



Software And Database Vulnerability Report

Submitted by:
Ishwariya MANI

Contents

1) Passwords_Hardcoded_in_executable	4
2) password_shown_whilest_typed	7
3) passwords_in_plaintext_in_config_file.....	9
4) passwords_stored_in_plaintext_in_database	11
5) database_user_over_privilege.....	15
6) network traffic not encrypted	17
7) SQL injection	21
8) password_aging_not_implemented	25

SYNTHESIS:

An Audit was conducted on the stock application from May 25th to June 10th, 2023, with the aim of assessing the app's security and compliance. The audit entailed a thorough evaluation of the stock app's security measures and practices, including password security, database security, and privileges and access control. The primary objective was to identify vulnerability and ensure that the application adhered to established security standard.

The high-risk vulnerabilities within stock app that pose significant threats to the security of the system. These vulnerabilities grant attackers the ability to gain unauthorized access and visibility into the system's operations and applications workflow. They can exploit these vulnerabilities to view and potentially alter sensitive information, including passwords. There are also Medium risk vulnerabilities, they may not have as severe an impact as the high-risk ones, but they still provide attackers with the ability to gain insight into the workflow of the application. This can be leveraged to identify potential weaknesses or exploit other vulnerabilities. Furthermore, there are low-risk vulnerabilities that need to be addressed to prevent unauthorized access, data breaches, and potential disruptions to the application's functionality.

Recommendations to fix these vulnerabilities in the stock app, firstly strengthen password security by enforcing strong password policies that include a combination of upper, and lowercase letters, numbers, and special characters. Additionally implementing MFA (Multi-Factor Authentications) like keeping OTP is better for security. Database security should be prioritized by regularly applying patches and updates, implementing proper access controls and privilege management, and encrypting sensitive data. It is good to conduct regular security assessments and vulnerability scans on the database to identify and address any weaknesses.

Vulnerability Sheet:

1) Passwords_Hardcoded_in_executable

1.1 CWE-259: Use of Hard-coded Password

<https://cwe.mitre.org/data/definitions/259.html>

1.2 CVSS 3.1 Base Score Metrics:

- <https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:L/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N&version=3.1>

