scanning:** Utilizing industry-standard web application security scanners to identify common vulnerabilities.
- **Manual Testing:** Manually probing the application for vulnerabilities that automated scanners may miss, such as business logic flaws.

3.3 Exploitation

Once potential vulnerabilities were identified, attempts were made to exploit them to confirm their existence and determine their impact. This was done in a controlled and non-disruptive manner.

3.4 Reporting

All findings were documented in this report, including detailed descriptions of the vulnerabilities, their potential impact, and recommendations for remediation.

4.0 Findings

4.1 Finding Severity Ratings

The vulnerabilities identified during this assessment are rated according to the following severity levels:

- **Critical:** Vulnerabilities that could lead to a complete system compromise, major data breach, or significant service disruption.
- **High:** Vulnerabilities that could lead to a significant compromise of data or resources, but not a full system compromise.
- **Medium:** Vulnerabilities that could provide an attacker with useful information or limited access to the system.
- **Low:** Vulnerabilities that pose a minor risk to the system.
- **Informational:** Findings that are not necessarily vulnerabilities but are worth noting for security best practices.

4.2 Vulnerability Details

4.2.1 SQL Injection (Critical)

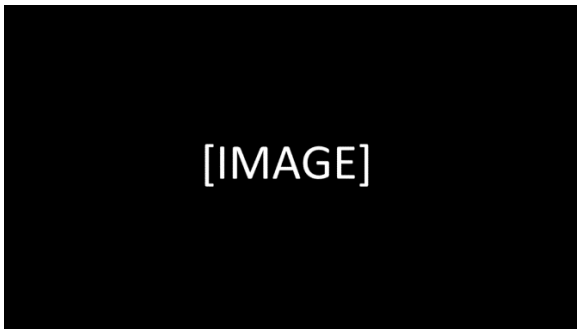
Severity:	**Critical** (CVSSv3.1 score: 9.8)
Description:	The "Sort by Price" feature on product category pages is vulnerable to SQL Injection. The application improperly trusts user-supplied input and directly includes it in a database query.
Business Risk:	**Likelihood:** High - Automated tools can easily discover and exploit this flaw on public-facing features without requiring authentication. **Impact:** Very High - Successful exploitation leads to a complete compromise of all customer data, including names, addresses, and order histories, as well as potential control over the backend server.
Affected URL:	https://store.company.com/category?sort=price_asc

Steps to Reproduce

1. Navigate to any product category page, for example: https://store.company.com/category?id=tshirts.
2. Click the dropdown to sort items and select "Sort by Price: Low to High."
3. Intercept this request using a web proxy (like Burp Suite) and modify the sort parameter in the URL.
4. Change the value of the sort parameter to a time-based SQL injection payload like ' AND (SELECT 42 FROM (SELECT(SLEEP(5)))b). The full URL would look like:

https://store.company.com/category?sort=' AND (SELECT 42 FROM (SELECT(SLEEP(5)))b)

5. Send the modified request and observe that the server's response is delayed by 5 seconds, confirming that the database executed the injected command.
6. This vulnerability can be further exploited with tools like SQLMap to extract database contents.



[Image: Screenshot of Burp Suite request showing the modified parameter and payload]

Remediation

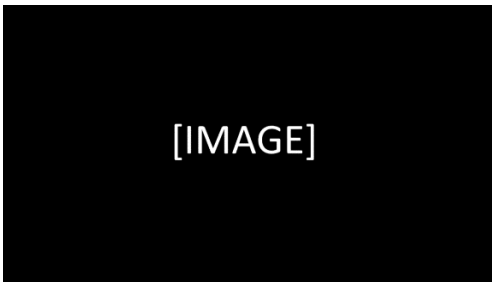
The development team must immediately stop using string concatenation to build database queries. All database queries must be rewritten to use **parameterized queries (also known as prepared statements).** This is a standard secure coding practice that treats user input as data only, never as executable code, completely mitigating this risk.

4.2.2 Stored Cross-Site Scripting (XSS) (High)

Severity	**High** (CVSSv3.1 score: 8.7)
Description	The "Product Review" feature is vulnerable to Stored XSS because it does not sanitize user-submitted text before storing and displaying it.
Business Risk	**Likelihood:** High – The vulnerability is in a feature accessible to any logged-in user, and common XSS payloads are well-known and easy to deploy. **Impact:** High – Can lead to widespread customer account takeovers, fraudulent activity performed on behalf of users, and significant reputational damage.
Affected URL	https://store.company.com/products/sample-product

Steps to Reproduce

1. Log in as any registered user on https://store.company.com.
2. Navigate to any product page, such as https://store.company.com/products/sample-product.
3. Scroll down to the "Write a Review" section.
4. In the review text box, paste the following basic XSS payload: <script>alert('XSS')</script>.
5. Submit the review.
6. Log out and navigate back to the same product page as a guest. Observe that an alert box appears, confirming the script was stored and is executed for all visitors.



