

Vulnerability Assessment and Penetration Testing (VAPT) Report

Document title: VAPT Report on portswigger, DVWA and zero.webappsecurity

Author: Abishek V

Date: 10.08.2025

TABLE OF CONTENT

- Document control
 - Issue Control
- Security Testing
 - Vulnerability Assessment and Penetration Testing (VAPT) Objective
 - Testing the target website
- Methodology
 - Information Gathering
 - Scanning and Enumeration
 - Vulnerability Analysis
 - Exploitation
 - Reporting
 - Remediation
- Summary of Finding
 - Risk Calculation and Classification
 - Detailed Findings
- Detailed Vulnerability Findings
- Conclusion

DOCUMENT CONTROL

ISSUE CONTROL

Classification	Confidential
Document title	VAPT Report on portswigger,DVWA and zero.webappsecurity
Scope	Portswigger,DVWA,Zero.webappsecurity
Date	10.08.2025
Author	Abishek V

SECURITY TESTING

Security testing is the process of checking if a system, application, or network is secure. The goal is to find and fix any weaknesses that could be exploited by attackers.

Key parts of security testing include:

1. Identifying Vulnerabilities: Finding potential security flaws in the system.
2. Testing Defenses: Checking how well the system's security measures protect against attacks.
3. Fixing Issues: Ensuring any discovered weaknesses are fixed to prevent future attacks.

OBJECTIVE OF SECURITY TESTING

The objective of security testing is to ensure that a system, application, or network is safe from cyber threats. Specifically, it aims to:

1. Find Weaknesses: Identify any security flaws that could be exploited by attackers.
2. Protect Data: Ensure that sensitive information is kept safe and private.
3. Prevent Attacks: Make sure that the system is strong enough to stop or withstand potential cyberattacks.
4. Build Trust: Ensure users and customers can trust that their data and the systems they use are secure.

Vulnerability Assessment and Penetration Testing (VAPT)

Vulnerability Assessment and Penetration Testing (VAPT) is a process used to identify and fix security weaknesses in a computer system, network, or application. It has two main parts:

1. Vulnerability Assessment: This part involves scanning the system to find potential security flaws or vulnerabilities that could be exploited by attackers.
2. Penetration Testing: After identifying vulnerabilities, this part involves actively trying to exploit them to see how far an attacker could get and what damage they could cause.

Together, VAPT helps organizations secure their systems by identifying and addressing potential threats before attackers can exploit them.

Role of Vulnerability Assessment and Penetration Testing (VAPT)

The role of Vulnerability Assessment and Penetration Testing (VAPT) is to help protect systems from cyberattacks by identifying and fixing security weaknesses.

- Vulnerability Assessment finds potential security issues that could be exploited.
- Penetration Testing tries to actually exploit those issues to see how much damage an attacker could cause.

By doing both, VAPT helps ensure that systems are secure and less likely to be hacked.

Use of VAPT in Risk Management

VAPT is used in risk management to reduce the chances of cyberattacks.

1. Identifying Risks: VAPT finds security weaknesses that could be exploited by attackers, helping organizations understand where they are vulnerable.
2. Prioritizing Fixes: It helps prioritize which vulnerabilities need to be fixed first based on how likely they are to be exploited and the potential impact.
3. Preventing Breaches: By addressing these vulnerabilities, VAPT lowers the risk of a security breach, protecting sensitive data and systems. In simple terms, VAPT helps manage and reduce the risk of cyberattacks by finding and fixing security gaps.

METHODOLOGY

- Information Gathering: Collecting data about a target, such as networks, websites, or systems, to understand how they work. This helps identify potential weak points.
- Scanning and Enumeration: Actively probing the target to discover open ports, services, and resources. It's like knocking on doors to see which ones are unlocked.
- Vulnerability Analysis: Analyzing the collected data to find weaknesses (vulnerabilities) in systems or networks. It's about identifying what could be exploited.
- Exploitation: Taking advantage of the discovered vulnerabilities to gain unauthorized access or control over the target.
- Reporting: Documenting the vulnerabilities, exploits, and findings in a report. This report explains what was found and how it can be fixed.
- Remediation: Fixing the vulnerabilities or taking action to protect the system from being exploited. It's about making the system more secure.

