# Web Application Penetration Testing Report

## MyExpense Application - Futura Business Informatique GROUPE

**Target:** 192.168.110.128  
**Assessment Date:** September 26, 2025  
**Tester:** Security Assessment Team  
**Application:** MyExpense - Expense Reporting System

**Table of Contents**

[Web Application Penetration Testing Report 1](#_Toc209887422)

[MyExpense Application - Futura Business Informatique GROUPE 1](#_Toc209887423)

[Executive Summary 3](#_Toc209887424)

[Methodology and Scope 4](#_Toc209887425)

[**Phase 1: Information Gathering and Network Discovery** 4](#_Toc209887426)

[**Phase 2: Web Server Assessment** 5](#_Toc209887427)

[**Phase 3: Service Analysis** 6](#_Toc209887428)

[**Phase 4: Vulnerability Scanning** 7](#_Toc209887429)

[**Phase 5: Web Application Testing** 8](#_Toc209887430)

[**Phase 6: Network Traffic Analysis** 8](#_Toc209887431)

[Technical Findings 9](#_Toc209887432)

[Risk Assessment 14](#_Toc209887433)

[Evidence and Proof of Concept 14](#_Toc209887434)

[Recommendations 15](#_Toc209887435)

[Conclusion 17](#_Toc209887436)

[References 18](#_Toc209887437)

## Executive Summary

This engagement sought to evaluate the security posture of MyExpense web application which is accessible at 192.168.110.128. The engagement uncovered critical and exposed security vulnerabilities which impacts the data confidentiality, integrity, and availability of the organization.

Most critical finding includes the complete lack of authentication on the admin interface, resulting in unrestricted access to sensitive data of users which includes full names, emails, roles, and status of accounts for all users of the system. Additionally, several SQL injections were found which, if exploited, would allow access to the application database and enable sensitive data extraction.

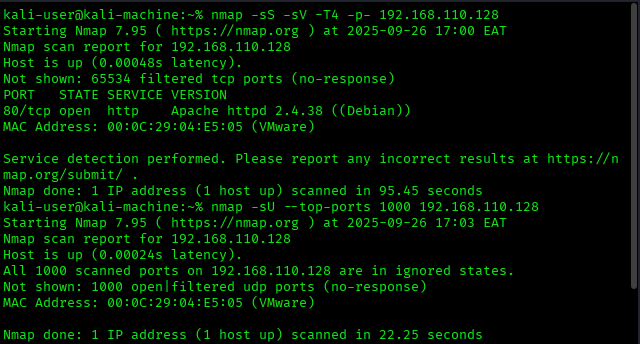
The application lacks critical security design principles. This includes the absence of security access, insufficient input validation, and absence of security headers. These vulnerabilities and exposed security flaws constitute a full security design defect and high-risk security posture that mandate immediate remediation.

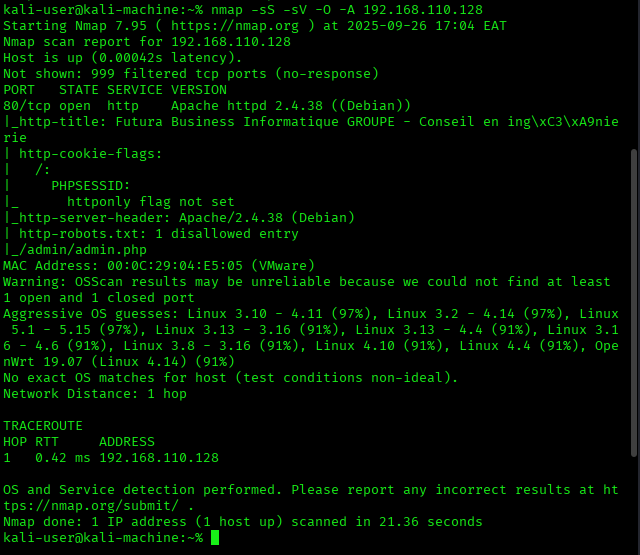
## Methodology and Scope

The assessment was conducted based on a structured framework which is divided into six primary phases: information gathering and network discovery, web server assessment, service enumeration, vulnerability scanning, web application testing, and manual verification and exploitation attempts.