[Image: Screenshot of the alert box appearing on the product page]

Remediation

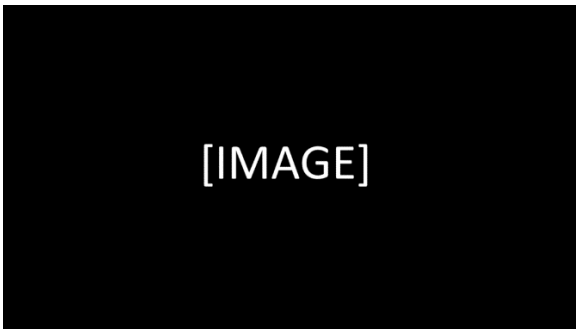
All user-supplied input must be properly sanitized using a standard library before it is rendered on the page. Specifically, characters like <, >, and " should be converted to their HTML entity equivalents (e.g., <, >, "). Implementing a Content Security Policy (CSP) is also a highly effective defence-in-depth measure.

4.2.3 Insecure Direct Object References (IDOR) (High)

Severity	**High** (CVSSv3.1 score: 8.1)
Description	The application fails to properly enforce access control on the order history page, allowing any authenticated user to view the orders of other customers.
Business Risk	**Likelihood:** High – The vulnerability can be easily exploited by any authenticated user by simply changing a number in the URL. This requires no special tools. **Impact:** High – Results in a significant breach of customer privacy, exposing sensitive order details and personal information which could lead to loss of customer trust.
Affected URL	https://store.company.com/orders?order_id=12345

Steps to Reproduce

1. Using two different browsers, log in as two different users (User A and User B).
2. As User A, place an order and navigate to the order confirmation page. Note the URL, for example: https://store.company.com/orders?order_id=1001.
3. As User B, who should not have access to User A's order, manually browse to the same URL: https://store.company.com/orders?order_id=1001.
4. Observe that the application incorrectly displays all of User A's private order details to User B.



[Image: Screenshot of User B's browser showing User A's order details]

Remediation

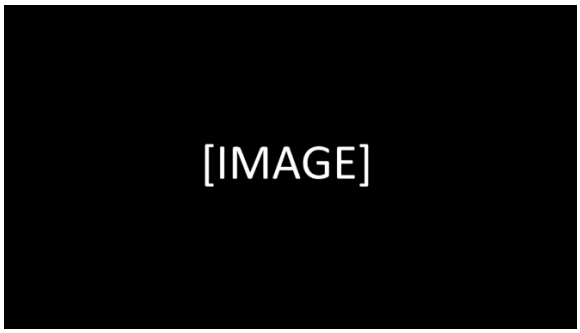
Before rendering any order information, the application's backend code must perform a check to verify that the customer_id associated with the currently logged-in user's session matches the customer_id associated with the requested order_id in the database. If they do not match, the request must be denied with an error message.

4.2.4 Weak Password Policy (Medium)

Severity	**Medium** (CVSSv3.1 Score: 6.5)
Description	The customer registration form allows very simple passwords and does not enforce any complexity or length requirements.
Business Risk	**Likelihood:** High – Attackers constantly use automated tools with lists of common passwords to find and take over accounts with weak credentials. **Impact:** Medium – If successful, it compromises individual user accounts, which can be scaled up to affect many users, leading to fraudulent orders and reputational harm.
Affected URL	https://store.company.com/account/register

Steps to Reproduce

1. Navigate to the user registration page at https://store.company.com/account/register.
2. Fill in the required details for a new account.
3. For the password field, enter 12345678.
4. Submit the form. Observe that the account is created successfully without any validation errors regarding password strength.



[Image: Screenshot of the successful registration message]

Remediation

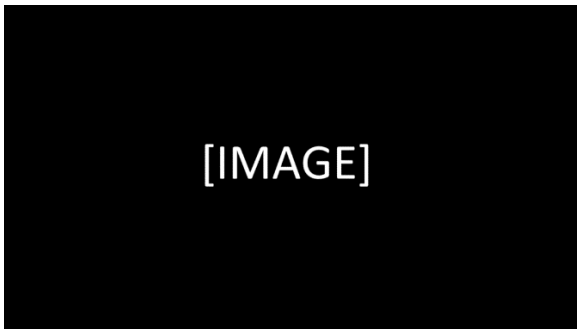
Enforce a strong password policy on the server-side: require a minimum length of 12 characters, a mix of uppercase letters, lowercase letters, numbers, and special characters. Additionally, implement a mechanism to check new passwords against a list of known compromised passwords.

