LAB TITLE: WEB APPLICATION FORENSICS, BOOKWORLD CYBERATTACK.

STUDENT NAME: BRIAN NJIRU.

STUDENT ID: 2025/ACTI/6177.

COURSE NAME: ADVANCED NETWORK SECURITY OPERATIONS

INSTRUCTOR NAME: AMINU IDRIS.

DATE OF SUBMISSION 22TH APRIL 2025.

VERSION: 1.0

1. EXECUTIVE SUMMARY

The lab uncovered a possible SQL injection attempt on the bookworld.com website. An analysis of the provided pcap file provided adequate evidence of malicious activity and a possible data exfiltration from this IP address, 111.224.250.131.

Majority of the activities in this lab were performed inside Wireshark, these activities included following the http traffic to identify malicious activity and analyzing the requests made to identify anomalies like automated requests and SQL injection attempts.

Other activities in the lab that were performed outside the Wireshark software included geolocating the IP address using IP2location.

2. INVESTIGATION METHODOLOGY.

To identify any malicious activity in the pcap file, I followed the following steps to identify and gather evidence.

2.1 Network Traffic Forensics

2.1.1 Attacker Identification

I used Wireshark to sift through the captured traffic and isolate the outbound connections that seemed suspicious and corresponded to abnormal activity. These connections included evident SQL injection attempts and automated requests trying to uncover hidden directories. I identified the attacker's IP address by observing the SQL injection attempts and unusual query volumes. I then geolocated the attacker's Ip address and determined the country of origin.

2.1.2 Exploit Identification

I inspected the HTTP traffic to identify SQL injection attempts targeting the web application. I searched for common SQLi payloads within GET or POST parameters. I identified the exact request URL used by the attacker for the first SQL injection attempt.

2.1.3 Sensitive Data Exfiltration.

I inspected HTTP traffic to uncover patterns that suggested data extraction. I focused on trying to identify where the attacker retrieved user information. I inspected the traffic to identify the request URI used by the attacker to exfiltrate data from the database.

2.2 Attack Attribution and Exploitation Analysis

2.2.1 Malicious Scripts Identification.

I examined network traffic for any signs of file uploads or unusual POST requests containing payloads. I then examined to identify any malicious scripts that were uploaded to the server, potentially leading to web shell access.

2.2.2 Obscured Directories Access Attempts.

I investigated requests that accessed non-public directories to identify how and why the attacker tried to access the directories, possibly trying to access restricted areas of the web server.

2.2.3 Malicious Payload and Backdoor Deployment.

I analyzed the traffic to find any executed shell commends or abnormal POST requests containing code execution commands.

3. FINDINGS

I analyzed the pcap file and identified the IP address 111.224.250.131 as the attacker's IP address. The traffic involving the IP address contained anomalies that indicated malicious activity.

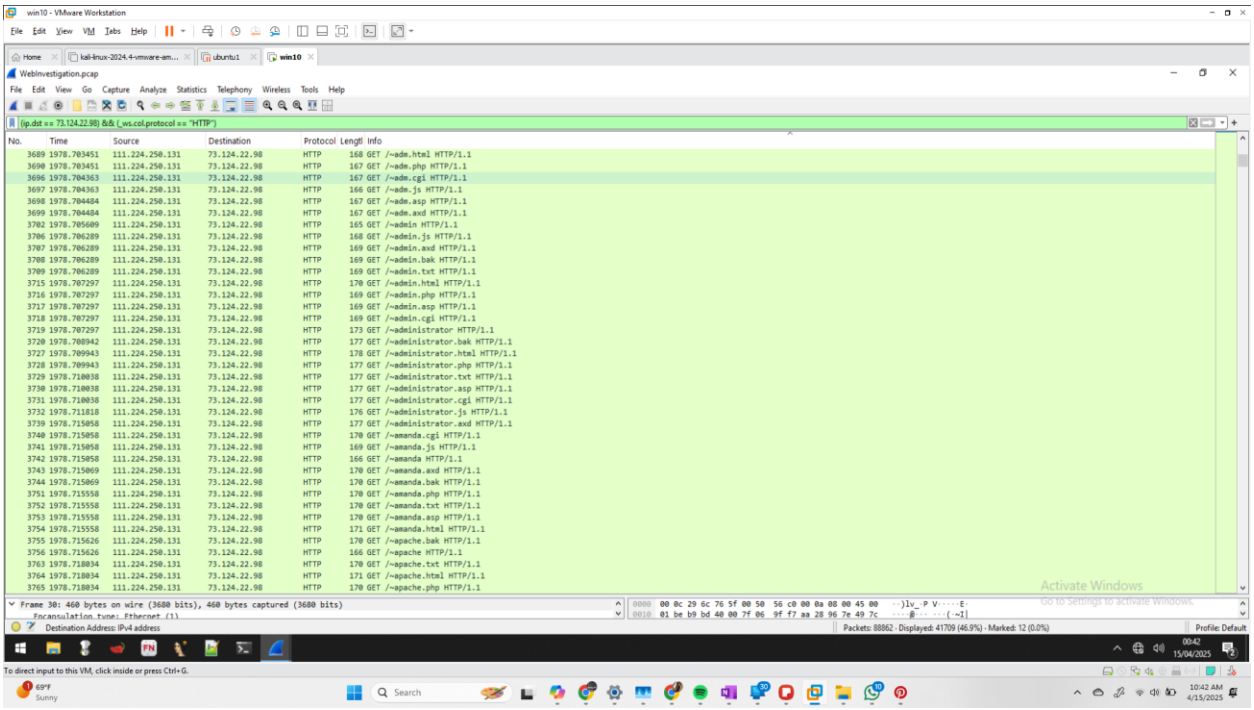


Figure 1: Malicious activity from the suspected IP address

I followed the TCP and HTTP stream of the traffic between the attacker's IP address and the web server. The attacker's IP address appears to have initially interacted with the website normally using expected search queries to search for book titles. I identified the first instance of an internal server error caused by an unusual search query. The anomaly in packet 347 and the response in packet 349 showed characteristics of an SQL error.

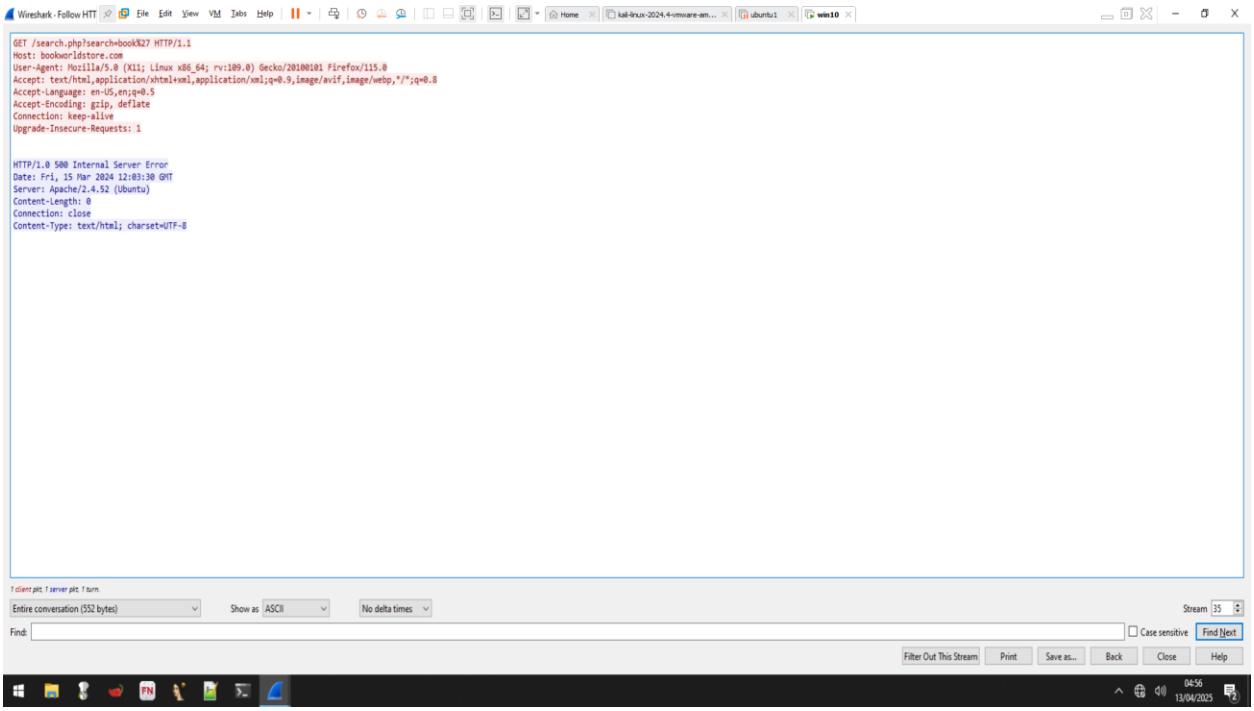


Figure 2: First instance of an SQLi error

The request contains '**%27**' which is the single quotation character. This request led to the web server responding with an internal server error. This served as an indicator to the attacker that the web server might be vulnerable to an SQL injection attack.

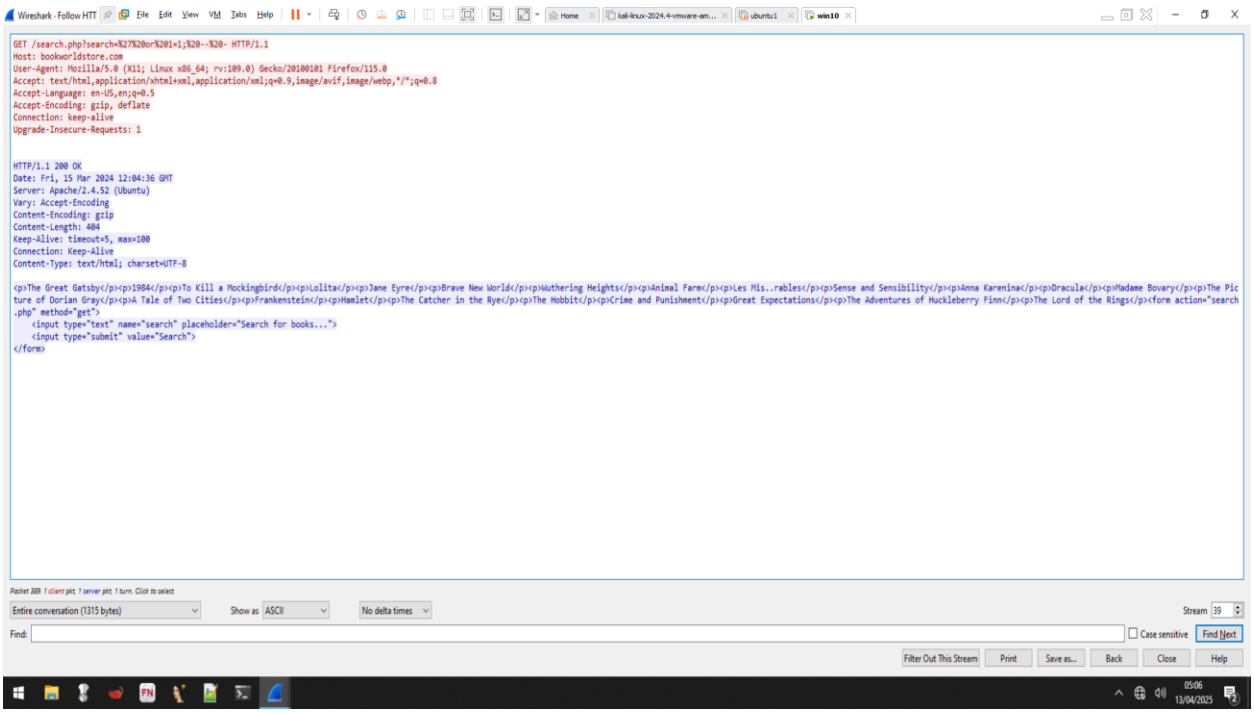
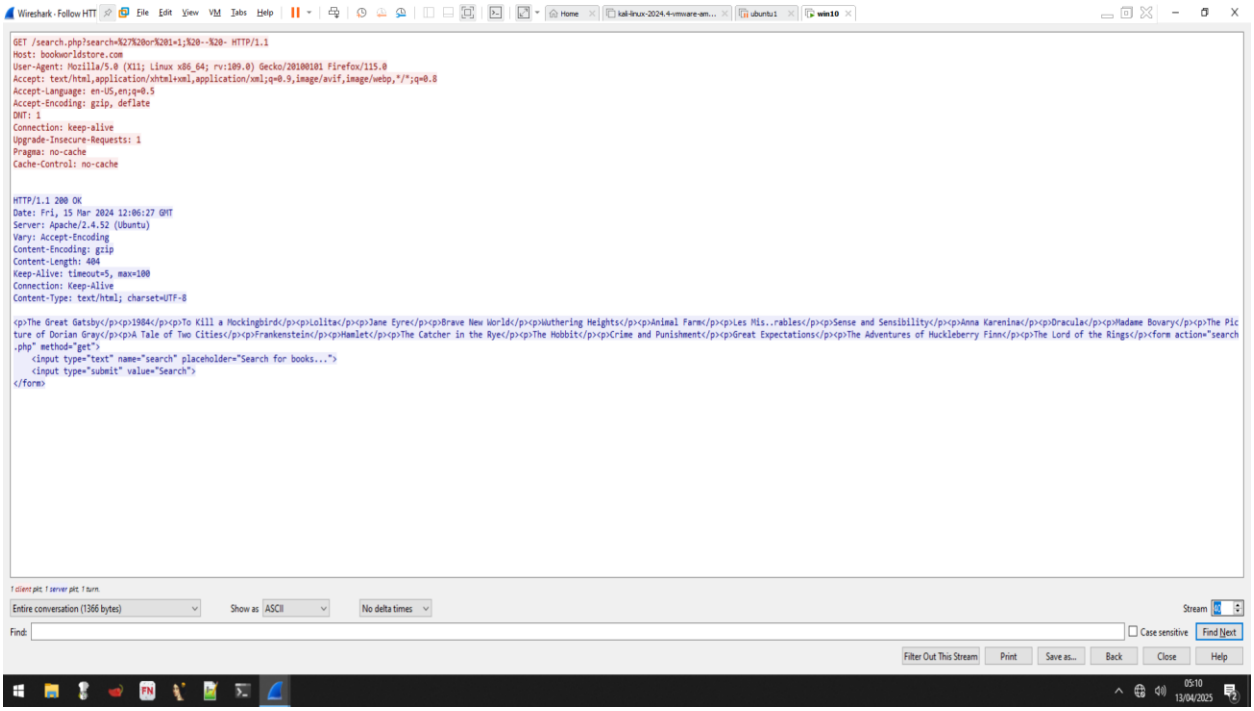


Figure 3: SQLi attempt without an internal server error

The attacker seems to have gotten a foothold after the SQLi attempts stop returning internal server errors.



The attacker sets the **DNT** to 1, the Do Not Track functionality is set and the attacker's activity will no longer be tracked. The attacker is gearing up for the actual attack.

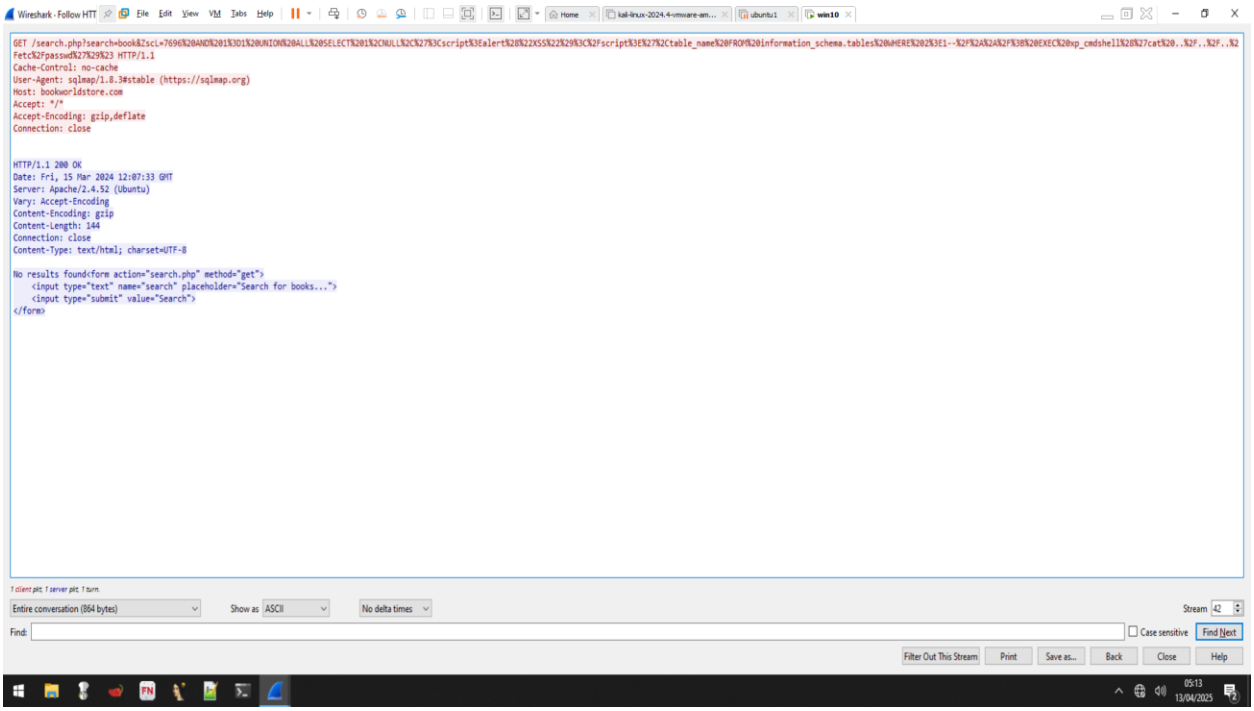


Figure 4: Multi-layered attack

The attacker proceeds to launch a multi-layered attack on the server. The attack request translates to the following text when decoded from the URL encoding:

GET/search.php?search=book&ZscL=7696 AND 1=1 UNION ALL SELECT 1,NULL,'<script>alert("XSS")</script>',table_name FROM information_schema.tables WHERE 2>1--//; EXEC xp_cmdshell('cat ../../../etc/passwd')#**

This request is a combined attack that leverages SQL injection, XSS (Cross-Site Scripting), and Remote Command Execution (RCE).

AND 1=1

UNION ALL SELECT 1,NULL,'<script>alert("XSS")</script>',table_name

FROM information_schema.tables WHERE 2>1--//**

This section is an SQL injection attempt. The aim is to extract data from the database (**information_schema.tables**) using a **UNION SELECT** and inject malicious scripts. **1=1** and **2>1** are tautologies to ensure the injected SQL is valid and executes. **--/**/** is used to comment out the rest of the legitimate SQL query and bypass some filters. The attacker is injecting a malicious script **<script>alert("XSS")</script>** to test for **XSS** vulnerability and try and extract table names from the information_schema.tables, which is a metadata table in MySQL.

<script>alert(“XSS”)</script>

The application would reflect database content directly into the HTML without sanitization, the attacker could replace the alert with session hijacking, keylogging or malicious redirects.

EXEC xp_cmdshell(‘cat ../../../etc/passwd’)#

This is a SQL Server-specific stored procedure used to execute system-level commands. The attacker attempts to run **cat ../../../etc/passwd** which is a Linux command. The term **xp_cmdshell** is Windows-only but **cat** is Unix-based. This suggests that the attacker is guessing or testing multiple environments. The path **/etc/passwd** contains system user information which is a common and classic privilege escalation target.

This is a multi-layered attack meant to test and exploit Database vulnerabilities (SQLi), Frontend sanitization flaws (XSS) and Backend command execution holes (RCE).

I geolocated the IP address 111.224.131 using various geolocating tools.

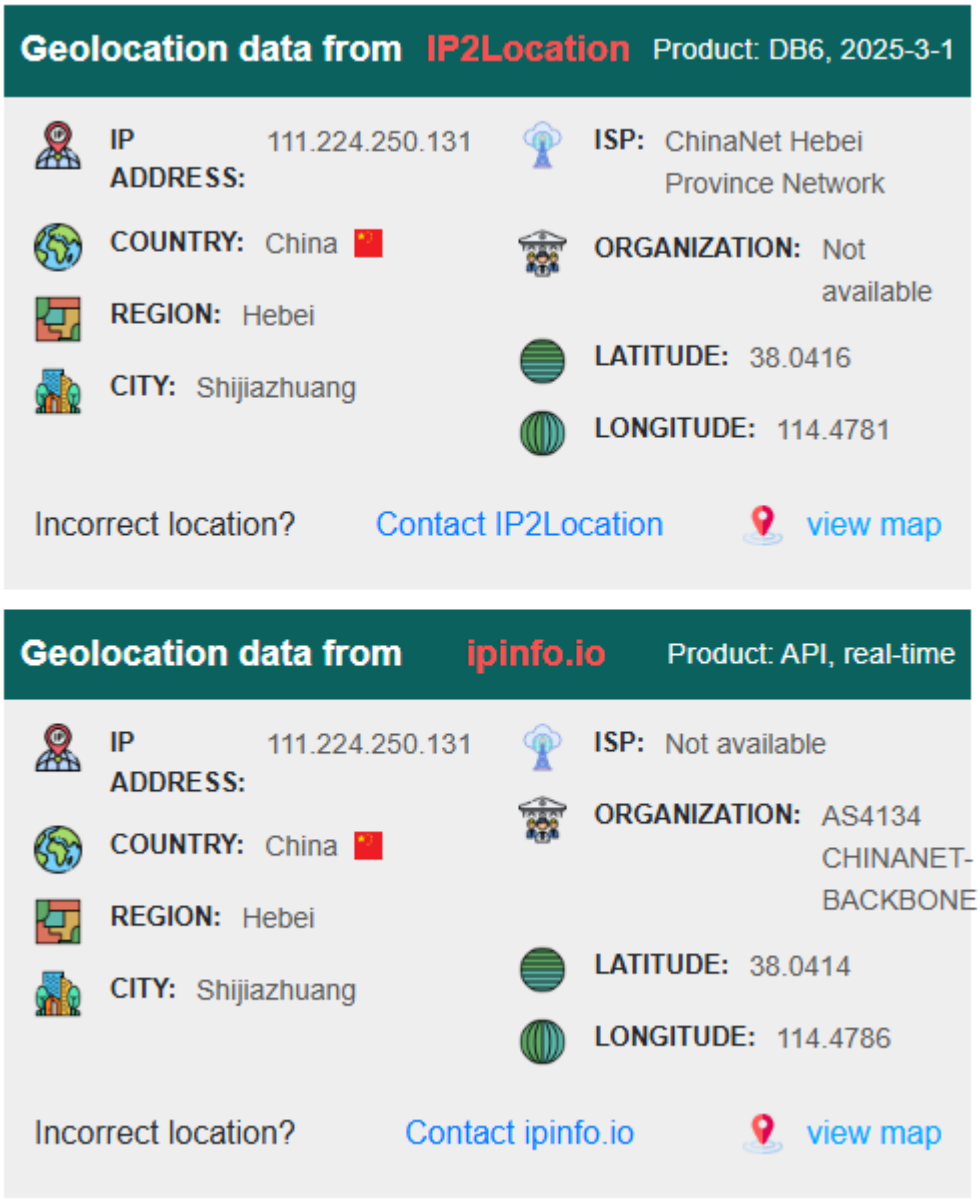


Figure 5: Geolocating data from IP2Location and ipinfo.io

The geolocation data reveals that the attacker's IP address is from China, specifically from the region of Hebei in the city of Shijiazhuang. The IP address has been identified as a source of spam and dictionary attacks by Project Honey Pot.

4. ATTACK ATTRIBUTION.

The attacker discovered a vulnerable parameter in **search.php** initiating a series of **UNION SELECT** payloads targeting **admin** and **customers** tables. Using clever SQL functions and hex obfuscation, the attacker exfiltrated an admin username-password pair. The attacker followed up with a query that leaked detailed customer information. The attacker likely used an automated script to enumerate hidden directories and admin endpoints. Upon discovering the admin login page, the attacker successfully authenticated using the stolen credentials. There is evidence suggesting a malicious file upload occurred, possibly to maintain access or pivot further. Bot-like traffic patterns indicate the use of automated reconnaissance tools post-compromise.

4.1 Initial Access

4.1.1 Exploit Public-Facing Application.

The attacker exploited an SQL injection vulnerability in **search.php** via crafted GET requests. These included **UNION ALL SELECT** that bypassed basic filters and returned data from the backend

4.2 Execution

4.2.1 Command and Scripting Interpreter: SQL

Multiple SQL payloads were executed through the vulnerable endpoint. Notably, the attacker used **JSON_ARRAYGG** and **CONCAT_WS** to structure sensitive data.

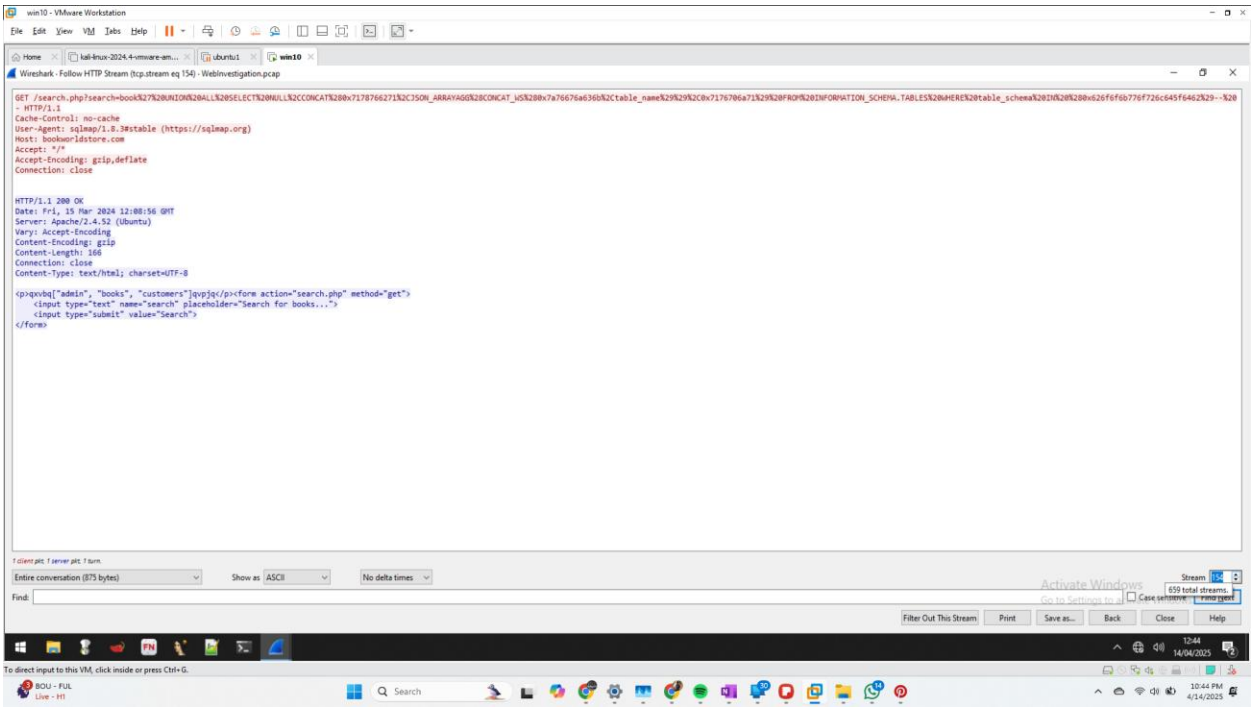


Figure 6: Attacker uses SQL payloads to structure sensitive data stolen from the web server.

4.3 Credential Access.

4.3.1 Credential Dumping: Application Database.

Credentials for an admin account were exfiltrated from the **admin** table via the following request

http GET /search.php?search=book' UNION ALL SELECT NULL, CONCAT(0x7178766271, JSON_ARRAYAGG(CONCAT_WS(0x7a76676a636b, id, password, username)), 0x7176706a71) FROM bookworld_db.admin-- -

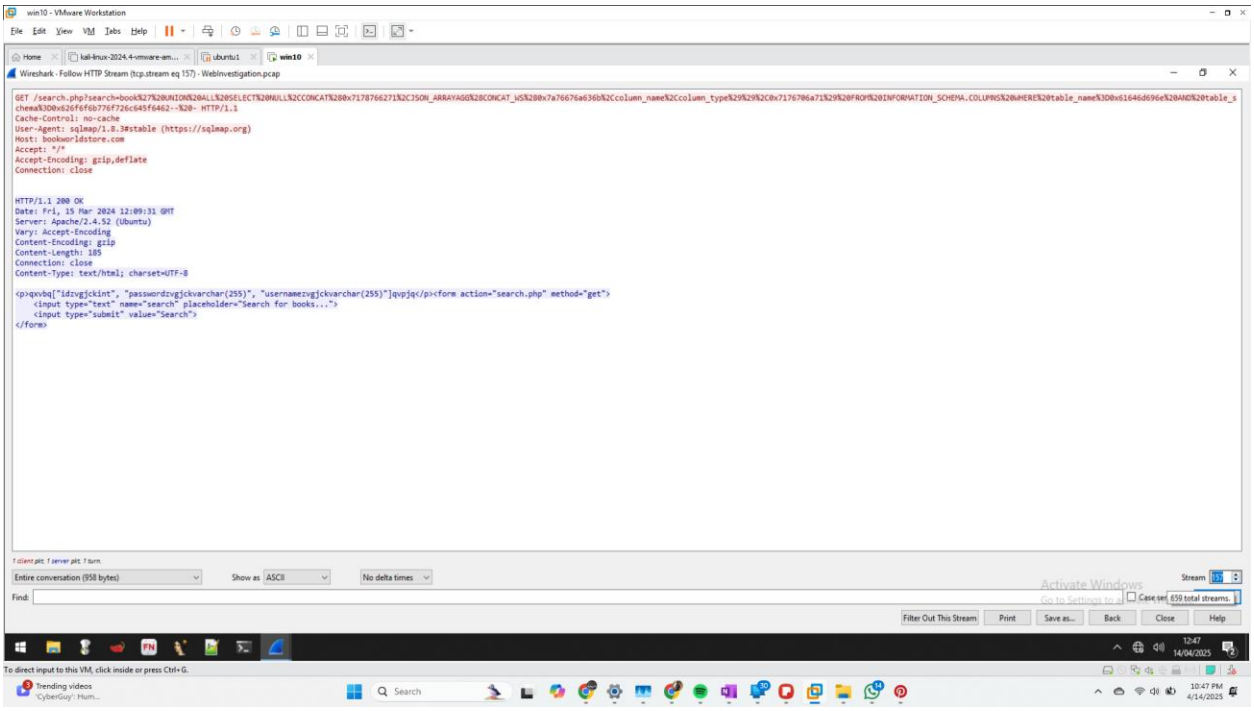


Figure 7: Attacker accesses the admin table and views the column names.

After the attacker obtained the column names, he/she proceeded to obtain sensitive data that contained login credentials.

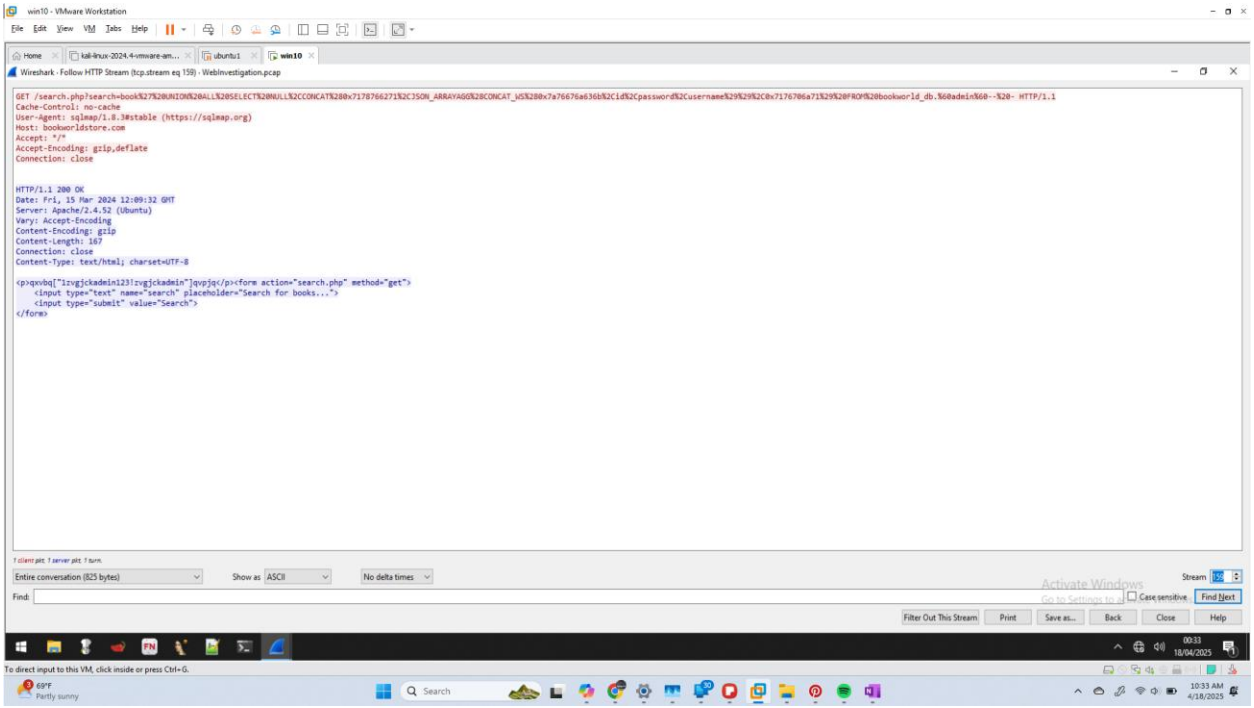


Figure 8: Attacker obtains administrative login credentials.

4.4 Collection

4.4.1 Data from Information Repositories.

The attacker accessed the **customers** table using the following request:

http GET /search.php?search=book' UNION ALL SELECT NULL, CONCAT(0x7178766271, JSON_ARRAYAGG(CONCAT_WS(0x7a76676a636b, address, email, first_name, id, last_name, phone)), 0x7176706a71) FROM bookworld_db.customers-- -

This returned a structured leak of customer information. This was evidence of systematic data harvesting.

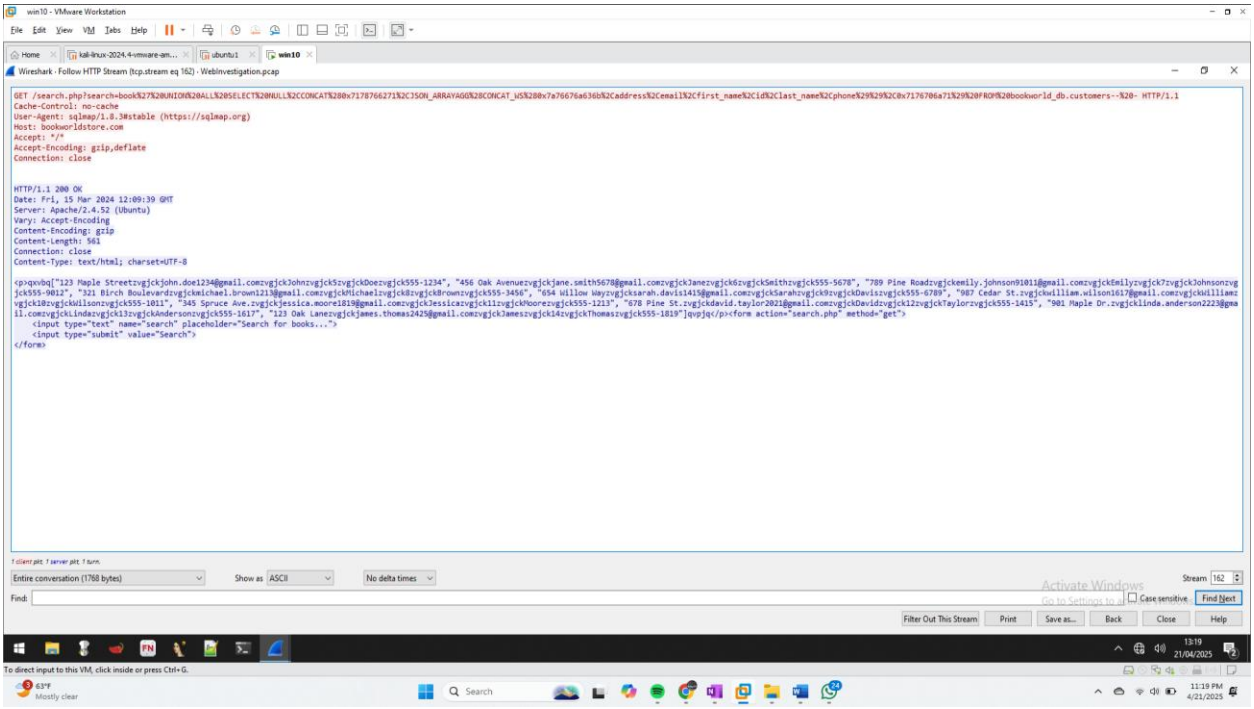


Figure 9: Attacker obtains sensitive customer information.

4.5 Persistence

4.5.1 Valid Accounts.

After retrieving valid credentials from the admin table, the attacker located the admin login portal and successfully logged in using the compromised account.

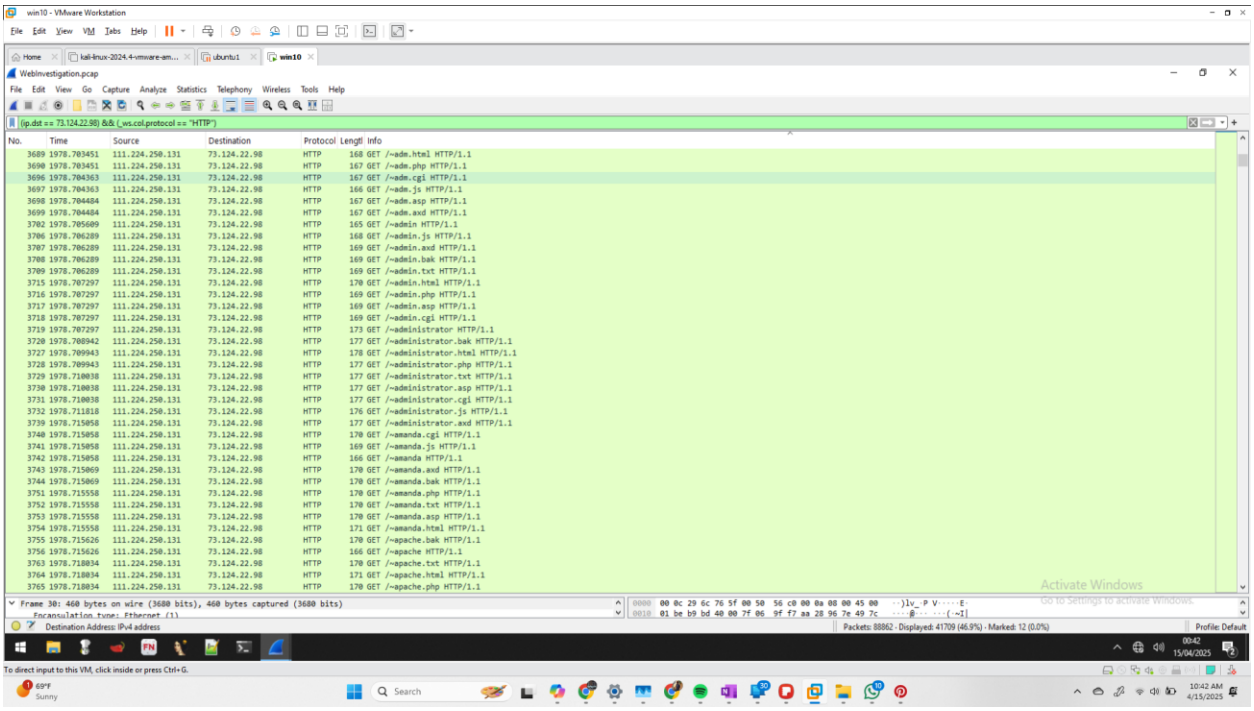


Figure 10: The attacker looks for hidden or obscure directories.

The attacker seems to have used a bot-driven approach to obtain the login portal. Figure 10 shows a sample of the requests recorded. These requests are milliseconds apart a d mechanically regular. This indicates a dictionary style path guessing approach and use of automated payloads was also encountered.

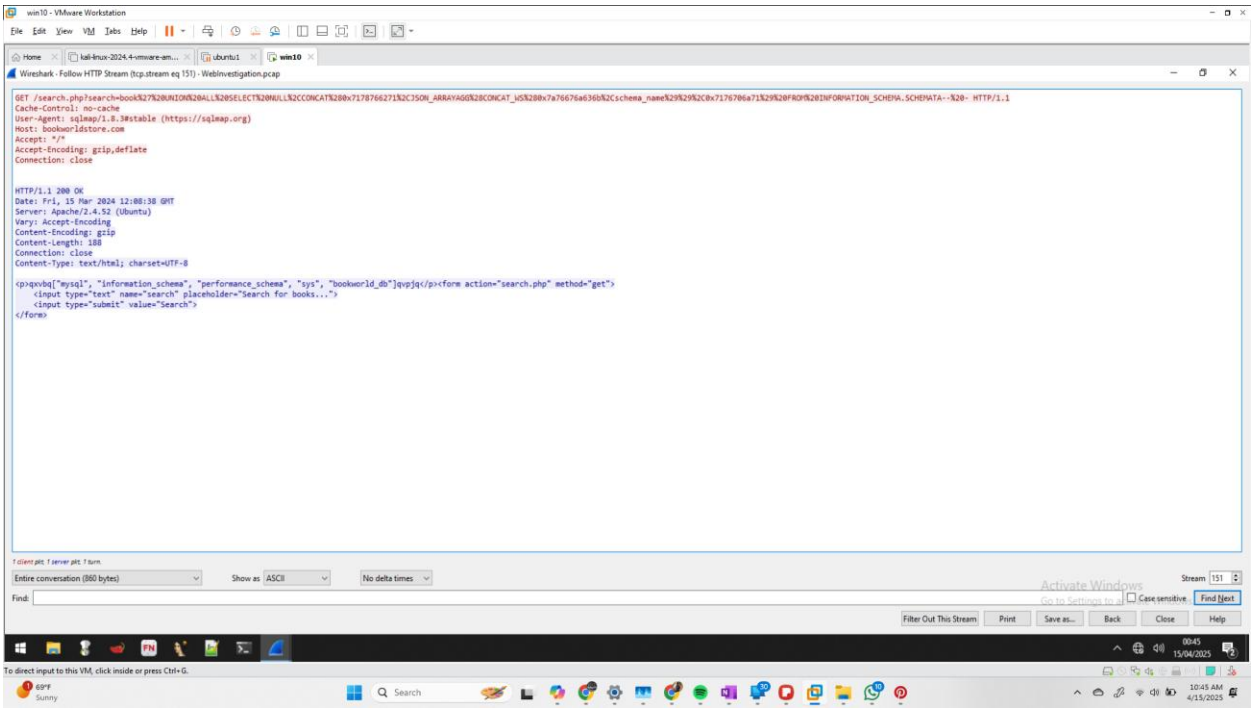


Figure 11: Attacker uses automated scripts or bots.

The User-Agent is **sqlmap/1.8.3#stable (https://sqlmap.org)**. This user agent is a sign of an automated attack or a bot, normal user agents are usually complex and long and often are from legitimate browsers. The heavy use of URL encoding also indicates bot activity.

5. IMPACT ASSESSMENT.

5.1.1 Data confidentiality breach.

The attacker successfully accessed and exfiltrated sensitive records from the backend database, including: admin credentials and customer PII such as full names and email addresses. These data leaks constitute a **severe confidentiality breach** under most protection laws.

5.1.2 Authentication compromise.

The attacker obtained a valid admin login credential pair and used it to access the administrative backend. This grants privileged access to backend functionality. Potential post-login actions may include: altering or deleting data, creating backdoor accounts, injecting malicious scripts or redirecting payloads and uploading additional malicious content. This escalates the threat from a SQL injection to a full account takeover, allowing the attacker to act as a legitimate administrator.

5.1.3 System Integrity Violation.

There are indicators that the attacker uploaded a malicious file and initiated directory scans possibly to: discover upload endpoints or access-sensitive directories, identify vulnerable configuration files or pivot to deeper systems.

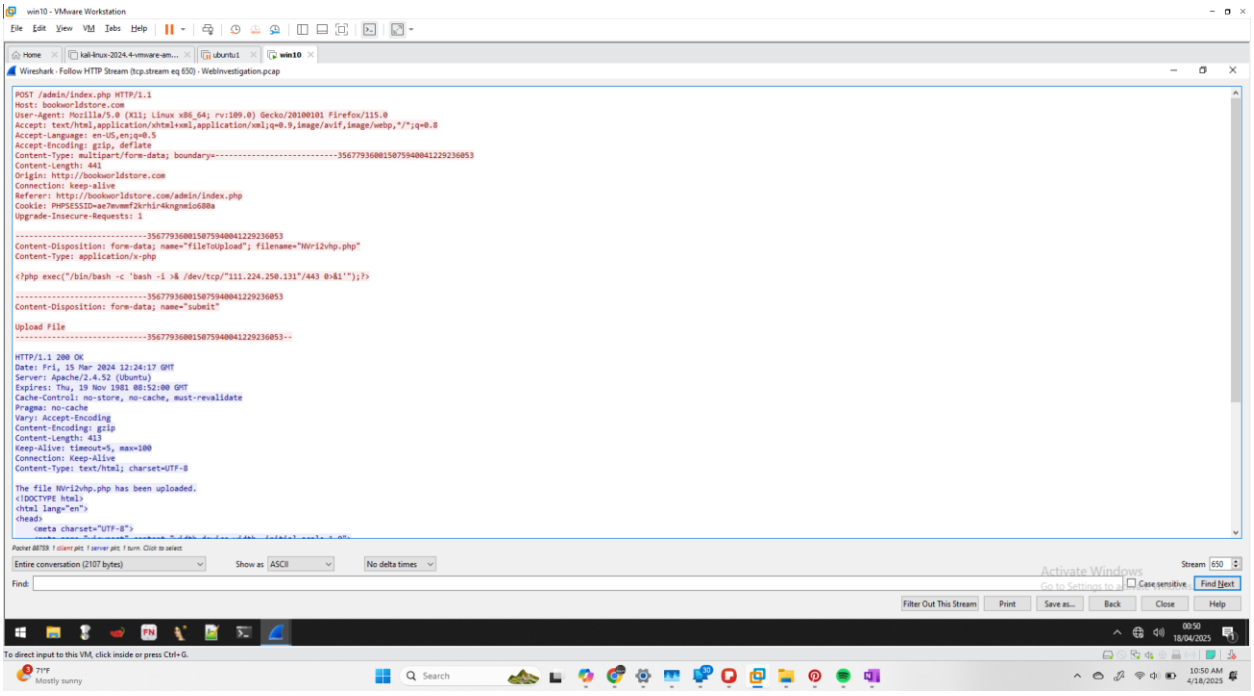


Figure 12: Attacker uploads a malicious file to the sever.

5.1.4 Automated Recon and Bot Activity.

Evidence suggests that the attacker deployed automated scripts or bots to: enumerate hidden directories, fuzz endpoints and attempt brute-force discovery. This increases the risk of: web server overload, exposure of forgotten or legacy files and identification of new vulnerabilities.

5.1.5 Reputational Damage

With customer and admin data exposed, BookWorld faces significant reputational risk including: loss of customer trust, negative media coverage and possible public disclosure of the breach on forums or dark web marketplaces.

5.1.6 Legal and regulatory Exposure

If BookWorld operates within jurisdictions enforcing data protection compliance, the following legal consequences may apply: Mandatory breach notification within a limited window, investigation by data protection authorities and fines or sanctions depending on negligence and data handling policies.

5.1.7 Financial Risk

The potential financial implications include: Costs of incident response and forensics, legal consultation and regulatory fines, customer compensation or credit monitoring services and infrastructure hardening and security upgrades.

5.1.8 Business Continuity Impact.

BookWorld systems could be further compromised resulting in potential downtime. Backdoors could have been installed, requiring full revalidation of system integrity. Future targeted attacks could occur if the attacker retains access or sells the data.

6. RECOMMENDATIONS.

6.1 Immediate Technical Containment.

6.1.1 Revocation of compromised admin credentials.

The compromised admin login credentials should be immediately disabled or reset. Stronger passwords and Multi-Factor Authentication should be used.

6.1.2 Termination of suspicious sessions.

Sessions that appear suspicious should be terminated. Any session tokens or cookies that could be active should be invalidated due to the compromise.

6.1.3 Isolation of infected systems.

The server should be isolated from the network to prevent further intrusion or propagation of the attack.

6.1.4 Inspection for persistence mechanisms

The server should be inspected to check for webshells, cron jobs, hidden users or modified **.htaccess/ .php** files.

6.1.5 Patching of SQL injection vulnerabilities .

All user inputs should be immediately sanitized and parameterized. Input validation and prepared statements should be employed.

6.2 Hardening and Prevention.

6.2.1 WAF Deployment.

A Web Application Firewall (WAF) such as ModSecurity or Cloudflare should be deployed to filter for malicious payloads.

6.2.2 File Upload Controls.

If uploads are allowed, implement strict MIME type checks, file extension whitelisting, anti-virus scanning and execution prevention.

6.2.3 Rate limiting and Bot protection.

Detect and block automated brute-force scans using rate-limiting CAPTCHA or bot mitigation tools.

6.2.4 Secure Configurations

Disable unused services restrict directory listing and limit file permissions.

6.2.5 Server-side Input Sanitization

Enforce server-side validation even if client-side validation exists. Sanitize all GET/POST parameters.

6.3 Authentication and Access Control.

6.3.1 Multi-Factor Authentication.

Require MFA for all admin-level accounts and backend systems.

6.3.2 Least Privilege Principle.

Restrict access to the database and admin portal to only essential roles. Use role-based access controls.

6.3.3 Account Lockout Policies.

Implement account lockout after repeated failed login attempts to prevent credential stuffing.

6.3.4 Credential Vaulting.

Store credentials securely using strong hashing algorithms.

6.4 Monitoring and Detection

6.4.1 Centralized logging.

Enable and centralize application, web server and database logs. Include request headers and response codes.

6.4.2 Intrusion Detection.

Deploy host and network-based intrusion detection systems.

6.4.3 SQLi Signature Detection.

Use pattern-matching to detect suspicious SQL keywords and structure in GET/POST parameters.

6.4.4 Anomaly Detection.

Identify deviations from normal user behavior.

7. CONCLUSION.

The forensic investigation of the **webinvestigation.pcap** file has confirmed that BookWorld's web application was actively targeted and compromised through a coordinated SQL injection attack originating from the IP address **111.224.250.131**. The attacker successfully extracted sensitive administrative and customer data, uploaded at least one malicious file and accessed the administrative login page using stolen credentials. Evidence suggests the use of automated tools or scripts to escalate the attack, likely through endpoint discovery and persistent probing.

This attack highlights critical weaknesses in input validation, access control and threat detection mechanisms within the current application stack. The compromise of both user and administrative data places BookWorld at significant risk of regulatory, reputational and financial consequences. Furthermore, the potential for post-exploitation persistence mechanisms such as backdoors or hidden scripts raises serious concerns regarding system integrity and ongoing exposure.

While the immediate threat has been identified, it is imperative that BookWorld moves swiftly to implement the recommended technical, organizational and procedural controls to restore system integrity, prevent recurrence and maintain customer trust.

The findings and analysis contained herein should guide both urgent remediation efforts and long-term improvements in the organization's security posture.