SUMMARY OF FINDING

In the VAPT (Vulnerability Assessment and Penetration Testing) report, this section typically includes key insights and results gathered from the assessment. Details of the vulnerability, severity, risk, impact and ways to solve it. This webpage contains several vulnerabilities that need to be focused and fixed.

Risk Calculation and Classification

Vulnerability Assessment and Penetration Testing report use risk calculation and classification to identify and categorize vulnerabilities based on the potential impact and severity of exploitation. This prioritizes remediation efforts, focusing on the most critical issues first. This is done based on vulnerability rating scales.

Risk Summary Table

Risk Level	Count
Critical	2
High	4
Medium	2
Low	2
Info	0
Total	10

Rating

Critical	High	Medium	Low	Info
----------	------	--------	-----	------

DETAILED FINDINGS

No.	Index Of Detailed Findings	Risk	Page No.
1	SQL Injection (DVWA)	Critical	15 - 16
2	Command Injection (DVWA)	Critical	17 - 18
3	File Upload (Web Shell) (DVWA)	High	19 - 20
4	Information Disclosure via Help Icon (Zero Web App)	High	21 - 22
5	Insecure Direct Object Reference (IDOR) (Zero Web App)	High	23 - 24
6	Unprotected Admin Functionality (PortSwigger Lab)	High	25 - 26
7	Stored XSS in Transaction Description (Zero Web App)	Medium	27 - 28
8	Flawed Business Logic in Coupon Codes (PortSwigger Lab)	Medium	29 - 30
9	Information Disclosure via Error Message (PortSwigger Lab)	Low	31 - 32
10	Directory Enumeration (Zero Web App)	Low	33 - 34

Finding 1: SQL Injection (DVWA)

Finding Summary	
Vulnerability	SQL Injection (DVWA)
Risk	Critical
Severity	Critical
CVSS Base Score	9.8
Status	Open
Occurrence	1
Affected Components	Login form (http://localhost/DVWA/vulnerabilities/sqli/)

Details

The SQL injection vulnerability was discovered in the login form of DVWA, where user input is not properly sanitized before being processed by the database query. During the penetration testing phase, a malicious payload was successfully injected into the username field, which resulted in the application returning all user records from the database without requiring valid authentication credentials. This vulnerability occurs because the application directly concatenates user input into SQL queries without using parameterized statements or input validation mechanisms. The successful exploitation demonstrates that the application's backend database is directly accessible through manipulated SQL queries.


Impact

- Attackers can bypass authentication, gaining unauthorized access to the entire application.
- Sensitive database records, including user credentials and personal data, can be extracted.
- Attackers may modify or delete critical data, disrupting application functionality.
- Potential for privilege escalation to administrative accounts, compromising system integrity.
- Exposure of sensitive data could lead to regulatory penalties (e.g., GDPR violations).

Suggested Remediation

- Implement parameterized queries or prepared statements to prevent SQL injection.
- Sanitize and validate all user inputs before processing in SQL queries.
- Use an ORM (e.g., SQLAlchemy) to abstract database interactions safely.
- Regularly audit database queries for vulnerabilities using tools like SQLMap.
- Restrict database permissions to limit the impact of potential injections.

Proof of Concept



Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

Authorisation Bypass

Open HTTP Redirect

Cryptography

API

DVWA Security

PHP Info

About

Vulnerability: SQL Injection

User ID:

Submit

ID: 1' OR '1'='1
First name: admin
Surname: admin

ID: 1' OR '1'='1
First name: Gordon
Surname: Brown

ID: 1' OR '1'='1
First name: Hack
Surname: Me

ID: 1' OR '1'='1
First name: Pablo
Surname: Picasso

ID: 1' OR '1'='1
First name: Bob
Surname: Smith

More Information

- https://en.wikipedia.org/wiki/SQL_injection
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- https://owasp.org/www-community/attacks/SQL_injection
- <https://bobby-tables.com/>

Finding 2: Command Injection (DVWA)

Finding Summary	
Vulnerability	Command Injection (DVWA)
Risk	Critical
Severity	Critical
CVSS Base Score	9.8
Status	Open
Occurrence	1
Affected Components	Ping utility (http://localhost/DVWA/vulnerabilities/exec/)

Details

The command injection vulnerability was identified in DVWA's ping utility function, which allows users to input IP addresses for network connectivity testing. The application fails to properly validate and sanitize user input before executing system commands on the underlying operating system. Through the use of command chaining operators in the input field, it was possible to execute arbitrary system commands beyond the intended ping functionality. This vulnerability demonstrates a critical security flaw where user-controlled input is directly passed to system command execution functions without adequate input filtering or validation.

Impact

- Allows arbitrary command execution, leading to potential full server compromise.
- Attackers can install backdoors or malware, ensuring persistent access.
- Sensitive system files and configurations can be accessed or modified.
- Potential to disrupt services, causing downtime and financial loss.
- Could facilitate lateral movement to other systems in the network.

Suggested Remediation

- Validate and sanitize all user inputs to prevent command injection.
- Use allow-lists to restrict inputs to expected formats (e.g., IP addresses only).
- Avoid executing system commands with user-supplied data; use safer APIs.
- Implement least privilege for the web server user to limit damage.
- Monitor and log system commands to detect and respond to suspicious activity.

Proof of Concept

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

Authorisation Bypass

Open HTTP Redirect

Cryptography

API

DVWA Security

PHP Info

Vulnerability: Command Injection

Ping a device

Enter an IP address:

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.  
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.051 ms  
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.030 ms  
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.031 ms  
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.039 ms  
  
--- 127.0.0.1 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3050ms  
rtt min/avg/max/mdev = 0.030/0.037/0.051/0.008 ms  
www-data
```

More Information

- <https://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/osif/>
- https://owasp.org/www-community/attacks/Command_Injection

Finding 3: File Upload (Web Shell) (DVWA)

Finding Summary	
Vulnerability	File Upload (Web Shell) (DVWA)
Risk	High
Severity	High
CVSS Base Score	8.6
Status	Open
Occurrence	1
Affected Components	File upload functionality (http://localhost/DVWA/vulnerabilities/upload/)

Details

The file upload vulnerability was discovered in DVWA's file upload functionality, which fails to implement proper file type validation and security controls. The application accepts arbitrary file types without adequately checking file extensions, MIME types, or file content, allowing malicious files to be uploaded to the web server. During testing, a PHP web shell was successfully uploaded through the vulnerable upload feature, granting remote command execution capabilities on the target system. This vulnerability is particularly dangerous because it provides attackers with a persistent backdoor into the system, allowing them to execute commands and potentially escalate privileges.

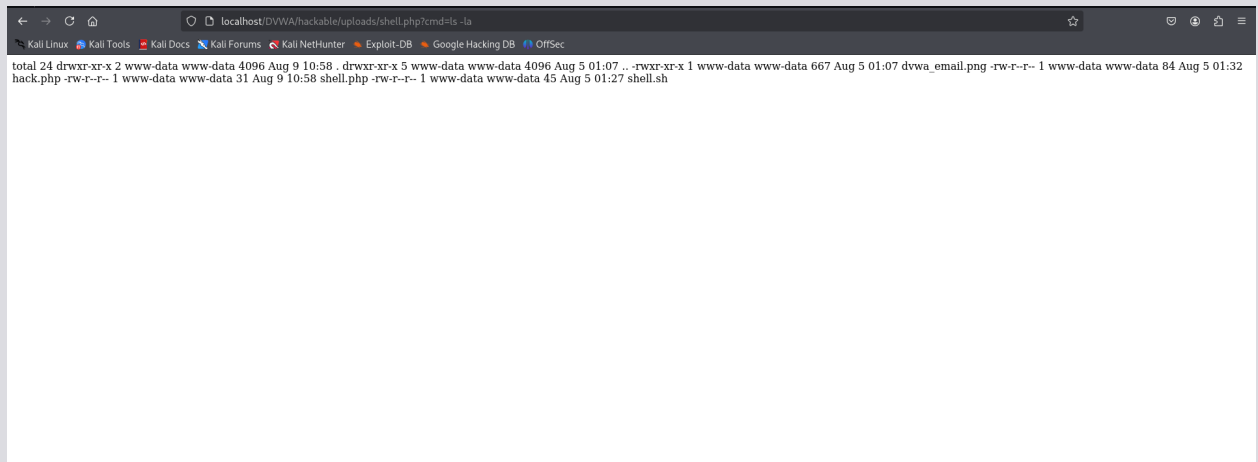
Impact

- Attackers can execute arbitrary code, gaining full control over the server.
- Sensitive data on the server, such as configuration files, can be accessed.
- Potential to deploy ransomware or other malicious payloads.
- Compromised servers could be used to attack other systems or users.
- Reputational damage due to compromised services or data breaches.

Suggested Remediation

- Restrict file uploads to specific, safe file types (e.g., images only).
- Validate file MIME types and content on the server side.
- Store uploaded files outside the webroot to prevent direct execution.
- Scan uploaded files for malicious content using antivirus solutions.
- Implement strict file permissions to limit access to uploaded files.

Proof of Concept



The screenshot shows a web browser window with the address bar displaying `localhost/DVWA/hackable/uploads/shell.php?cmd=ls -la`. The browser's tab bar includes several tabs: "Kali Linux", "Kali Tools", "Kali Docs", "Kali Forums", "Kali NetHunter", "Exploit-DB", "Google Hacking DB", and "OffSec". The main content area of the browser displays the output of the `ls -la` command, showing a directory listing of files and folders. The output is as follows:

```
total 24 drwxr-xr-x 2 www-data www-data 4096 Aug 9 10:58 . drwxr-xr-x 5 www-data www-data 4096 Aug 5 01:07 .. -rwxr-xr-x 1 www-data www-data 667 Aug 5 01:07 dvwa_email.png -rw-r--r-- 1 www-data www-data 84 Aug 5 01:32 hack.php -rw-r--r-- 1 www-data www-data 31 Aug 9 10:58 shell.php -rw-r--r-- 1 www-data www-data 45 Aug 5 01:27 shell.sh
```

Finding 4: Information Disclosure via Help Icon (Zero Web App)

Finding Summary	
Vulnerability	Information Disclosure via Help Icon (Zero Web App)
Risk	High
Severity	High
CVSS Base Score	7.5
Status	Open
Occurrence	1
Affected Components	Login page (http://zero.webappsecurity.com/login.html)

Details

The information disclosure vulnerability was found on the Zero Web App's login page, where sensitive authentication credentials are exposed through a help icon tooltip or similar user interface element. This critical security flaw reveals actual username and password combinations that can be used to access legitimate user accounts within the application. The vulnerability occurs due to poor security practices in the application's front-end code, where sensitive information is embedded in client-side elements accessible to anyone viewing the page source. This type of vulnerability provides immediate and direct access to user accounts without requiring sophisticated attack techniques.

Impact

- Immediate unauthorized access to user accounts without brute-forcing.
- Exposure of sensitive user data, such as financial or personal information.
- Potential for account takeover, leading to fraudulent activities.
- Erosion of user trust and reputational damage to the organization.
- Could enable further attacks, such as privilege escalation or data theft.

Suggested Remediation

- Remove sensitive information from client-side code and help tooltips.
- Implement secure password storage using hashing (e.g., bcrypt).
- Use generic help messages that do not expose credentials.
- Conduct regular code reviews to identify and remove sensitive data leaks.
- Educate developers on secure coding practices to prevent similar issues.

Proof of Concept

Zero Bank

Log in to ZeroBank

Login

?

Login/Password - username/password

Password

Keep me signed in

☐

Sign in

[Forgot your password ?](#)

Download Weblinspect

Terms of Use

Contact Micro Focus

Privacy Statement

Finding 5: Insecure Direct Object Reference (IDOR) (Zero Web App)

Finding Summary	
Vulnerability	Insecure Direct Object Reference (IDOR) (Zero Web App)
Risk	High
Severity	High
CVSS Base Score	7.7
Status	Open
Occurrence	1
Affected Components	Account summary (http://zero.webappsecurity.com/bank/account-activity.html?accountId=1)

Details

The Insecure Direct Object Reference (IDOR) vulnerability was discovered in the Zero Web App's account activity functionality, where the application fails to implement proper authorization checks for accessing user account information. The vulnerability manifests through direct object references in URL parameters, specifically the `accountId` parameter, which can be manipulated to access other users' sensitive financial data. During testing, it was possible to modify the `accountId` value in the URL to view transaction histories and account balances belonging to different users. This vulnerability occurs because the application relies solely on client-side access controls and does not perform server-side verification to ensure authenticated users can only access their own data.

Impact

- Unauthorized access to other users' sensitive financial data.
- Potential for data manipulation, leading to financial fraud.
- Exposure of personal information, violating user privacy.
- Regulatory penalties due to non-compliance with data protection laws.
- Loss of customer trust and potential legal action from affected users.

Suggested Remediation

- Implement server-side access control checks for every object request.
- Use indirect references (e.g., tokens) instead of direct IDs in URLs.
- Validate user permissions before serving sensitive data.
- Log access attempts to detect and respond to unauthorized requests.
- Regularly test for IDOR vulnerabilities using automated scanners.

Proof of Exploit

The screenshot shows a web browser window with the URL `http://zero.webappsecurity.com/bank/account-activity.html?accountId=2`. The browser's address bar and tabs are visible at the top. The page itself is titled "Zero Bank" and features a navigation menu with links: "Account Summary", "Account Activity", "Transfer Funds", "Pay Bills", "My Money Map", and "Online Statements". The "Account Activity" link is currently selected. Below the navigation menu, there are two tabs: "Show Transactions" (which is active) and "Find Transactions". Under the "Show Transactions" tab, the heading "Show Transactions" is followed by the instruction "Choose an account to view.". Below this is a dropdown menu labeled "Account" with "Checking" selected. A table of transactions is displayed below the dropdown menu. The table has four columns: "Date", "Description", "Deposit", and "Withdrawal". The transactions listed are: a check deposit of 186.7 on 2012-09-06, a telecom payment of 99.6 on 2012-09-05, and a car payment of 1548 on 2012-09-01.

Zero Bank

Search Settings username

Account Summary Account Activity Transfer Funds Pay Bills My Money Map Online Statements

Show Transactions Find Transactions

Show Transactions

Choose an account to view.

Account Checking

Date	Description	Deposit	Withdrawal
2012-09-06	CHECK DEPOSIT	186.7	
2012-09-05	TELECOM		99.6
2012-09-01	CAR PAYMENT		1548

Finding 6: Unprotected Admin Functionality (PortSwigger Lab)

Finding Summary	
Vulnerability	Unprotected Admin Functionality (PortSwigger Lab)
Risk	High
Severity	High
CVSS Base Score	7.7
Status	Open
Occurrence	1
Affected Components	/administrator-panel (https://0ae600a504901a6382268d8a00b900a0.web-security-academy.net/)

Details

The unprotected admin functionality vulnerability was identified in the PortSwigger Lab application, where administrative interfaces are accessible without proper authentication or authorization mechanisms. The vulnerability exists due to the complete absence of access controls on the administrator panel, allowing any user to directly navigate to administrative URLs and perform privileged operations. During testing, the admin interface was discovered to be publicly accessible through a predictable URL path, enabling unauthorized users to access sensitive administrative functions. This security flaw represents a critical oversight in the application's security architecture, where administrative functionality is exposed without any form of protection.

Impact

- Attackers can perform administrative actions, such as deleting users.
- Potential to disrupt application functionality by removing critical accounts.
- Exposure of sensitive administrative data or configurations.
- Could lead to full application compromise if admin accounts are abused.
- Reputational and operational damage due to unauthorized access.

Suggested Remediation


- Restrict admin panel access with strong authentication mechanisms.
- Implement IP-based allow-lists for administrative interfaces.
- Use role-based access control to limit admin functionality to authorized users.
- Monitor and log admin actions to detect suspicious activity.
- Regularly review admin panel configurations for security gaps.

Proof of Exploit

Zero Bank

Log in to ZeroBank

Login

 Login/Password - username/password

Password

Keep me signed in

☐

Sign in

[Forgot your password ?](#)

Download Websinspect

Terms of Use

Contact Micro Focus
Privacy Statement

Finding 7: Stored XSS in Transaction Description (Zero Web App)

Finding Summary	
Vulnerability	Stored XSS in Transaction Description (Zero Web App)
Risk	Medium
Severity	Medium
CVSS Base Score	6.1
Status	Open
Occurrence	1
Affected Components	Money transfer form (http://zero.webappsecurity.com/bank/transfer-funds.html)

Details

The stored Cross-Site Scripting (XSS) vulnerability was discovered in the Zero Web App's money transfer functionality, specifically within the transaction description field that fails to properly sanitize and validate user input. This vulnerability allows malicious JavaScript code to be permanently stored in the application's database and executed whenever the infected content is viewed by other users. During testing, malicious JavaScript payloads were successfully injected into the transaction description field, which were then executed when the transaction history was accessed. The vulnerability occurs due to insufficient input validation and output encoding mechanisms, allowing attackers to inject client-side scripts that execute in other users' browsers.

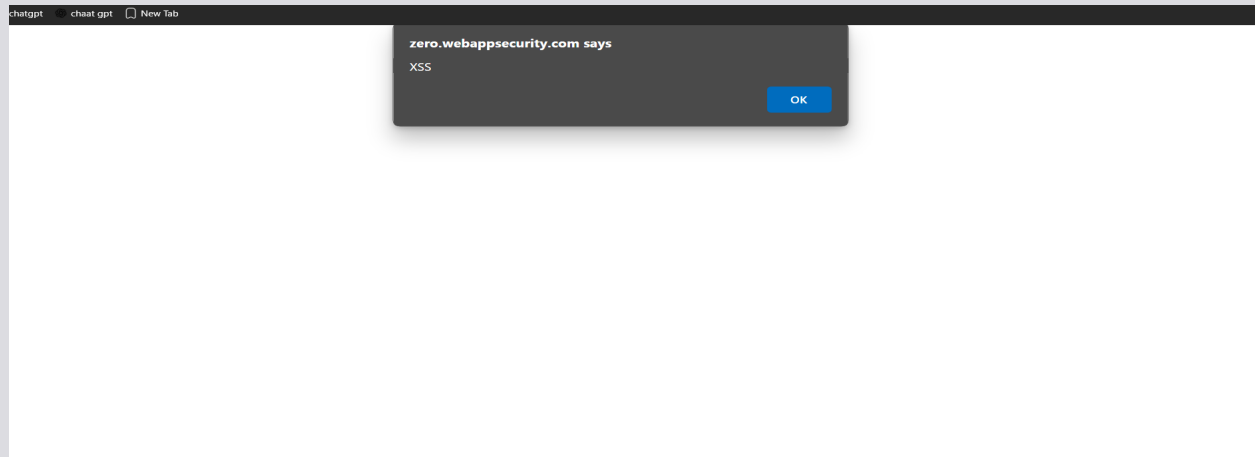
Impact

- Attackers can steal user cookies, enabling session hijacking.
- Malicious scripts can redirect users to phishing or malicious sites.