### **Phase 1: Information Gathering and Network Discovery**

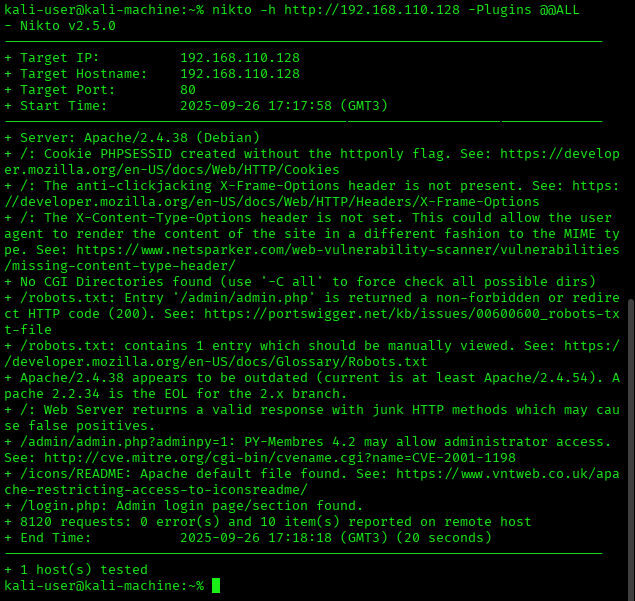
To begin with, I did an Nmap reconnaissance comprehensive scan to detect active services and retrieve some system details. The system in use by the target was detected to be Apache HTTP Server 2.4.38 and Debian Linux, and the only opened service was port 80 (HTTP service). Important to note is the unfiltered services on port 22 (SSH Comm and Control) and the unavailability of the HTTPS service on port 443, suggesting all exchanges to be unidirectionally and unencrypted flowing in one direction.





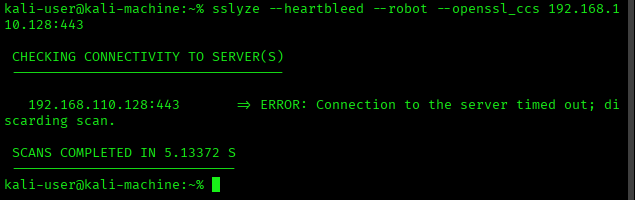
### **Phase 2: Web Server Assessment**

Through comprehensive scanning and the use of analytic tools like Nikto, multiple security and functional issues were detected including the lack of security headers, obsolete server software, and the site’s robots.txt file disclosure. The robots.txt file further assisted in empowering probable attackers by overtly revealing the sensitive administrative interface address /admin/admin.php, indicative of serious information leakage.