4.2.5 Software Version Disclosure (Low)

Severity	**Low** (CVSSv3.1 Score: 3.7)
Description	The web server reveals its specific software and version number (e.g., "Apache/2.4.53") in its HTTP response headers.
Business Risk	**Likelihood:** Very High – The information is publicly visible to anyone who looks at the server's response headers using standard browser tools. **Impact:** Low – Does not directly lead to a compromise but provides valuable intelligence for attackers, helping them find relevant public exploits for future attacks.
Affected URL	All pages on https://store.company.com

Steps to Reproduce

1. Open a command line terminal.
2. Run the command curl -I https://store.company.com.
3. In the response headers, observe the Server: line which discloses the version information.



[Image: Screenshot of the terminal output showing the Server header]

Remediation

Configure the web server to suppress or hide its version information from all public-facing responses. For Apache, this can be done by setting ServerTokens Prod and ServerSignature Off in the configuration file.

5.0 Conclusion and Recommendations

5.1 Conclusion

The penetration test of the store.company.com platform revealed critical security issues that expose [Company Name] and its customers to a significant level of risk. The findings documented in this report, particularly the Critical SQL Injection vulnerability, require immediate and decisive action. By addressing these vulnerabilities, [Company Name] can significantly improve its security posture, protect its customers' valuable data, and build a more resilient and trustworthy online business. This assessment should be viewed as a crucial first step in an ongoing process of security improvement.

5.2 Recommendations

It is strongly recommended that [Company Name] take the following actions:

1. **Immediately Remediate Critical & High Findings:** Remediate the SQL injection vulnerability and all other Critical/High severity findings without delay. These represent the most immediate threats to your business and customer data.
2. **Remediate All Vulnerabilities:** Address all other identified vulnerabilities in a timely manner to improve the overall security posture of the application.
3. **Implement a Secure Software Development Lifecycle (SDLC):** Integrate security practices throughout the entire development process, from design to deployment, to prevent similar vulnerabilities in the future.
4. **Conduct Regular Penetration Tests:** Proactively identify and address new security vulnerabilities by commissioning independent penetration tests on an annual basis or after major application changes.
5. **Provide Security Awareness Training:** Train all employees, especially developers, to help them recognize and avoid common security threats and write more secure code.

6.0 Disclaimer / Limitations of Assessment

This penetration test report provides a "snapshot in time" of the security posture of the in-scope applications and systems at the time of assessment. The findings in this report are based on the methodologies and tools described herein. It is important to understand that a penetration test does not, and cannot, guarantee 100% security. Malicious actors are constantly developing new attack techniques, and new vulnerabilities may be discovered in software after the conclusion of this assessment. Therefore, security should be viewed as a continuous process of improvement, not a one-time fix. The remediation of the findings in this report is the responsibility of [Company Name].

7.0 References

The methodologies and risk ratings used in this report are aligned with the following industry-standard frameworks and resources:

Resource	Link
OWASP Web Security Testing Guide (WSTG)	https://owasp.org/www-project-web-security-testing-guide/
OWASP Top 10	https://owasp.org/www-project-top-ten/
Common Vulnerability Scoring System (CVSS) v4.0	https://www.first.org/cvss/v4-0/user-guide

Appendix A: Risk Rating Definitions

The following definitions were used to classify the severity of the findings in this report.

Rating	Description
Critical	A vulnerability that, if exploited, could lead to a complete compromise of the application and its data. This typically includes remote code execution, full database compromise, or administrative access. These findings require immediate attention.
High	A vulnerability that could lead to a significant compromise of data or resources, such as unauthorized access to other users' sensitive information or the ability to perform privileged actions on their behalf. These findings should be remediated as a high priority.
Medium	A vulnerability that could provide an attacker with useful information or limited unauthorized access. These flaws may not lead to a direct compromise on their own but could be chained with other vulnerabilities in a more complex attack.
Low	A finding that poses a minor risk. These are typically best practice recommendations that improve the overall security posture of the application but are unlikely to be exploited in a meaningful way on their own (e.g., information leakage).

Appendix B: Tools Used

A combination of commercial, open-source, and custom tools were used to conduct this assessment. The following is a non-exhaustive list of the primary tools employed:

- **Reconnaissance & Enumeration:**
  - Nmap
  - nslookup
- **Vulnerability Scanning & Analysis:**
  - Burp Suite Professional
  - Nessus
  - OWASP ZAP
- **Exploitation & Post-Exploitation:**
  - Burp Suite Professional (Repeater, Intruder)
  - SQLMap
  - Manual Payload Crafting
  - Browser Developer Tools

AKSHAT PARIKH

CYBERSECURITY CONSULTANT

Contact the Penetration Tester: Akshat Parikh

Email: akshatparikh.pt@gmail.com
LinkedIn: https://www.linkedin.com/in/akshatparikh-pt/
Website: