

Black box Penetration Test

For **Can U hack it/Squeel Autoparts**

V32.0

2025-02-13

By: Outpost24 Application Security Team

Table of contents

1. Summary.....	5
1.1 Scope.....	5
1.2 Objective.....	5
1.3 Assumptions.....	5
1.3.1 Rules of Engagement.....	5
1.4 Timeline.....	6
1.5 Summary of Findings.....	6
1.6 Vulnerability analysis.....	6
1.7 Summary of Recommendations.....	7
1.7.1 General Recommendations.....	7
2. Methodology.....	8
2.1 Reconnaissance.....	8
2.2 Exploitation.....	8
2.3 Reporting.....	8
3. Detailed Findings.....	9
3.1 SQUEEL-01: Login Bypass via SQL Injection.....	9
3.1.1 Description.....	9
3.1.2 Potential Impact.....	9
3.1.3 Steps to recreate.....	9
3.1.4 Evidence.....	10
3.1.5 Recommendations.....	10
3.2 SQUEEL-02: Command Execution via Search Field.....	11
3.2.1 Description.....	11
3.2.2 Potential impact.....	11
3.2.3 Steps to recreate.....	11
3.2.4 Evidence.....	12
3.2.5 Recommendations.....	12
3.3 SQUEEL-03: Role Manipulation in Login Request.....	13
3.3.1 Description.....	13
3.3.2 Potential impact.....	13
3.3.3 Steps to recreate.....	13
3.3.4 Evidence.....	13
3.3.5 Recommendations.....	14
3.4 SQUEEL-04: Price Manipulation in E-Commerce Checkout.....	15
3.4.1 Description.....	15
3.4.2 Potential impact.....	15
3.4.3 Steps to recreate.....	15
3.4.4 Evidence.....	15
3.4.5 Recommendations.....	16
3.5 SQUEEL-05: Command injection in the search field.....	17
3.5.1 Description.....	17
3.5.2 Potential impact.....	17

3.5.3 Steps to recreate.....	17
3.5.4 Evidence.....	18
3.5.5 Recommendations.....	18
3.6 SQUEEL-06: Command injection via Search Field.....	19
3.6.1 Description.....	19
3.6.2 Potential impact.....	19
3.6.3 Steps to recreate.....	19
3.6.4 Evidence.....	20
3.6.5 Recommendations.....	20
3.7 SQUEEL-07: Supplier Portal 2FA Bypass via Epoch Time Calculation.....	21
3.7.1 Description.....	21
3.7.2 Potential impact.....	21
3.7.3 Steps to recreate.....	21
3.7.4 Evidence.....	22
3.7.5 Recommendations.....	22
3.8 SQUEEL-08: Session Hijacking via Cookie Manipulation.....	23
3.8.1 Description.....	23
3.8.2 Potential impact.....	23
3.8.3 Steps to recreate.....	23
3.8.4 Evidence.....	24
3.8.5 Recommendations.....	24
3.9 SQUEEL-09: Directory Traversal via File Field.....	25
3.9.1 Description.....	25
3.9.2 Potential impact.....	25
3.9.3 Steps to recreate.....	25
3.9.4 Evidence.....	26
3.9.5 Recommendations.....	26
3.10 SQUEEL-10: Gift Card Exploit for Unlimited Funds.....	27
3.10.1 Description.....	27
3.10.2 Potential impact.....	27
3.10.3 Steps to recreate.....	27
3.10.4 Evidence.....	28
3.10.5 Recommendations.....	29
3.11 SQUEEL-11: SQL injection in Submit Flag Field.....	30
3.11.1 Description.....	30
3.11.2 Potential impact.....	30
3.11.3 Steps to recreate.....	30
3.11.4 Evidence.....	31
3.11.5 Recommendations.....	32
3.12 SQUEEL-12: Reflected XSS on Search Page.....	33
3.12.1 Description.....	33
3.12.2 Potential impact.....	33
3.12.3 Steps to recreate.....	33
3.12.4 Evidence.....	34

3.12.5 Recommendations.....	34
3.13 SQUEEL-13: User Enumeration via Login Timing Discrepancy.....	35
3.13.1 Description.....	35
3.13.2 Potential impact.....	35
3.13.3 Steps to recreate.....	35
3.13.4 Evidence.....	36
3.13.5 Recommendations.....	36
3.14 SQUEEL-14: Stored XSS in Feedback Field.....	37
3.14.1 Description.....	37
3.14.2 Potential impact.....	37
3.14.3 Steps to recreate.....	37
3.14.5 Recommendations.....	38

1. Summary

This report presents the results of a penetration test and security assessment conducted on Can U Hack It/Squeel Autoparts web application infrastructure. For simplicity sake this company will from here on be referred to as solely Squeel Autoparts. The objective of the assessment was to evaluate its security posture and identify potential vulnerabilities that could compromise the confidentiality, integrity, or availability of the application. The report includes a detailed analysis of the findings and provides recommendations to address the identified vulnerabilities and prevent future occurrences.

1.1 Scope

The assessment was a black-box penetration test, with the tester having no prior knowledge of the application's internal architecture or codebase. The scope of the test was limited to the domain <https://heimd411.eu.pythonanywhere.com/> and its subdomains. Any systems, services, or resources outside this domain were explicitly excluded from the assessment. The testing focused on identifying vulnerabilities in the application, including those related to authentication, input validation, session management, and other key areas of concern for web application security.

1.2 Objective

The objective of the assessment was to:

- Identify and exploit potential vulnerabilities in Squeel Autoparts' web application.
- Assess the risks associated with identified vulnerabilities.
- Rank the vulnerabilities based on their impact using the Common Vulnerability Scoring System (CVSS).
- Provide actionable recommendations to mitigate risks and improve the security posture of the application.

1.3 Assumptions

The following assumptions were made during the assessment:

- The domain <https://heimd411.eu.pythonanywhere.com/> and its subdomains are public and under the control of the client.
- The assessment was conducted under a non-disclosure agreement (NDA) and in compliance with the agreed-upon Rules of Engagement.
- The tester had no access to source code, credentials, or other internal resources during the engagement.

1.3.1 Rules of Engagement

- Ethical hacking practices were followed at all times.
- Denial of Service (DoS) or Distributed Denial of Service (DDoS) attacks were prohibited.

- Only in-scope systems (domain and subdomains) were tested.
- The tester adhered to the agreed timeline and methodology for the engagement.
- The assessment was conducted with professionalism, rigor, and a commitment to finding and responsibly disclosing vulnerabilities.

1.4 Timeline

Penetration Testing	Start Date/Time	End Date/Time
Wepp App Pen Test 1	25/11-24 08:30	12/02-25 17:00

Table 1: Penetration testing timeline

1.5 Summary of Findings

Risk	Findings
Low	0
Medium	4
High	4
Critical	6

Table 2: Findings and their respective risk

1.6 Vulnerability analysis

The penetration test revealed ten critical and high-severity vulnerabilities that pose significant risks to the confidentiality, integrity, and availability of Squeel Autoparts' web application. The findings demonstrate systemic weaknesses across multiple security domains, including authentication, authorization, input validation, and business logic controls.

The most severe issues included multiple command injection vulnerabilities in the search functionality, allowing attackers to execute arbitrary system commands and potentially gain unauthorized system access. This critical flaw exposed sensitive system information and highlighted dangerous weaknesses in input validation. Additionally, a role manipulation vulnerability in the login mechanism allowed unauthorized privilege escalation to administrative access, undermining the entire access control framework.

The application's e-commerce functionality contained serious flaws in transaction processing, including a price manipulation vulnerability in the checkout process and a gift card exploitation mechanism that could lead to significant financial losses. These issues demonstrated inadequate server-side validation and business logic controls. The previously identified SQL injection vulnerability in the login functionality and the bypassing of 2FA using predictable epoch time calculations further compound the authentication weaknesses.

Other critical findings included multiple instances of command injection through the search field, exposing system information and potentially enabling privilege escalation. A directory traversal vulnerability in the file field allowed unauthorized access to sensitive files outside the intended directory structure. Session hijacking via cookie manipulation, SQL injection in the Submit Flag field, and both reflected and stored XSS vulnerabilities demonstrated pervasive weaknesses in input validation, session management, and output encoding.

These findings underscore systemic weaknesses in input validation, access control, authentication mechanisms, and business logic validation, which collectively leave the application vulnerable to a wide range of attacks that could result in unauthorized access, data breaches, system compromise, and financial losses.

1.7 Summary of Recommendations

To address these vulnerabilities, Squeel Autoparts must implement comprehensive security controls across multiple layers of the application, focusing on robust input validation, secure authentication, proper access control, and strong transaction validation.

1.7.1 General Recommendations

- **Input Validation and Access Control**
 - Implement comprehensive input validation with parameterized queries and deploy a WAF to prevent injection attacks
 - Establish server-side role-based access control (RBAC) and upgrade to a TOTP-based 2FA system
 - Secure session management using HTTP-only cookies with secure flags and client-specific attributes
- **Transaction and Business Logic Security**
 - Implement server-side validation for all pricing data with cryptographic signing for sensitive parameters
 - Prevent discount abuse on gift cards and monitor transactions for suspicious patterns.
 - Validate file paths and restrict access to sensitive directories using least privilege principles
- **Security Infrastructure and Processes**
 - Conduct regular security training and code reviews while maintaining automated scanning tools
 - Deploy intrusion detection systems and Content Security Policies (CSPs)
 - Establish comprehensive logging and monitoring with regular security assessments

2. Methodology

The penetration test was conducted in three phases: reconnaissance, exploitation, and reporting. The following sections provide an overview of each phase.

2.1 Reconnaissance

The reconnaissance phase began with gathering publicly available information about the target web application, Squeel Autoparts, hosted at <https://heimd411.eu.pythonanywhere.com/>.

This phase involved both passive and active reconnaissance techniques, including:

- Analyzing the structure and functionality of the application.
- Enumerating visible and hidden endpoints, directories, and parameters.
- Conducting basic fingerprinting to identify server and framework technologies in use.

The information gathered in this phase was critical for identifying potential attack vectors and setting the foundation for the exploitation phase.

2.2 Exploitation

In the exploitation phase, the testers used the information from the reconnaissance phase to test for vulnerabilities. The testing followed the OWASP Web security testing framework to ensure coverage of the most prevalent web application security risks.

Key activities included:

- SQL Injection Testing: Input manipulation was performed on login forms to verify if user inputs were sanitized. A bypass was successfully achieved using an SQL payload.
- Cross-Site Scripting (XSS) Testing: Both reflected and stored XSS vulnerabilities were tested by injecting scripts in input fields and observing execution in the browser.
- Manual Testing: Key parts of the application were manually tested to identify business logic flaws or overlooked vulnerabilities.

All identified vulnerabilities were documented with details about their impact and exploitability.

2.3 Reporting

After identifying the vulnerabilities, the testers analyzed the findings to assess their severity and potential impact. Each vulnerability was assigned a unique identifier (SQUEEL-XX) and rated using CVSS 3.1 to provide a consistent and standardized severity rating. The final report includes:

- Detailed descriptions of vulnerabilities with reproduction steps.
- Impact analysis based on possible attack scenarios.
- Recommendations for remediation tailored to the application.

This structured approach ensures that the results are actionable and aligned with industry best practices for security assessments.

3. Detailed Findings

3.1 SQUEEL-01: Login Bypass via SQL Injection

Severity: **Critical 9.8**

The image shows the CVSS v3.1 Base Score Calculator interface. It is a grid-based tool with the following sections:

- ATTACK VECTOR:** Network (selected), Adjacent, Local, Physical.
- ATTACK COMPLEXITY:** Low (selected), High.
- PRIVILEGES REQUIRED:** None (selected), Low, High.
- USER INTERACTION:** None (selected), Required.
- SCOPE:** Changed (selected), Unchanged.
- CONFIDENTIALITY:** High (selected), Low, None.
- INTEGRITY:** High (selected), Low, None.
- AVAILABILITY:** High (selected), Low, None.

At the bottom, the **SEVERITY SCORE VECTOR** is displayed as: **Critical 9.8 CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H**.

Figure 1.1: Calculated CVSS score for SQUEEL-01.

3.1.1 Description

The login functionality was found to be vulnerable to SQL injection. By entering the payload ' OR 1=1-- into the Username field, an attacker could bypass authentication and gain access to the application as the first user in the database (Kalle). This issue arises from insufficient input validation and the failure to use prepared statements in the SQL queries handling authentication.

3.1.2 Potential Impact

An attacker could exploit this vulnerability to:

- Bypass authentication and gain unauthorized access to any user account.
- Access sensitive user data, such as Personally Identifiable Information (PII).
- Potentially compromise administrative privileges if the targeted account belongs to an admin.

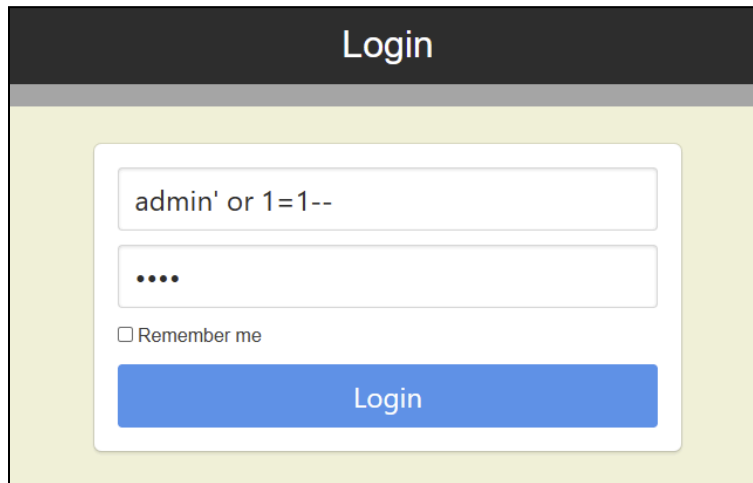
3.1.3 Steps to recreate

1. Navigate to the login page.
2. Enter the following payload in the Username field:

' OR 1=1--

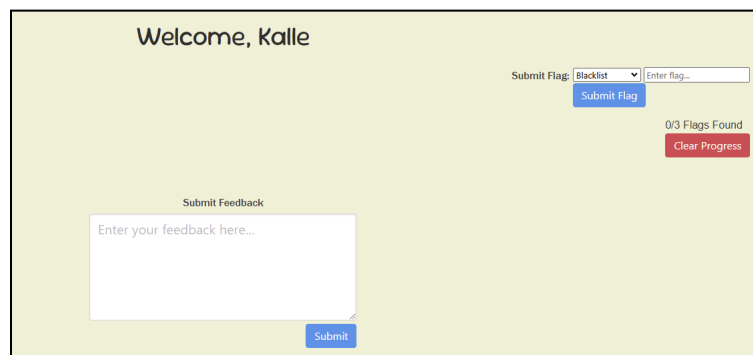
3. Input any value into the Password field.
4. Click Log In.

3.1.4 Evidence



The screenshot shows a login interface with a dark header containing the word "Login". Below the header is a light green background. In the center, there is a white box containing a username field with the text "admin' or 1=1--", a password field with four dots, a "Remember me" checkbox, and a blue "Login" button.

Figure 1.2: Login screen with the payload entered in the Username field.



The screenshot shows the application's home page after a successful login. The header displays "Welcome, Kalle". On the right side, there is a "Submit Flag" dropdown menu currently set to "Blacklist", followed by an "Enter Flag..." input field and a blue "Submit Flag" button. Below these elements, it indicates "0/3 Flags Found" and a red "Clear Progress" button. On the left side, there is a "Submit Feedback" section with a text area labeled "Enter your feedback here..." and a blue "Submit" button.

Figure 1.3: Application logged in as the first user in the database.

3.1.5 Recommendations

- Use parameterized queries for all database interactions to prevent SQL injection attacks.
- Implement strict input validation to reject unexpected or malicious input, including special characters.
- Apply the principle of least privilege, ensuring that database accounts have minimal permissions required for their tasks.

3.2 SQUEEL-02: Command Execution via Search Field

Severity: **Critical 9.8**

CVSS v3.1 Base Score Calculator			
ATTACK VECTOR	ATTACK COMPLEXITY	PRIVILEGES REQUIRED	USER INTERACTION
Network	Low	None	None
Adjacent	High	Low	Required
Local		High	
Physical			
SCOPE	CONFIDENTIALITY	INTEGRITY	AVAILABILITY
Changed	High	High	High
Unchanged	Low	Low	Low
	None	None	None
SEVERITY SCORE VECTOR			
Critical 9.8 CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H			

Figure 2.1: Calculated CVSS score for SQUEEL-02.

3.2.1 Description

The search field allows direct execution of system commands. Entering `ls` lists all files in the directory, confirming an OS command injection vulnerability.

3.2.2 Potential impact

An attacker could enumerate files, access sensitive information, or use this as a stepping stone for further attacks.

3.2.3 Steps to recreate

1. Navigate to the search field.
2. Enter `ls` in the search field.
3. Submit the request
4. Observe that the directory contents are being displayed on the page.

3.2.4 Evidence

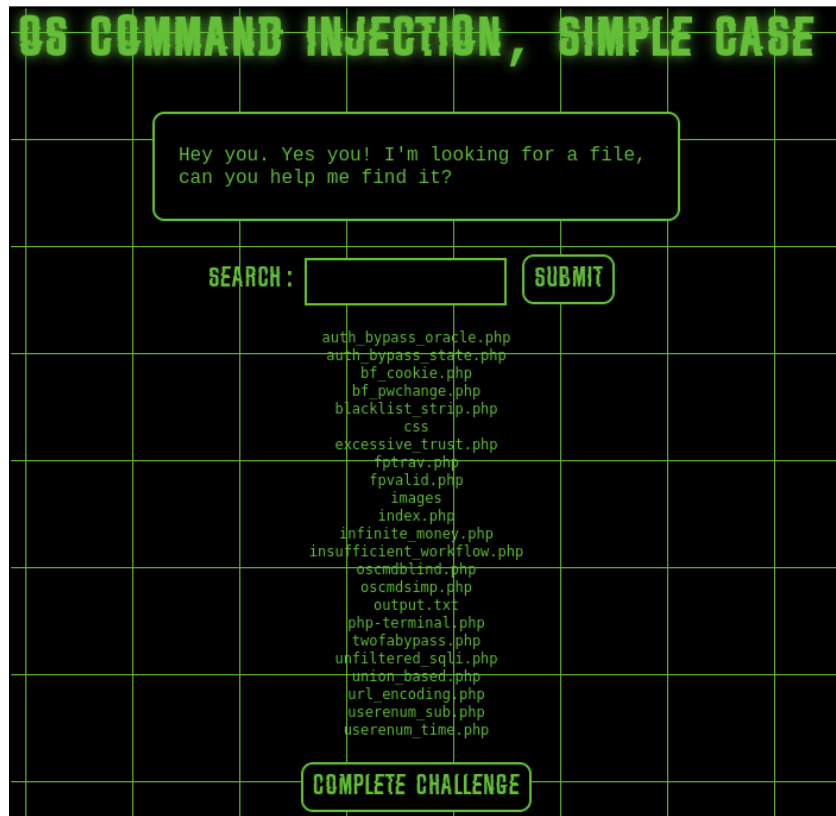


Figure 2.2: Search field displaying the output of `ls`, showing all files in the directory.

3.2.5 Recommendations

- Disable command execution in user inputs.
- Use secure coding practices to prevent OS command injection.
- Restrict application permissions to prevent unnecessary access.

3.3 SQUEEL-03: Role Manipulation in Login Request

Severity: **Critical 9.6**

The image shows a CVSS v3.1 Base Score Calculator interface. It is divided into two main sections: the top section for Attack Vector, Attack Complexity, Privileges Required, and User Interaction; and the bottom section for Scope, Confidentiality, Integrity, and Availability. The bottom section also includes a Severity Score Vector box at the very bottom.

ATTACK VECTOR	ATTACK COMPLEXITY	PRIVILEGES REQUIRED	USER INTERACTION
Network	Low	None	None
Adjacent	High	Low	Required
Local		High	
Physical			

SCOPE	CONFIDENTIALITY	INTegrity	AVAILABILITY
Changed	High	High	High
Unchanged	Low	Low	Low
	None	None	None

SEVERITY SCORE VECTOR

Critical 9.6 CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:N

Figure 3.1: Calculated CVSS score for SQUEEL-03.

3.3.1 Description

The login request allows users to select their role, but the role is client-side controlled. By intercepting and modifying the request, an attacker can escalate privileges to an admin role.

3.3.2 Potential impact

An attacker could gain unauthorized administrative access, potentially modifying user data, accessing sensitive information, or taking control of the application.

3.3.3 Steps to recreate

1. Attempt to log in and observe the role selection parameter.
2. Intercept the login request
3. Modify the role parameter from user to

```
select_role=admin
```

4. Submit the modified request.
5. Observe that access is granted to the admin panel.

3.3.4 Evidence

```
14 Cookie: PHPSESSID=8512bf
15 Upgrade-Insecure-Request
16 Priority: u=0, i
17
18 select_role=admin
```

Figure 3.2: Intercepted login request with modified role parameter.

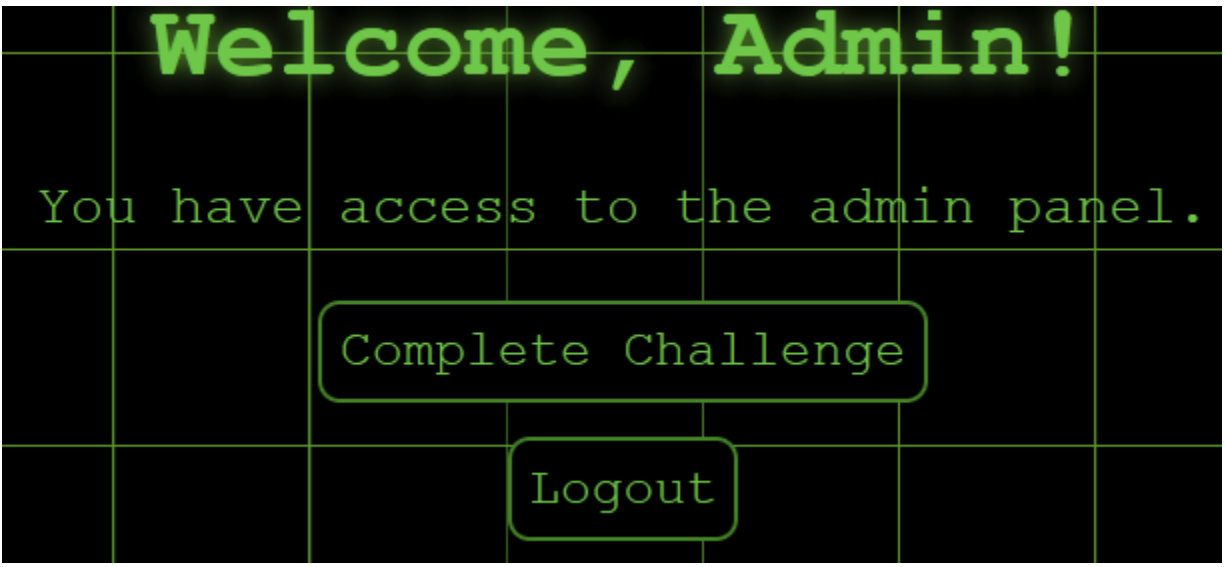


Figure 3.3: Admin panel access granted.

3.3.5 Recommendations

- Enforce role-based access control (RBAC) on the server side.
- Validate and restrict user roles during authentication.
- Implement proper session management.

3.4 SQUEEL-04: Price Manipulation in E-Commerce Checkout

Severity: **High 8.2**

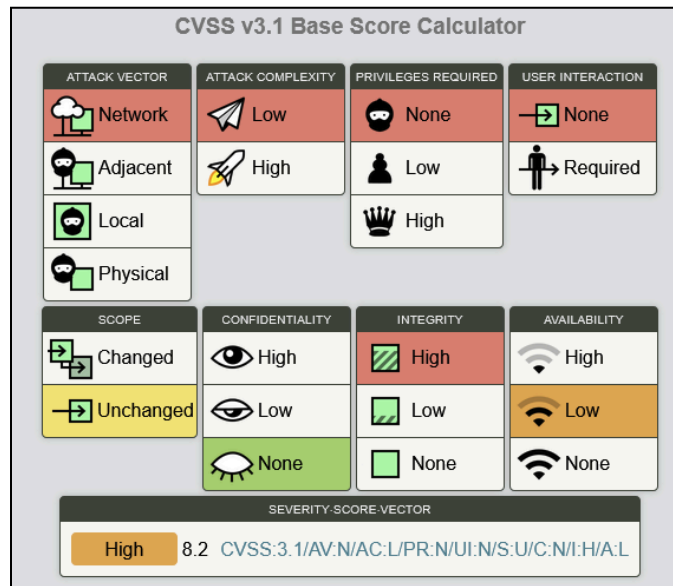


Figure 4.1: Calculated CVSS score for SQUEEL-04.

3.4.1 Description

The checkout request contains a modifiable price parameter, allowing users to change the price of an item and complete a fraudulent purchase.

3.4.2 Potential impact

An attacker can purchase expensive items at a reduced price, resulting in financial losses for the business.

3.4.3 Steps to recreate

1. Initiate a purchase for an expensive item.
2. Intercept the request and modify the price like so:

```
price=1
```

3. Submit the modified request.
4. Observe that the item is successfully purchased for \$1.

3.4.4 Evidence

```
Connection: keep-alive
item=Graphics+Card&price=1&token=
26abee3d4bece6000a6a6b5969ab873079
```

Figure 4.2: Request with modified price parameter.

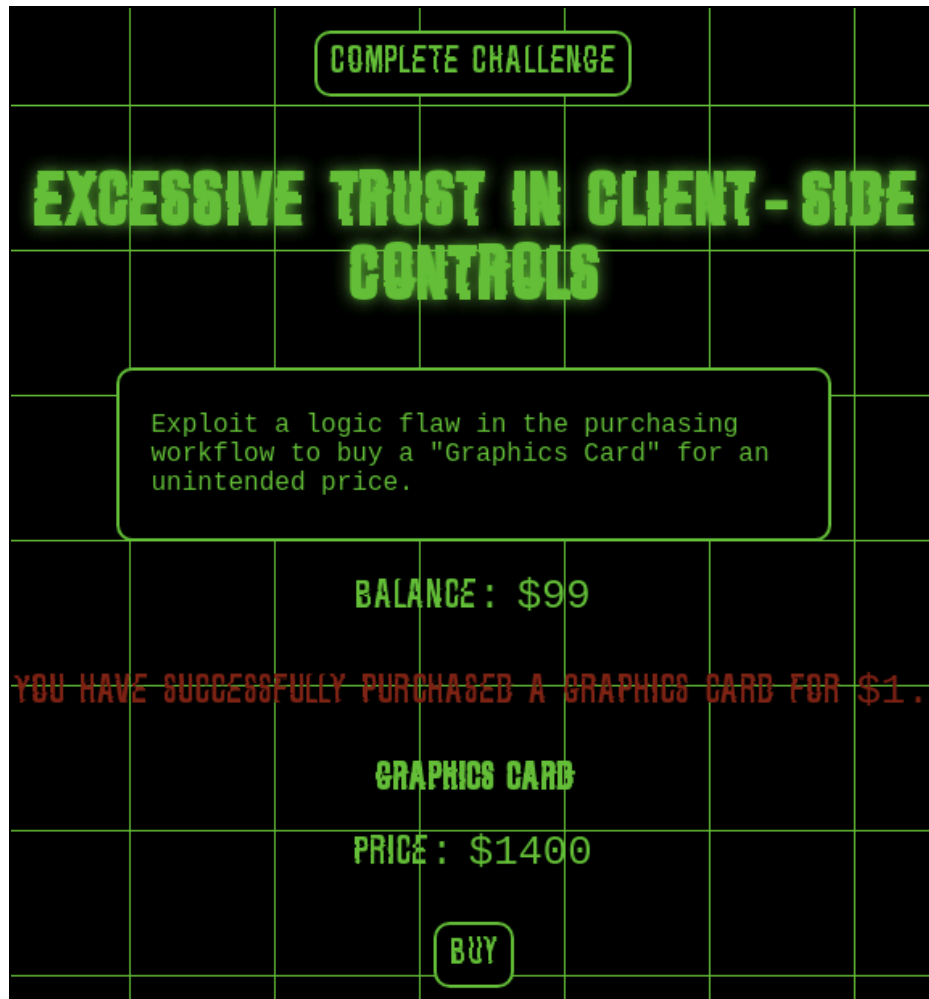


Figure 4.3: Confirmation of purchase at manipulated price.

3.4.5 Recommendations

- Implement server-side validation to enforce correct pricing.
- Use cryptographic signing to prevent price manipulation.
- Log and monitor unusual purchasing patterns.

3.5 SQUEEL-05: Command injection in the search field

Severity: **Critical 9.1**

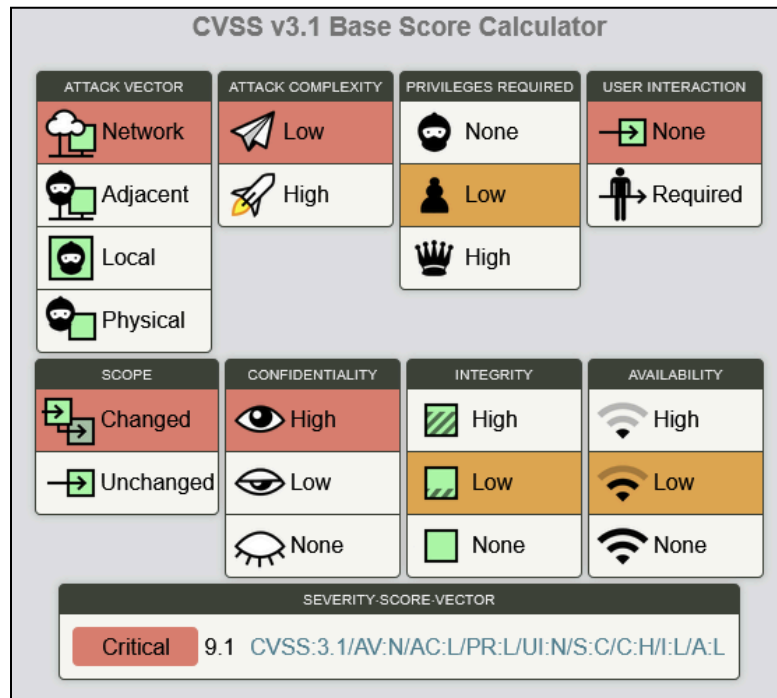


Figure 5.1: Calculated CVSS score for SQUEEL-05.

3.5.1 Description

The search field allows command injection, revealing the system user `www-data`. This could lead to privilege escalation attacks.

3.5.2 Potential impact

An attacker could enumerate users, escalate privileges, and perform lateral movement within the system.

3.5.3 Steps to recreate

1. Enter `whoami` in the search field.
2. Submit the request.
3. Observer that `www-data` is displayed.
4. Enter:

```
cat output.txt
```

5. Observe that the file contents are displayed.

3.5.4 Evidence

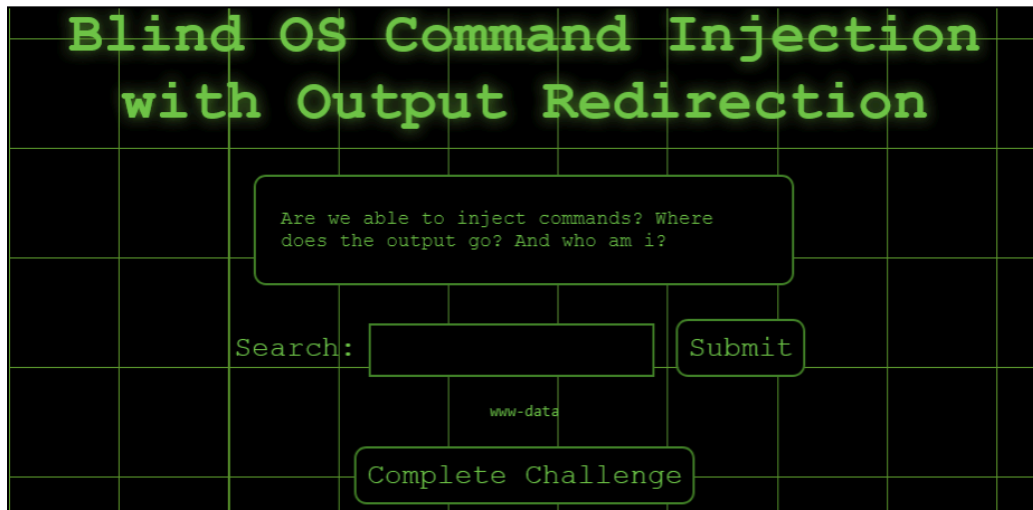


Figure 5.2: Search field displaying `www-data` after executing `whoami`.

3.5.5 Recommendations

- Implement strict input validation to prevent command execution.
- Use a secure coding framework to sanitize user input.
- Restrict system user permissions to minimize exploitation risk.

3.6 SQUEEL-06: Command injection via Search Field

Severity: **Critical 9.1**

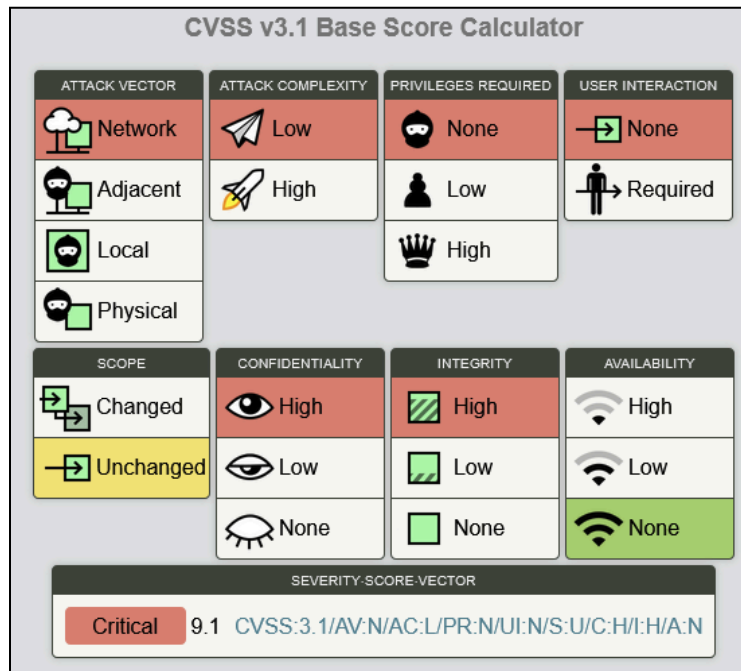


Figure 6.1: Calculated CVSS score for SQUEEL-06.

3.6.1 Description

A command injection vulnerability exists in the search field. When a user inputs `pwd`, the server executes the command and returns `/var/www/html/public`, indicating improper input handling.

3.6.2 Potential impact

An attacker could execute arbitrary system commands, gaining unauthorized access to system files, modifying data, or escalating privileges.

3.6.3 Steps to recreate

1. Navigate to the search field.
2. Enter `pwd` as the search term.
3. Submit the request.
4. Observe that the current working directory `/var/www/html/public` is displayed on the page.

3.6.4 Evidence



Figure 6.2: Search field with `pwd` entered, displaying `/var/www/html/public` on the page.

3.6.5 Recommendations

- Sanitize user input and prevent command execution.
- Use parameterized queries or allowlisted commands.
- Run the application with minimal privileges to limit the impact of exploitation.

3.7 SQUEEL-07: Supplier Portal 2FA Bypass via Epoch Time Calculation

Severity: **High 8.9**

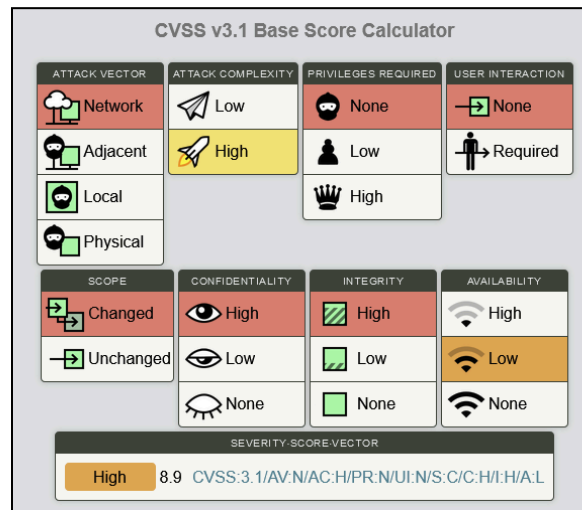


Figure 7.1: Calculated CVSS score for SQUEEL-07.

3.7.1 Description

The supplier verification functionality is vulnerable to a 2FA bypass attack. By entering a valid supplier username and calculating the epoch time divided by 60 modulo 1,000,000, an attacker can generate a valid 6-digit 2FA code. This bypasses the intended security mechanism and allows unauthorized access to the supplier portal.

3.7.2 Potential impact

An attacker could exploit this vulnerability to:

- Gain unauthorized access to supplier accounts.
- View and potentially modify sensitive supplier information.
- Disrupt supplier operations or exploit the system further.

3.7.3 Steps to recreate

1. Navigate to:

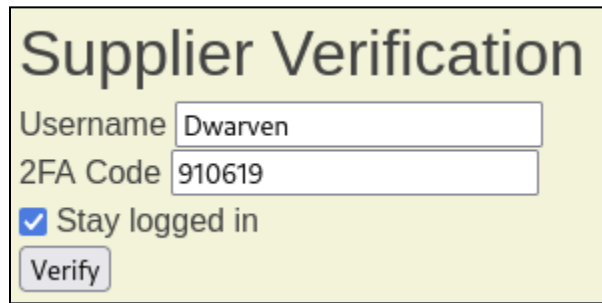
<https://heimd411.eu.pythonanywhere.com/supplier/verify>

2. Enter a valid supplier username in the username field.
3. Calculate the 2FA key using the formula:

$(\text{epoch} / 60) \% 1000000$

- Enter the calculated 2FA code into the code field.
- Submit the form to access the supplier portal.

3.7.4 Evidence



A screenshot of a web form titled "Supplier Verification". It contains two input fields: "Username" with the value "Dwarven" and "2FA Code" with the value "910619". Below these fields is a checkbox labeled "Stay logged in" which is checked. At the bottom is a button labeled "Verify".

Figure 7.2: Supplier login with the epoch time entered and a valid supplier username.

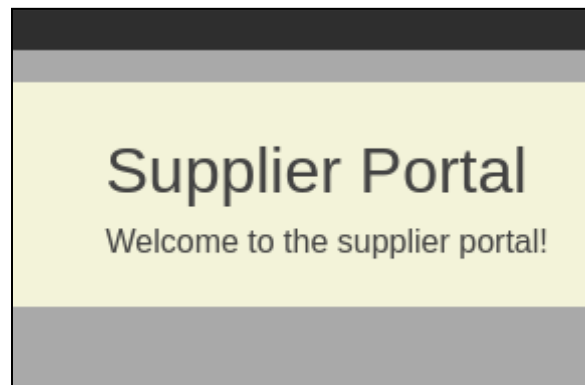


Figure 7.3: The welcome message of the supplier portal after successful 2FA bypass.

3.7.5 Recommendations

- Implement a more secure 2FA mechanism, such as time-based one-time passwords (TOTP) using a shared secret and time synchronization.
- Use rate-limiting and account lockout policies to prevent brute-forcing or guessing of 2FA codes.
- Log and monitor all 2FA login attempts for unusual activity.

3.8 SQUEEL-08: Session Hijacking via Cookie Manipulation

Severity: **High 8.7**

The image shows the CVSS v3.1 Base Score Calculator interface. It is a grid-based tool with various categories and their corresponding values. The categories and their values are:

ATTACK VECTOR	ATTACK COMPLEXITY	PRIVILEGES REQUIRED	USER INTERACTION
Network	Low	None	None
Adjacent	High	Low	Required
Local		High	
Physical			

SCOPE	CONFIDENTIALITY	INTEGRITY	AVAILABILITY
Changed	High	High	High
Unchanged	Low	Low	Low
	None	None	None

SEVERITY SCORE VECTOR

High 8.7 CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:C/C:H/I:H/A:N

Figure 8.1: Calculated CVSS score for SQUEEL-08.

3.8.1 Description

The session management system was found to be vulnerable to session hijacking. By logging in as an attacker and replacing their session cookie with that of the victim's cookie, the attacker could gain unauthorized access to the victim's account. This issue arises from the lack of session cookie validation and proper security controls.

3.8.2 Potential impact

An attacker could exploit this vulnerability to:

- Impersonate any user with a valid session cookie.
- Access sensitive data tied to victim accounts.
- Perform actions on behalf of the victim, including unauthorized transactions or modifications.

3.8.3 Steps to recreate

1. Create two accounts: attacker and victim.
2. Log in as the victim, capture the session cookie.
3. Log in as the attacker, replace their session cookie with the victim's cookie.
4. Forward the modified request to gain access to the victim's account.

3.8.4 Evidence

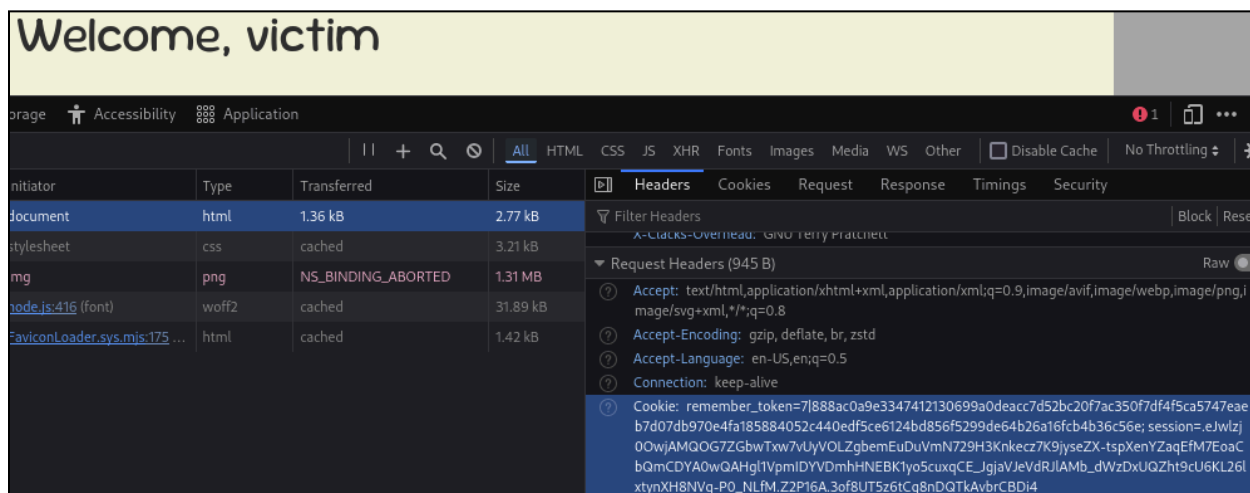


Figure 8.2: Captured session cookie of the victim.

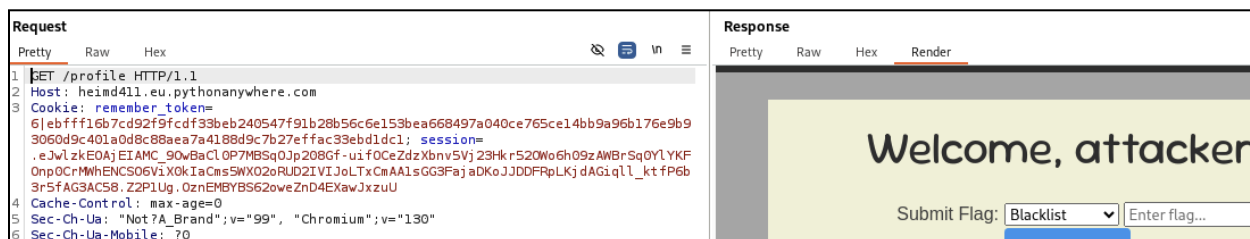


Figure 8.3: Attacker's session cookie before replacement.

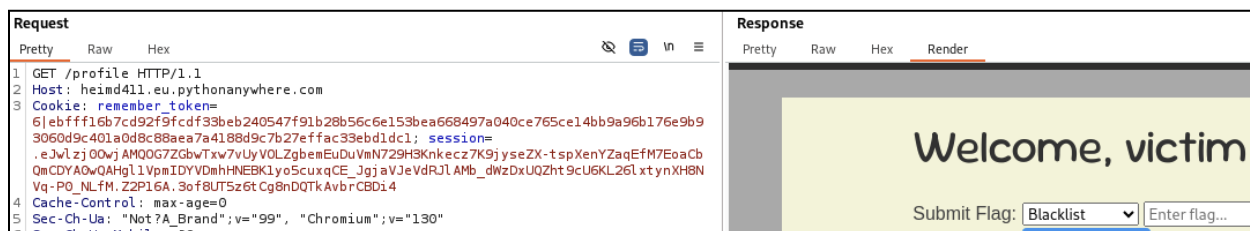


Figure 8.4: Modified request showing the attacker logged in as the victim.

3.8.5 Recommendations

- Implement secure session management practices, such as binding sessions to client-specific attributes like IP address or User-Agent.
- Use HTTP-only and Secure flags for cookies to prevent theft via client-side vulnerabilities.
- Regularly refresh and invalidate session tokens to reduce the impact of session theft.

3.9 SQUEEL-09: Directory Traversal via File Field

Severity: **High 7.5**

CVSS v3.1 Base Score Calculator			
ATTACK VECTOR	ATTACK COMPLEXITY	PRIVILEGES REQUIRED	USER INTERACTION
Network	Low	None	None
Adjacent	High	Low	Required
Local		High	
Physical			
SCOPE	CONFIDENTIALITY	INTEGRITY	AVAILABILITY
Changed	High	High	High
Unchanged	Low	Low	Low
	None	None	None
SEVERITY SCORE VECTOR			
High 7.5 CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N			

Figure 9.1: Calculated CVSS score for SQUEEL-09.

3.9.1 Description

The application allows user input in a file field and does not properly sanitize it. By entering `../uploads/secret.txt`, an attacker can access arbitrary files outside the intended directory, potentially exposing sensitive data.

3.9.2 Potential impact

An attacker could access sensitive files, such as configuration files, credentials, or application source code, leading to data breaches or further exploitation.

3.9.3 Steps to recreate

1. Navigate to: the file field.
2. Enter:

`../uploads/secret.txt`

3. Submit the request.
4. Observe that the contents of `secret.txt` are displayed on the page.

3.9.4 Evidence



Figure 9.2: The file upload field with the input `../uploads/secret.txt` and the retrieved file contents displayed on the page.

3.9.5 Recommendations

- Implement proper input validation and restrict directory traversal sequences (`../`).
- Use server-side validation to allow only expected file paths.
- Set appropriate file system permissions to prevent unauthorized access to sensitive files.

3.10 SQUEEL-10: Gift Card Exploit for Unlimited Funds

Severity: **High 7.2**

CVSS v3.1 Base Score Calculator			
ATTACK VECTOR	ATTACK COMPLEXITY	PRIVILEGES REQUIRED	USER INTERACTION
Network	Low	None	None
Adjacent	High	Low	Required
Local		High	
Physical			
SCOPE	CONFIDENTIALITY	INTEGRITY	AVAILABILITY
Changed	High	High	High
Unchanged	Low	Low	Low
	None	None	None
SEVERITY SCORE VECTOR			
High 7.2 CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:L/I:L/A:N			

Figure 10.1: Calculated CVSS score for SQUEEL-10.

3.10.1 Description

The system allows a 30% discount coupon to be applied to gift card purchases, enabling users to generate unlimited funds.

3.10.2 Potential impact

An attacker can exploit this loophole to generate infinite funds and make fraudulent purchases, leading to significant financial losses.

3.10.3 Steps to recreate

1. Add 10 gift cards worth \$100 to the cart.
2. Apply the 30% discount coupon.
3. Complete the purchases for \$70.
4. Observe that the account balance increases beyond the initial value.

3.10.4 Evidence

CART		COUPON	
BALANCE: \$100		COUPON CODE: <input type="text" value="THIRTYOFF"/>	
<div>\$10 Gift Card - \$10 \$10 Gift Card - \$10 \$10 Gift Card - \$10 \$10 Gift Card - \$10 \$10 Gift Card - \$10 \$10 Gift Card - \$10 \$10 Gift Card - \$10 \$10 Gift Card - \$10 \$10 Gift Card - \$10 \$10 Gift Card - \$10</div>		<div>APPLY COUPON (THIRTYOFF)</div> <div>REDEEM GIFT CARD</div> <div>GIFT CARD CODE: <input type="text"/></div> <div>REDEEM</div>	
TOTAL: \$100			
DISCOUNTED TOTAL: \$70			
<div>PURCHASE</div> <div>CLEAR CART</div>			

Figure 10.2: Gift cards with applied discount.

CART	
BALANCE: \$532	
<div>Intel CPU - \$600</div>	
TOTAL: \$600	
DISCOUNTED TOTAL: \$420	
<div>PURCHASE</div> <div>CLEAR CART</div>	

Figure 10.2: Increased account balance.

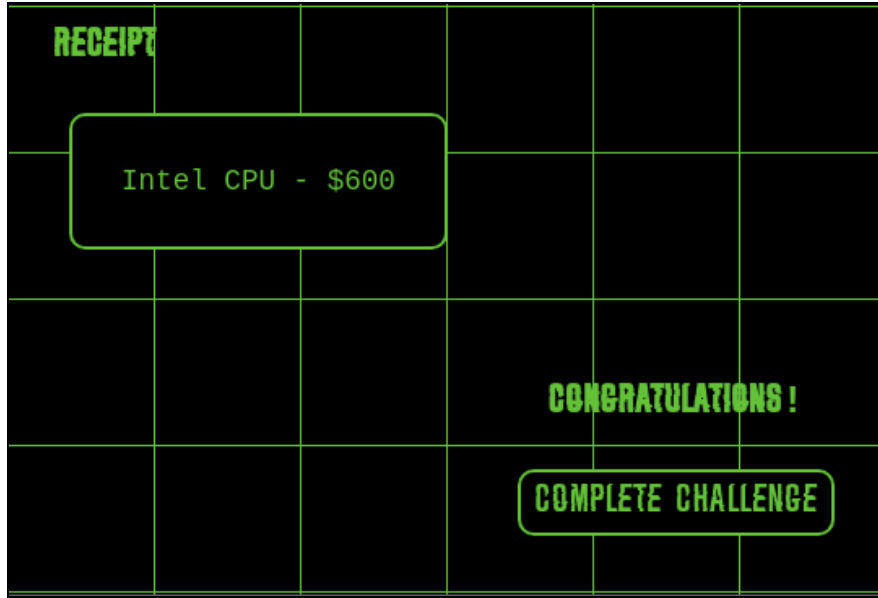


Figure 10.3: Successful fraudulent purchase.

3.10.5 Recommendations

- Prevent coupons from applying to gift card purchases.
- Implement server-side validation to restrict discount abuse.
- Monitor and audit transactions for anomalies.

3.11 SQUEEL-11: SQL injection in Submit Flag Field

Severity: **Medium 6.5**

The image shows a CVSS v3.1 Base Score Calculator interface. It is divided into several sections: Attack Vector, Attack Complexity, Privileges Required, User Interaction, Scope, Confidentiality, Integrity, Availability, and a final Severity-Score-Vector summary.

ATTACK VECTOR	ATTACK COMPLEXITY	PRIVILEGES REQUIRED	USER INTERACTION
Network	Low	None	None
Adjacent	High	Low	Required
Local		High	
Physical			

SCOPE	CONFIDENTIALITY	INTegrity	AVAILABILITY
Changed	High	High	High
Unchanged	Low	Low	Low
	None	None	None

SEVERITY-SCORE-VECTOR			
Medium	6.5	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N	

Figure 11.1: Calculated CVSS score for SQUEEL-11.

3.11.1 Description

The Submit Flag field on the profile page was found to be vulnerable to SQL injection, enabling database enumeration. By leveraging the vulnerability, the penetration tester was able to execute payloads that revealed database schema information and sensitive data such as secret flags. This issue results from inadequate input validation or sanitization in the application.

3.11.2 Potential impact

An attacker could exploit this vulnerability to:

- Enumerate database schema and retrieve table names.
- Extract sensitive information stored in the database, such as user details.

3.11.3 Steps to recreate

4. Log in to the application or register and log in as a new user.
5. Navigate to the Profile page.
6. In the Submit Flag field, input the following payloads (one at a time) while selecting URL Encoded as the option:
 - To retrieve table names:

```
' UNION SELECT tbl_name FROM sqlite_master WHERE type='table'--
```

- To retrieve SQL statements for the *secret_flags* table:

```
' UNION SELECT sql FROM sqlite_master WHERE tbl_name='secret_flags'--
```

- To retrieve flag values from the *secret_flags* table:

```
' UNION SELECT flag FROM secret_flags--
```

7. Submit the payloads and observe the output

3.11.4 Evidence

Query result: cart
Query result: part
Query result: secret_flags
Query result: user
Query result: user_progress
Invalid flag!
Welcome, Kalle

Figure 11.2: Output showing table names retrieved using the first payload

Query result: CREATE TABLE secret_flags (id INTEGER PRIMARY KEY, flag TEXT NOT NULL)
Invalid flag!
Welcome, Kalle

Figure 11.3: Output displaying the SQL statements for the secret_flags table

Query result: None
Query result: 1
Query result: FLAG[DATABASES_ARE_FUN]
Flag accepted!
Welcome, Kalle

Figure 11.4: Output showing the retrieved secret flag..

3.11.5 Recommendations

- Ensure user inputs are validated and sanitized to prevent malicious payloads.
- Limit database error messages to generic responses, avoiding exposure of schema details.
- Restrict database user permissions to minimize access to sensitive data.
- Use a Web Application Firewall (WAF) to detect and block SQL injection attempts.

3.12 SQUEEL-12: Reflected XSS on Search Page

Severity: **Medium 5.4**

CVSS v3.1 Base Score Calculator			
ATTACK VECTOR	ATTACK COMPLEXITY	PRIVILEGES REQUIRED	USER INTERACTION
Network	Low	None	None
Adjacent	High	Low	Required
Local		High	
Physical			
SCOPE	CONFIDENTIALITY	INTEGRITY	AVAILABILITY
Changed	High	High	High
Unchanged	Low	Low	Low
	None	None	None
SEVERITY SCORE VECTOR			
Medium 5.4 CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:N			

Figure 12.1: Calculated CVSS score for SQUEEL-12.

3.12.1 Description

The search functionality was found to be vulnerable to reflected Cross-Site Scripting (XSS). By injecting a malicious script payload into the search field, the script was executed in the user's browser. This vulnerability occurs due to the application's failure to sanitize or encode user input before reflecting it in the HTML response.

3.12.2 Potential impact

An attacker could exploit this vulnerability to:

- Steal session cookies or other sensitive data via a crafted malicious link.
- Modify the appearance or behavior of the web page.
- Escalate the attack to perform phishing or deliver malware.

3.12.3 Steps to recreate

1. Navigate to the Search page.
2. Enter the following payload in the search bar:

```
<script>alert("XSS")</script>
```

3. Press Search and observe the result. A pop-up box with the message "XSS" will appear.

3.12.4 Evidence

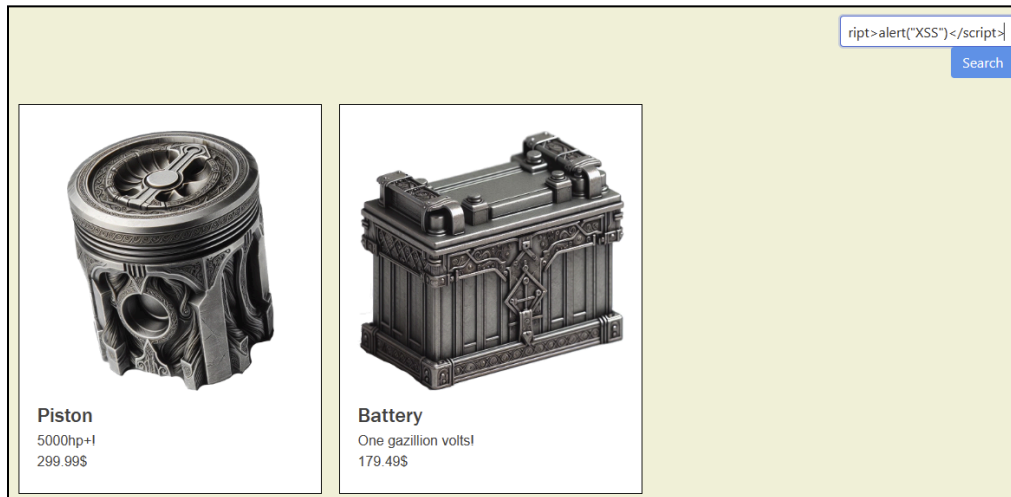


Figure 12.2: Search bar with the malicious payload entered.

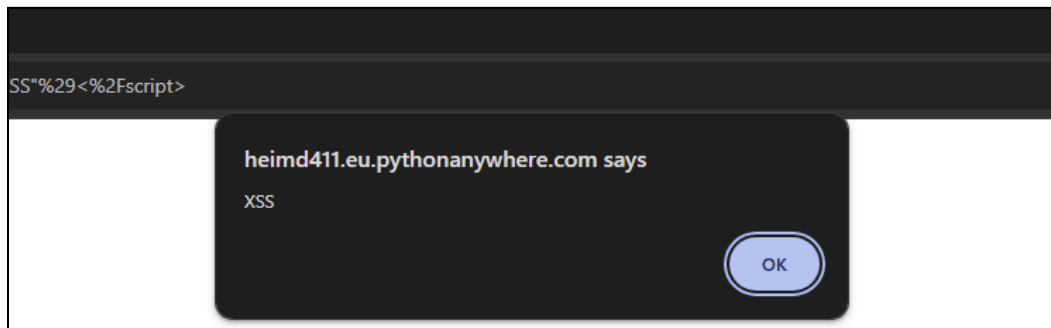


Figure 12.3: Pop-up box displaying "XSS" after submitting the payload

3.12.5 Recommendations

- Sanitize and encode all user input before rendering it in the browser. Use trusted libraries like OWASP's Java Encoder for this purpose.
- Implement a Content Security Policy (CSP) to limit the execution of unauthorized scripts in the browser.

3.13 SQUEEL-13: User Enumeration via Login Timing Discrepancy

Severity: **Medium 5.3**

CVSS v3.1 Base Score Calculator			
ATTACK VECTOR	ATTACK COMPLEXITY	PRIVILEGES REQUIRED	USER INTERACTION
Network	Low	None	None
Adjacent	High	Low	Required
Local		High	
Physical			
SCOPE	CONFIDENTIALITY	INTEGRITY	AVAILABILITY
Changed	High	High	High
Unchanged	Low	Low	Low
	None	None	None
SEVERITY SCORE VECTOR			
Medium 5.3 CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N			

Figure 13.1: Calculated CVSS score for SQUEEL-13.

3.13.1 Description

The login functionality is vulnerable to user enumeration due to a response discrepancy in processing valid vs. invalid emails. Valid email addresses result in significantly longer response received compared to invalid ones. This enables attackers to enumerate users and target specific accounts for further attacks.

3.13.2 Potential impact

An attacker could exploit this vulnerability to:

- Enumerate valid email addresses of application users.
- Use this information to perform targeted phishing or brute-force attacks.
- Undermine user confidentiality and trust in the application.

3.13.3 Steps to recreate

1. Capture the login request.
2. Use a tool like Turbo Intruder to send login requests with a list of both real and fake email addresses.
3. Observe the response received for each request.
4. Identify valid emails based on longer response.

3.13.4 Evidence

Admin Portal				
Username	Email	Admin	Supplier	Actions
administrator	admin@squeel.com	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Dwarven	dwarven@fortress.com	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<button>Delete</button>
Kalle	kalle.anka@ankeborg.se	<input type="checkbox"/>	<input type="checkbox"/>	<button>Delete</button>
Kurt	Kurt@jmail.com	<input type="checkbox"/>	<input type="checkbox"/>	<button>Delete</button>
Forge	TheForge@Fortress.com	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<button>Delete</button>
attacker	attacker@test.com	<input type="checkbox"/>	<input type="checkbox"/>	<button>Delete</button>
victim	victim@test.com	<input type="checkbox"/>	<input type="checkbox"/>	<button>Delete</button>
test	test@test.com	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<button>Delete</button>

Figure 13.2: Admin portal showing all users and their email addresses.

Payload	Response received ▾
kalle.anka@ankeborg.se	963
Kurt@jmail.com	957
victim@test.com	954
TheForge@Fortress.com	873
dwarven@fortress.com	852
attacker@test.com	768
admin@squeel.com	234
fake.name2024@freemailbox.org	49
example.person@tempmail.info	47
	46

Figure 13.3: Burp Suite Intruder results showing response times for valid and invalid email addresses, sorted by response received.

3.13.5 Recommendations

- Use constant-time processing for login attempts to prevent timing-based attacks.
- Implement rate-limiting and CAPTCHA to deter automated brute-force attempts.
- Provide generic error messages for login failures to avoid revealing account existence.

3.14 SQUEEL-14: Stored XSS in Feedback Field

Severity: **Medium 4.6**

The image shows a CVSS v3.1 Base Score Calculator interface. It consists of several sections for selecting attack characteristics:

- ATTACK VECTOR:** Network (selected), Adjacent, Local, Physical.
- ATTACK COMPLEXITY:** Low (selected), High.
- PRIVILEGES REQUIRED:** None (selected), Low, High.
- USER INTERACTION:** None (selected), Required.
- SCOPE:** Changed (selected), Unchanged.
- CONFIDENTIALITY:** High (selected), Low, None.
- INTEGRITY:** High (selected), Low, None.
- AVAILABILITY:** High (selected), Low, None.

At the bottom, the **SEVERITY: SCORE: VECTOR** is displayed as: **Medium 4.6 CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:U/C:L/I:L/A:N**.

Figure 14.1: Calculated CVSS score for SQUEEL-14.

3.14.1 Description

A stored Cross-Site Scripting (XSS) vulnerability was identified in the Feedback functionality on the profile page. By submitting a malicious script payload in the feedback field, the payload was stored in the database and executed whenever the feedback was viewed. This vulnerability enables persistent script execution, impacting all users who view the feedback.

3.14.2 Potential impact

An attacker could exploit this reflected XSS vulnerability to execute arbitrary scripts in the context of the victim's browser. This could allow the attacker to:

- Steal session cookies or other sensitive data, potentially leading to account hijacking.
- Redirect users to malicious websites for phishing or malware delivery.
- Manipulate the appearance or behavior of the web application to deceive users

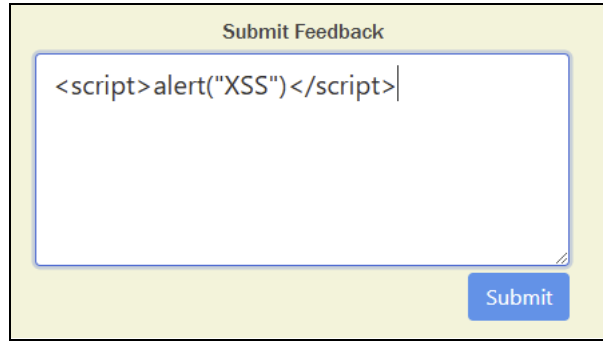
3.14.3 Steps to recreate

5. Log in to the application.
6. Navigate to the Profile page and locate the Feedback field.
7. Enter the following payload in the field:

```
<script>alert("XSS")</script>
```

8. Submit the feedback and then a pop-up box with the message "XSS" will appear.

3.4.4 Evidence



Submit Feedback

`<script>alert("XSS")</script>`

Submit

Figure 14.2: Feedback field with the malicious payload entered.

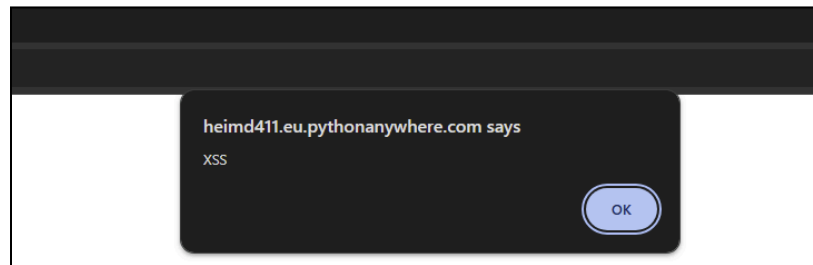


Figure 14.3: Pop-up box displaying "XSS" after submitting the payload.

3.14.5 Recommendations

- Implement secure cookie attributes such as HTTP-only and Secure flags to mitigate the impact of cookie theft during an XSS attack.
- Use automated scanning tools to detect stored XSS issues during the software development lifecycle.