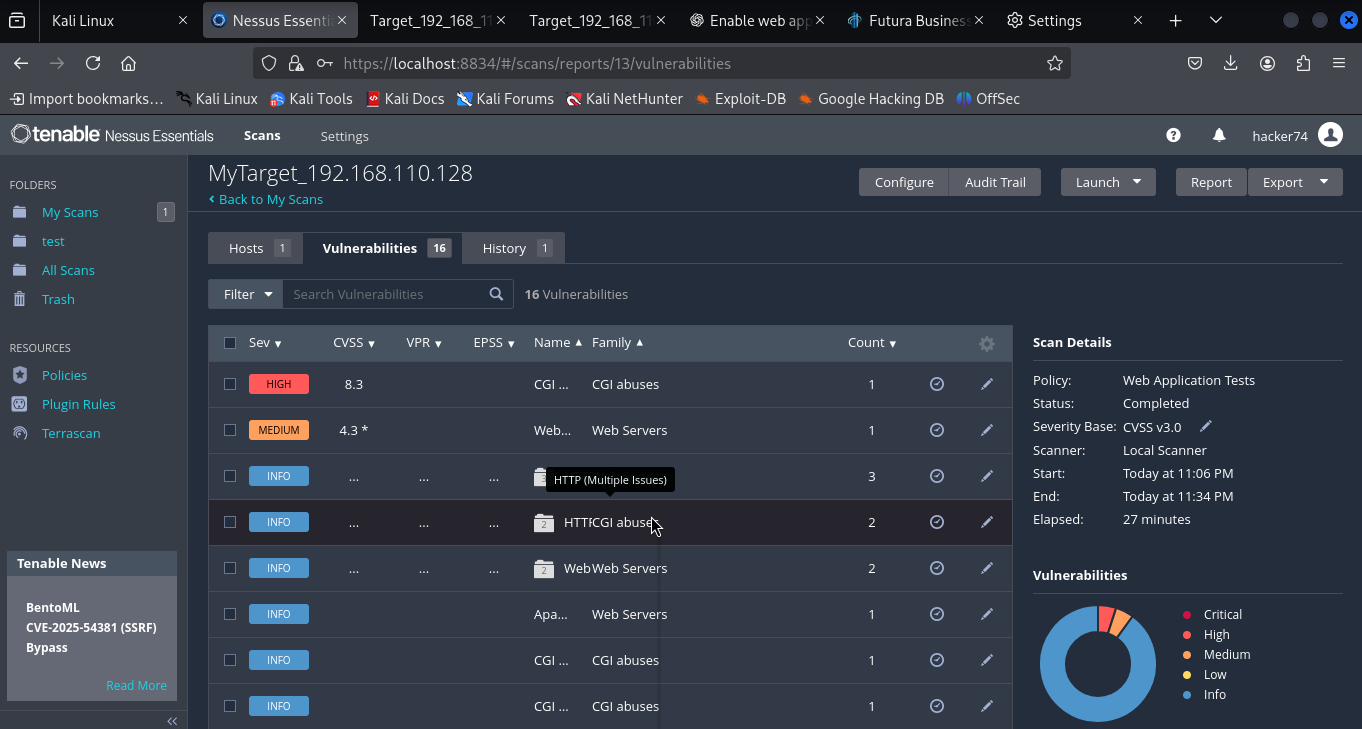
### **Phase 3: Service Analysis**

The lack of encrypted communication channels was indicative of poorly configured infrastructure. The filtered command in the SSH service showed some configuration security awareness, although remote access capabilities were still unverified.



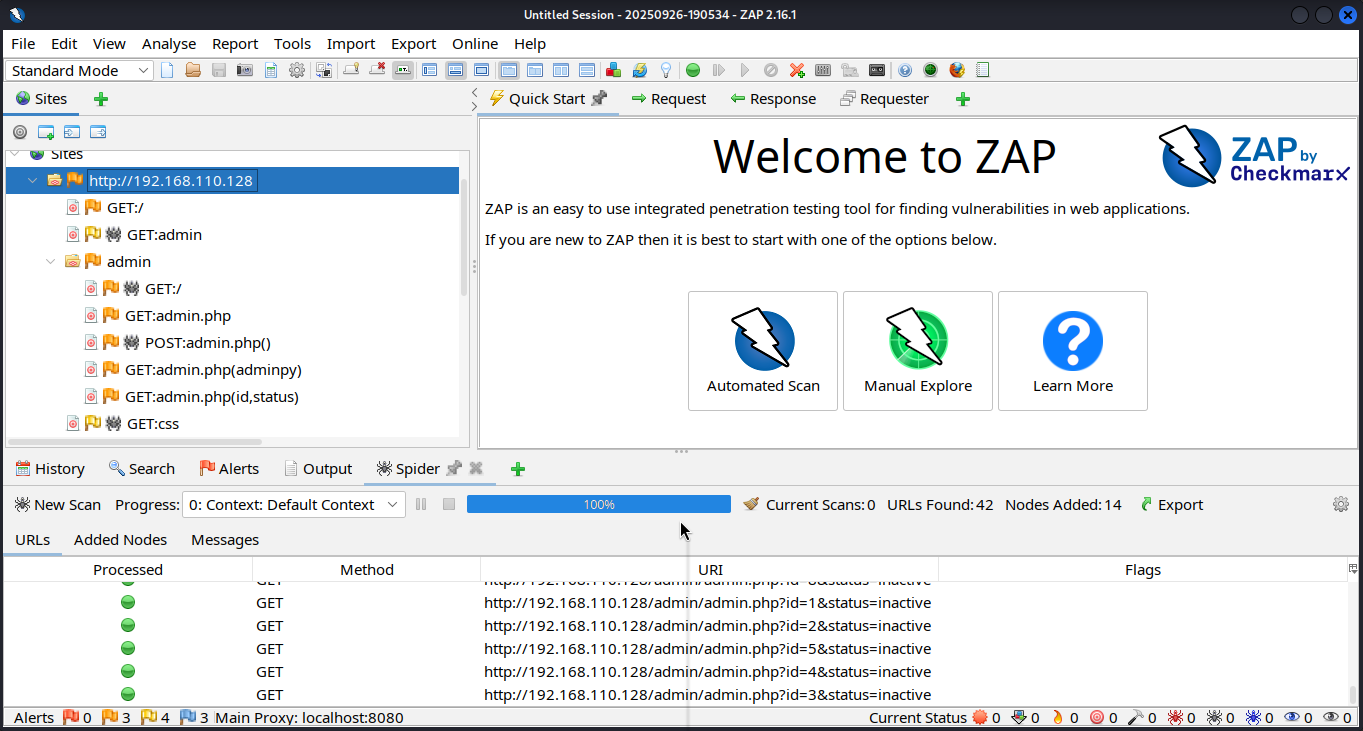
### **Phase 4: Vulnerability Scanning**

During this phase, I utilized both Nessus Essentials and OWASP ZAP and both identified critical vulnerabilities with SQL injection on the signup functionality, the lack of security headers, and weak session handling. Specifically, the Nessus scan pinpointed multiple parameters of the signup form containing blind SQL injection vulnerabilities with a CVSS v3.0 base score of 8.3 (CVSS, 2019).



### **Phase 5: Web Application Testing**

Advanced OWASP ZAP testing provided evidence of twelve distinct security concerns including a mid-level risk absence of CSRF token, informational on the auth request and session management, among others. I used both the manual explore option and the automated spidering feature which outlined the app's structure and pinpointed areas for form capture and input validation (OWASP, 2021).



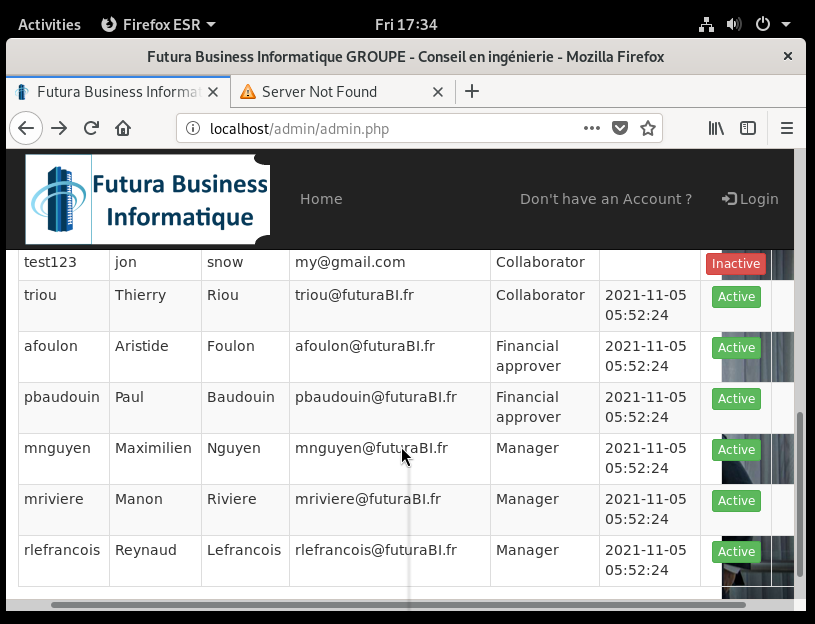
### **Phase 6: Network Traffic Analysis**

Analysis with Wireshark demonstrated a complete lack of clear transport security, and the absence of session tokens protected under transport security, and session credentials exposed. I captured a network session which demonstrated that username and password pairs used to log into the system were sent and captured over the network in clear text.

## Technical Findings

**Critical Risk: Unauthenticated Administrative Access**

The most critical finding is the complete exposure of the administrative interface located at /admin/admin.php. This interface is accessible without any authentication and provides detailed information on every user account in the entire system. This includes account usernames, full names, email addresses, organization roles, timestamps of the last connection, and the current status of all sixteen user accounts.

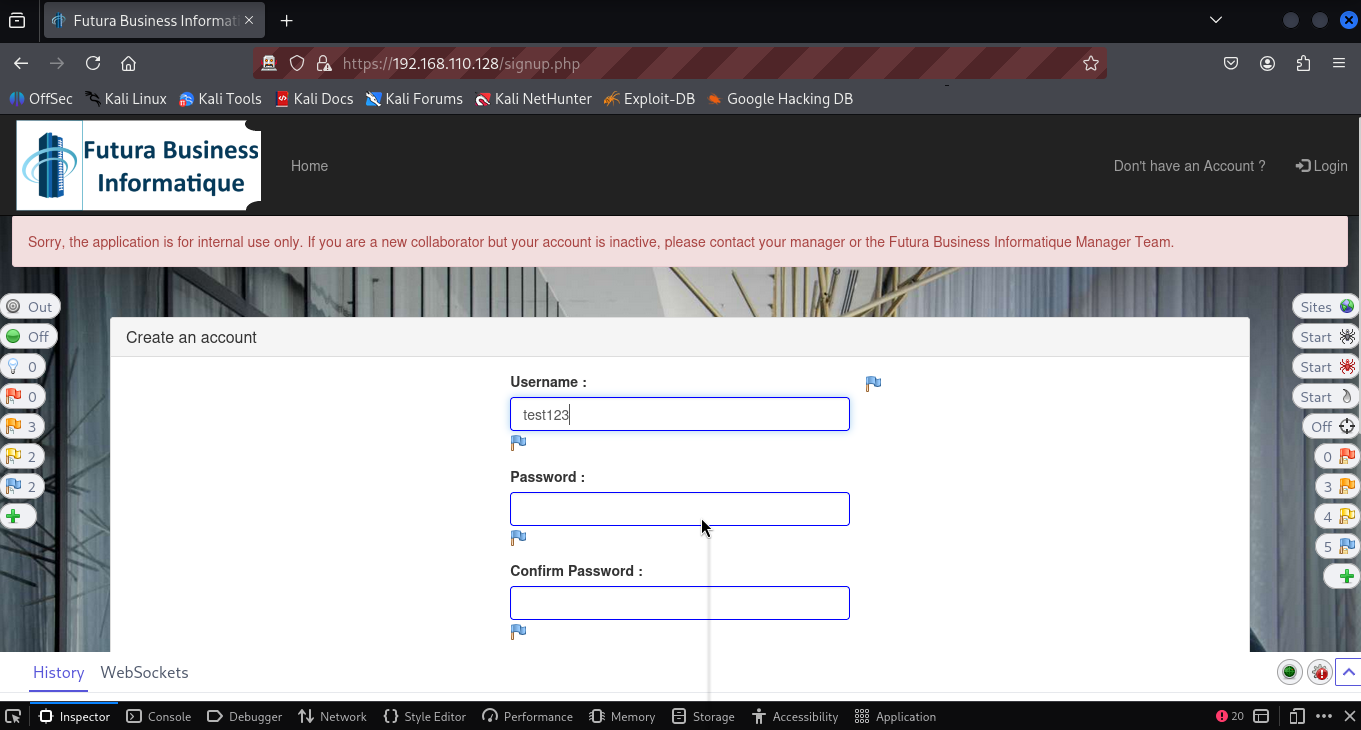


The administrative interface also outlines the organization’s hierarchy, revealing one Administrator account (rmasson), three Manager accounts (mnguyen, mriviere, rlefrancois), two Financial Approver accounts (afoulon, pbaudouin), and a number of Collaborator accounts. This presents a potential risk to the organization, as the information can be used for social engineering and credential stuffing attacks.

There is also the capability to manipulate user accounts via a GET request targeting the id and status of an account. Testing shows this interface provides some control, as accounts do return a 401 unauthorized HTTP status. Nevertheless, the mere exposure of these controls is an unacceptable and significant lapse in security.

**High Risk: SQL Injection Vulnerabilities**

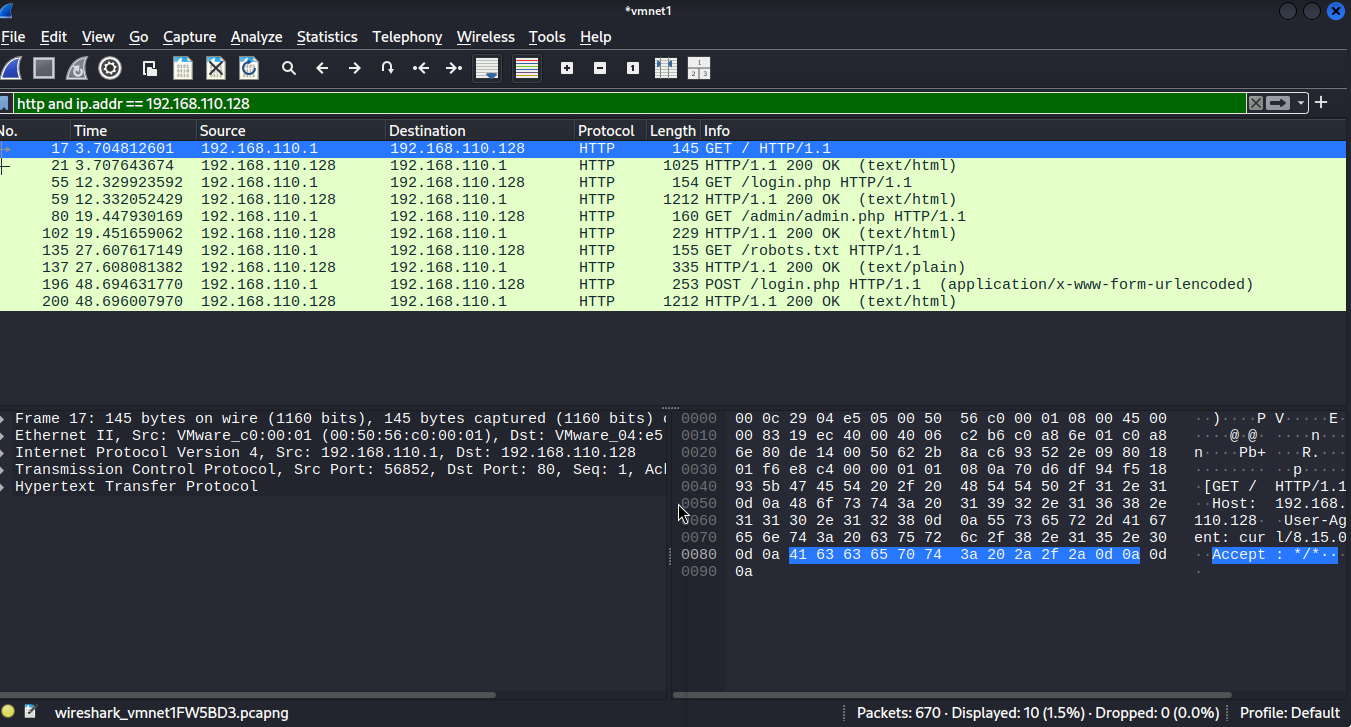
Automated scans and manual inspections uncovered a number of SQL injection vulnerabilities. Nessus scan results suggested blind SQL injection vulnerabilities on signup forms for email, username, lastname and firstname parameters. Through the "technical error" responses from the application when SQL injection error messages are sent, it indicates successful query manipulations.

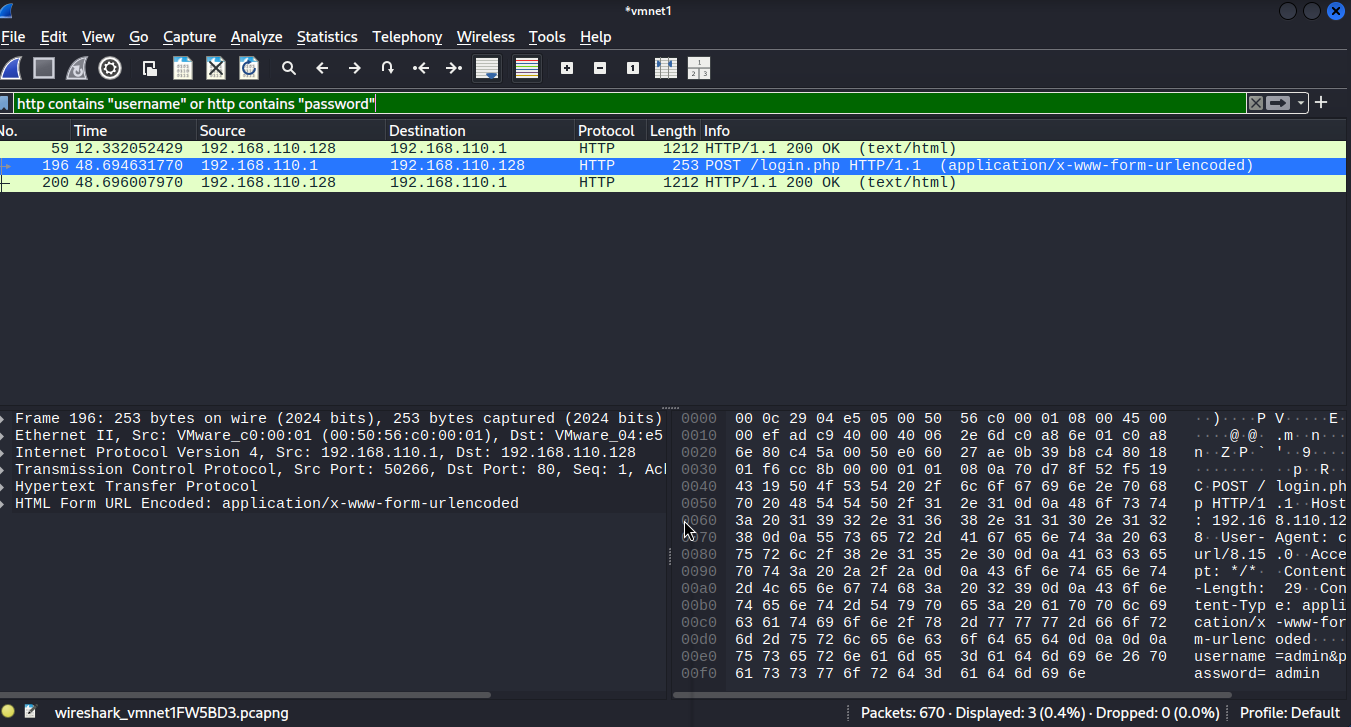


SQL injection attempts on the authentication mechanism using admin'-- and ' OR 1=1-- were designed to trigger application errors, confirming the presence of SQL injection vulnerabilities. These vulnerabilities may allow an adversary to bypass authentication, extraction of backend database sensitive information, and execution of arbitrary database commands.

**High Risk: Cleartext Credential Transmission**

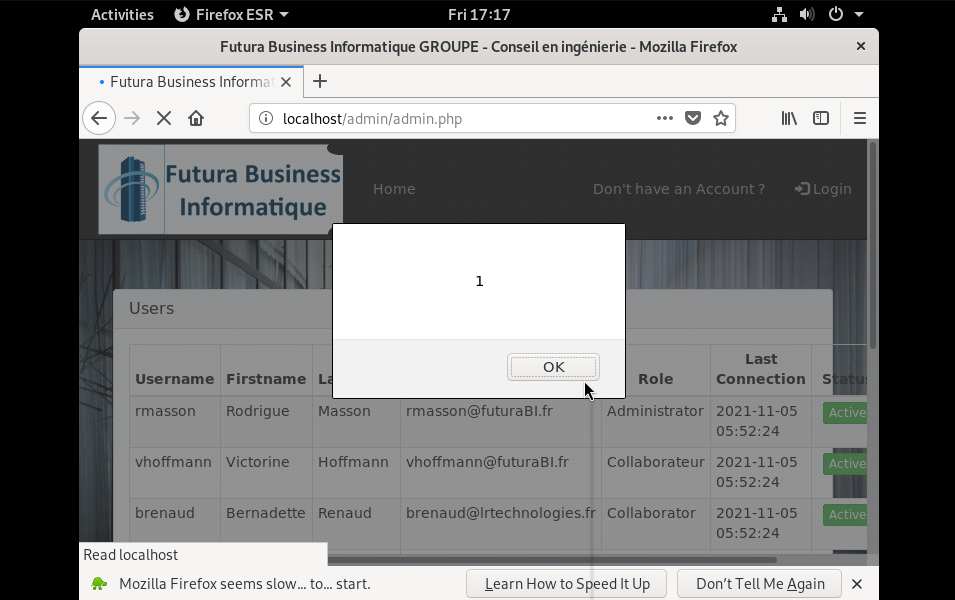
Analysis of the network traffic shows that all credentials used in authentication are sent over HTTP with no encryption and in cleartext. Plaintext and unencrypted usernames and passwords were captured in the traffic and successfully intercepted with Wireshark during retrieval of an authentication sessions. This fundamental security failure is irresponsible. User credentials are exposed to anyone with the right means to monitor network traffic.





**Medium Risk: Cross-Site Scripting (XSS)**

Stored Cross-Site Scripting (XSS) vulnerabilities were identified within the user database. Specifically, 16 user account IDs demonstrating that the signup form allows for the storage of malicious JavaScript, as both the firstname and lastname fields contain script tags and lacked proper input validation as well as output encoding. This stored XSS payload is then rendered in the administrative interface, which poses a risk to admin users.



**Medium Risk: Session Management Weaknesses**

PHP session with PHPSESSID cookies completely lack necessary security traits which is also problematic. First, the cookies do not have the HttpOnly flag, which means they can be stolen through XSS as they are exposed to client-side JavaScript. Moreso, the cookies do not have the Secure flag. This is a minor risk since the application does not have HTTPS functionality, but still a considerable risk to have no secure flag at all.

**Medium Risk: Missing Security Headers**

The application lacks critical security headers. It does not have a Content-Security-Policy, X-Frame-Options, or X-Content-Type-Options headers. The lack of anti-clickjacking headers poses a risk of UI redress attacks, the Content-Security-Policy header is critical in blocking any code injection attacks and the lack of it means the application has no protection against such attacks.

## Risk Assessment

Regarding the MyExpense application, the organization faces an extreme risk. The organization faces severe risk. Unauthenticated administrative access and SQL injection vulnerabilities drastically raise the risk exposure for hostile takeover of the system. An attacker would have access to sensitive employee data, user accounts, and system privileges.

There would be arguably no way for an attacker to be stopped as there is no transport layer encryption to secure the data. Nevertheless, if an authentication bypass was to occur, there would be no way to track the activities of the attacker. Moreover, passwords and other sensitive data are transmitted in cleartext, violating a core security principle and exposing the organization to severe legal risk.

Cleartext transmitted passwords will also violate legal requirements related to personal data.

Stored cross site scripting in the administrative interface opens further administrative users to great risk. An attacker can compromise an administrative account and escalate privileges to an account at will without much effort (NIST, 2008).

## Evidence and Proof of Concept

**Administrative Interface Exposure**

Proof of concept consists of being able to access the administrative interface directly. The administrative interface can be accessed without being authenticated and can be obtained via simple HTTP GET requests to /admin/admin.php. The interface will fully write out all user information for each of the sixteen system accounts. All personal and role data for each account will be fully exposed.

**SQL Injection Confirmation**

SQL injections were confirmed by testing the login form sequentially using various SQL injection payloads. After performing a set of standard authentication attempts with invalid credentials, I received messages saying, “Incorrect username or password.” On the other hand, responses to the SQL injection attempts returned messages saying, “Sorry, a technical error has occurred,” which indicates that the system was throwing SQL exceptions, confirming the vulnerability.

**Traffic Interception Evidence**

I was able to analyze the network traffic that I had captured, and for login attempts, it contained the full HTTP request made using the POST method, where the username and password values were displayed in plaintext. The captured traffic showed the authentication attempts where the username was “test123,” and the password was “password." This was done plainly, with absolutely no encryption or obfuscation.

**Cross-Site Scripting Evidence**

I verified the presence of stored XSS through the admin interface, which showed user accounts that had unescaped script tags. In account ID 16, I was able to confirm that storage and rendering of malicious JavaScript was successful, and it showed <script>alert(1)</script> in the firstname and lastname fields.

## Recommendations

**Immediate Actions Required**

Implementation of proper authentication and authorization controls for the administrative interface represents the highest priority remediation task. The administrative interface must have a level of security that requires strong authentication and also implement role-based access to ensure that sensitive user data is only accessed by authorized personnel (Stuttard & Pinto, 2011).

These must be addressed immediately using parameterized queries or prepared statements. All user input fields must have some form of input validation, particularly on login and signup forms, where we know there are exploitable vulnerabilities.

**Critical Security Enhancements**

To comply with suggested standards, all communications between clients and the server must be encrypted. Make the proper SSL and TLS configurations, obtain SSL certificates, and configure the web server to redirect HTTP traffic to HTTPS.

Furthermore, make sure to incorporate the best practices in validation and sanitization to address potential SQL and XSS vulnerabilities. All user input must be checked and, after processing, encoded correctly to minimize the risk of cross-scripting when rendering.

**Additional Security Measures**

To mitigate the risk of session fixation, implement a session timeout feature, reset the session on login, and apply session management best practices by including the HttpOnly and Secure cookie attributes (SANS, 2023).

Finally, all suggested security headers must be integrated to minimize the risk of attack. These include sets of Content-Security-Policy, X-Frame-Options, X-Content-Type-Options, and Strict-Transport-Security headers.

**Monitoring and Detection**

In-Depth logging and monitoring systems must be implemented and must assist in forensic analysis. Logging various interactions with the system (authentications, possible attacks, and admin logging activities) must be detailed to enable effective analysis of possible system threats.

Automated system scanning, combined with manual penetration testing, must be routinely conducted to identify security threats and newly discovered vulnerabilities. This will ensure that security vulnerabilities and risks are comprehensively and thoroughly dealt with.

## Conclusion

Data from the MyExpense application clearly demonstrates that fundamental security gaps exist. There are numerous and significant risks, which include; unauthenticated admin access, possible SQL injection with cleartext password and other unencrypted credential access. Remedial actions must be taken with great urgency.

By implementing basic security principles of authentication, transport layer encryption, input validation, and other modern security innovations, we will be able to significantly eliminate the greatest risks presented within the system.

The gaps in system security are indicative of more systemic issues and fundamental gaps in comprehensive application security. This calls for the closing of application security gaps in development, deployment, and maintenance of security application lifecycles.

Organizations using this application will need to limit access until essential vulnerabilities are addressed, set additional security controls at the network layers, and detect network exploitation attempts while these vulnerabilities are being addressed.

# References

**Common Vulnerability Scoring System (CVSS) v3.1. (2019).** CVSS v3.1 Specification Document**. Forum of Incident Response and Security Teams (FIRST). Retrieved from** <https://www.first.org/cvss/v3.1/specification-document>

**NIST Special Publication 800-115. (2008).** Technical Guide to Information Security Testing and Assessment**. National Institute of Standards and Technology. Retrieved from** <https://csrc.nist.gov/publications/detail/sp/800-115/final>

**OWASP Foundation.** (2021). OWASP Top Ten 2021: The Ten Most Critical Web Application Security Risks. Retrieved from <https://owasp.org/Top10/>

**SANS Institute.** (2023). Web Application Penetration Testing Methodology. SANS Reading Room. Retrieved from <https://www.sans.org/white-papers/>

**Stuttard, D., & Pinto, M.** (2011). The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws (2nd ed.). Wiley Publishing.

Engebretson, P. (2013). *The Basics of Hacking and Penetration Testing: Ethical Hacking and Penetration Testing Made Easy*. Reference from <https://www.amazon.com/Basics-Hacking-Penetration-Testing-Ethical/dp/0124116442>

Stuttard, D., & Pinto, M. (2011). *The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws* (2nd ed.). Reference from <https://www.amazon.com/Web-Application-Hackers-Handbook-Exploiting/dp/1118026470>