- Potential to manipulate displayed content, misleading users.
- Could facilitate social engineering attacks against users.
- Reputational damage due to compromised user experience.

Suggested Remediation

- Escape all user inputs before rendering to prevent script execution.
- Implement output encoding for all dynamic content displayed on pages.
- Apply a strong Content Security Policy (CSP) to restrict script sources.
- Regularly test for XSS vulnerabilities using tools like Burp Suite.
- Train developers on secure coding practices to prevent XSS issues.

Proof of Exploit



Finding 8: Flawed Business Logic in Coupon Codes (PortSwigger Lab)

Finding Summary	
Vulnerability	Flawed Business Logic in Coupon Codes (PortSwigger Lab)
Risk	Medium
Severity	Medium
CVSS Base Score	5.8
Status	Open
Occurrence	1
Affected Components	Checkout process (https://0a0000aa03b6b1d582f1bf48003e00e0.web-security-academy.net/cart)

Details

The flawed business logic vulnerability was identified in the PortSwigger Lab's e-commerce checkout process, where the coupon code validation mechanism contains critical design flaws that can be exploited to manipulate pricing calculations. The vulnerability stems from inadequate server-side validation of discount applications, allowing users to apply multiple coupon codes beyond intended limits. During testing, it was discovered that by alternating between different coupon codes, users could stack discounts and potentially reduce order totals to zero or negative values. This business logic flaw occurs because the application fails to properly track and validate coupon usage history and discount limitations during checkout.

Impact

- Financial loss due to abuse of discount mechanisms.
- Potential to exploit negative totals for fraudulent purchases.
- Disruption of revenue streams and pricing integrity.
- Customer dissatisfaction if pricing errors are publicized.
- Increased operational costs to address fraudulent transactions.

Suggested Remediation

- Implement server-side validation to enforce one-time coupon usage.
- Validate order totals to prevent negative or zero values.
- Log coupon usage to detect and block suspicious patterns.
- Test business logic thoroughly during development and updates.
- Use rate-limiting to prevent rapid coupon code submissions.

Proof of Exploit

Store credit:
\$100.00

Cart

Home | My account | 1

Name	Price	Quantity
Lightweight "133" Leather Jacket	\$1337.00	<div>- 1 +</div> <div>Remove</div>

Coupon:

Add coupon

Apply

Code	Reduction
NEWCUST5	-\$5.00
SIGNUP30	-\$401.10
NEWCUST5	-\$5.00
SIGNUP30	-\$401.10
NEWCUST5	-\$5.00
SIGNUP30	-\$401.10

Total: **\$118.70**

Finding 9: Information Disclosure via Error Message (PortSwigger Lab)

Finding Summary	
Vulnerability	Information Disclosure via Error Message (PortSwigger Lab)
Risk	Low
Severity	Low
CVSS Base Score	3.7
Status	Open
Occurrence	1
Affected Components	Login form (https://0a3c00e50376790780017bad00630044.web-security-academy.net/product?productId=1)

Details

The information disclosure vulnerability was discovered in the PortSwigger Lab's authentication system, where error messages provide different responses based on whether a username exists in the system or not. This vulnerability allows attackers to enumerate valid usernames by analyzing the subtle differences in error messages returned during failed login attempts. During testing, it was observed that the application returns distinct error messages when attempting to authenticate with valid versus non-existent usernames, creating information leakage. This type of vulnerability occurs when developers implement overly verbose error handling that inadvertently reveals information about the application's user base.

Impact

- Facilitates targeted brute-force attacks against valid usernames.
- Increases the risk of account compromise for known users.
- Could aid social engineering attacks by confirming user presence.
- Minor reputational damage due to perceived security weaknesses.
- May assist attackers in mapping the application's user base.

Suggested Remediation

- Use generic error messages for all authentication failures.
- Implement rate-limiting to deter brute-force attempts.
- Log failed login attempts to monitor for suspicious activity.
- Educate developers on secure error handling practices.
- Regularly test login forms for information disclosure issues.

Proof of Exploit

```
Kali Linux | Kali Tools | Kali Docs | Kali Forums | Kali NetHunter | Exploit-DB | Google Hacking DB | OJSSec
```

```
Internal Server Error: java.lang.NumberFormatException: For input string: ""he"  
at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:67)  
at java.base/java.lang.Integer.parseInt(Integer.java:647)  
at java.base/java.lang.Integer.parseInt(Integer.java:777)  
at lab.c.w.y.z(Unknown Source)  
at lab.o.g.p.z.l(Unknown Source)  
at lab.o.g.i.e.z.p.f(Unknown Source)  
at lab.o.g.i.e.lambda$handleHttpRequest$$0(Unknown Source)  
at x.x.s.t.lambda$run$$$inlined$lambda$1($Unknown Source)  
at x.x.s.t.$l(Unknown Source)  
at x.x.s.t.lambda$runCheckedFunction$4(Unknown Source)  
at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:289)  
at lab.o.g.p.i.e.yj(Unknown Source)  
at lab.server.k.a.m.(Unknown Source)  
at lab.o.g.v.B(Unknown Source)  
at lab.o.g.v.h(Unknown Source)  
at lab.server.k.a.k.p.b(Unknown Source)  
at lab.server.k.a.b.l.lambda$handleRequest$0(Unknown Source)  
at lab.c.t.r.q(Unknown Source)  
at lab.server.k.a.k.d.g(Unknown Source)  
at lab.server.k.a.r.V(Unknown Source)  
at x.x.s.t.lambda$run$$$inlined$lambda$1($Unknown Source)  
at x.x.s.t.$l(Unknown Source)  
at x.x.s.t.lambda$runCheckedFunction$4(Unknown Source)  
at lab.server.py.B(Unknown Source)  
at lab.server.k.a.r.G(Unknown Source)  
at lab.server.k.w.c(Unknown Source)  
at lab.server.k.u.c(Unknown Source)  
at lab.server.k.c.c(Unknown Source)  
at lab.server.gf.(Unknown Source)  
at lab.x.e.lambda$consume$0(Unknown Source)  
at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1144)  
at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:642)  
at java.base/java.lang.Thread.run(Thread.java:158)
```

```
Apache Struts 2.2.2.11
```

Finding 10: Directory Enumeration (Zero Web App)

Finding Summary	
Vulnerability	Directory Enumeration (Zero Web App)
Risk	Low
Severity	Low
CVSS Base Score	3.4
Status	Open
Occurrence	1
Affected Components	Web server directories (http://zero.webappsecurity.com/)

Details

The directory enumeration vulnerability was identified through automated scanning of the Zero Web App's web server, revealing the existence of hidden or unlinked directories that are not intended to be publicly accessible. Using directory brute-forcing techniques and common wordlists, several sensitive directories were discovered that could potentially contain administrative interfaces, configuration files, or other sensitive resources. The vulnerability exists because the web server is configured to allow directory browsing or does not properly restrict access to directories that should remain hidden. During the enumeration process, directories such as admin panels and backup folders were identified, which could provide attackers with additional attack vectors.

Impact

- Exposes hidden functionality that could be exploited further.
- Aids attackers in mapping the application's structure.
- Increases the risk of targeting sensitive endpoints.
- Minor reputational impact due to exposed directories.
- Could lead to discovery of unprotected admin panels or files.

Suggested Remediation

- Disable directory listing on the web server (e.g., Apache, Nginx).
- Restrict access to sensitive directories using server configurations.
- Implement robots.txt to discourage crawling of sensitive paths.
- Use security headers to enhance overall application security.
- Regularly scan for exposed directories using tools like Gobuster.

Proof of Exploit

```
(root@kali) ~/home/kali
# gobuster dir -u http://zero.webappsecurity.com/ -w /usr/share/wordlists/dirb/common.txt

=====
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:             http://zero.webappsecurity.com/
[+] Method:          GET
[+] Threads:         10
[+] Wordlist:         /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes: 404
[+] User Agent:       gobuster/3.6
[+] Timeout:         10s
=====
Starting gobuster in directory enumeration mode
=====
/admin           (Status: 302) [Size: 0] [--> /admin/]
/cgi-bin         (Status: 302) [Size: 0] [--> /cgi-bin/]
/cgi-bin/        (Status: 403) [Size: 961]
/docs            (Status: 302) [Size: 0] [--> /docs/]
/errors          (Status: 302) [Size: 0] [--> /errors/]
/help            (Status: 302) [Size: 0] [--> /help/]
/include         (Status: 302) [Size: 0] [--> /include/]
/index.html      (Status: 200) [Size: 12471]
```

CONCLUSION

The conclusion summarizes the findings and recommendations from the Vulnerability Assessment and Penetration Testing (VAPT) engagement. It emphasizes the importance of addressing identified vulnerabilities to enhance the overall security posture of the organization.

End Of Report