- **Score:**



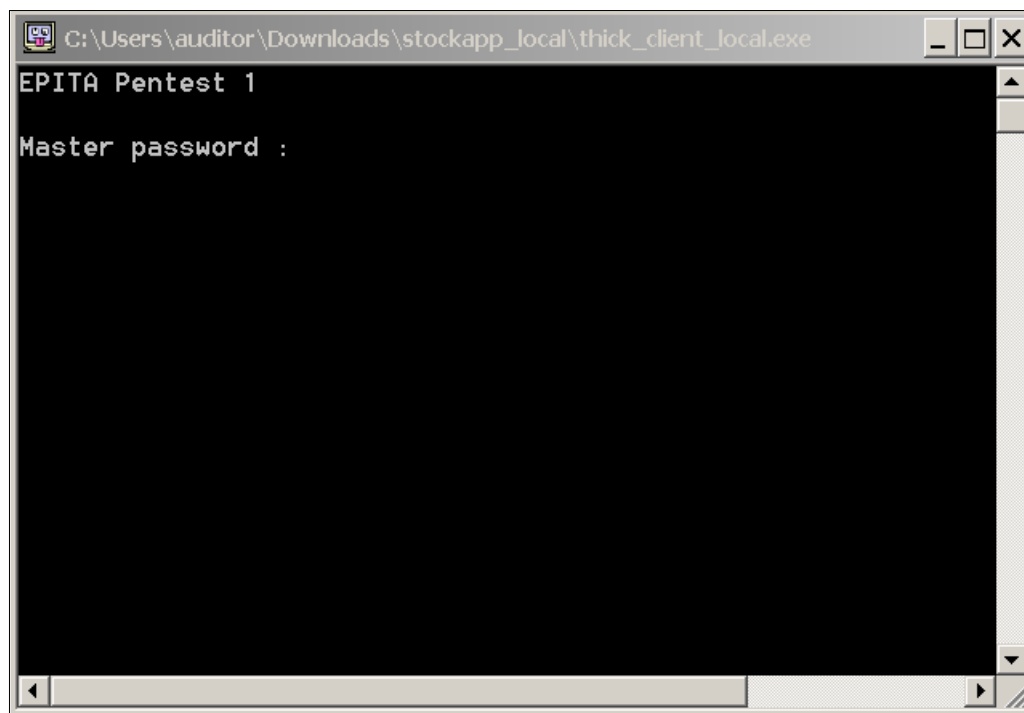
- **Risk Level:** High

1.3 Description:

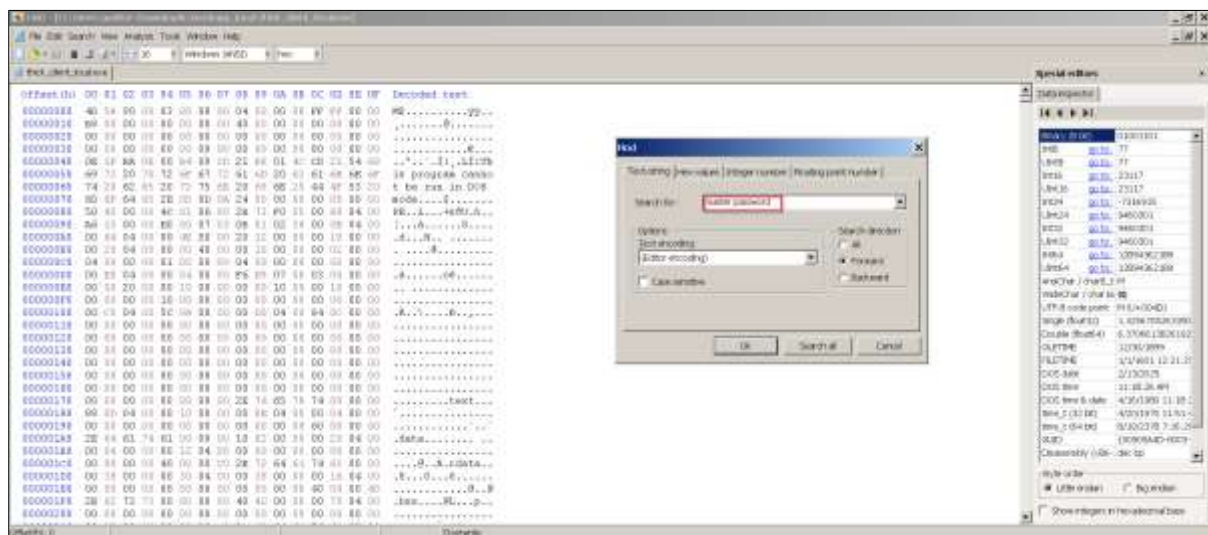
The password Hardcoded in executable vulnerability is storing of passwords directly in the executable code of a thick client application, it allows hackers to misuse this passwords, leading to unauthorized access, data breaches of customer accounts. This impacts the compromised customer trust and financial losses.

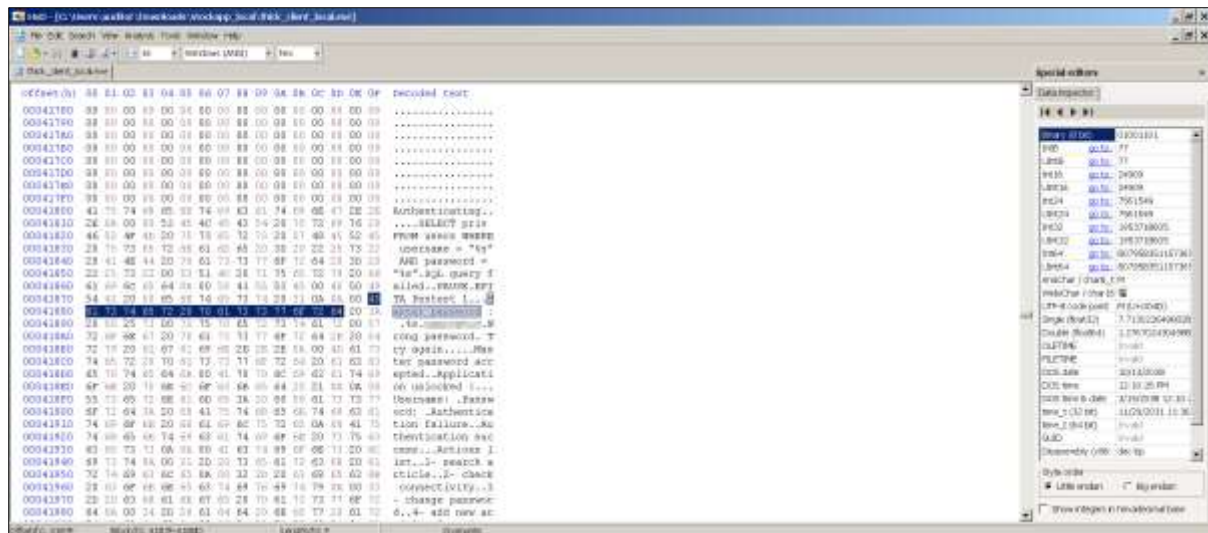
1.4 Exploitation:

- a) When the thick client local.exe application opened, it prompted for a password.



- b) To identify the password, one can utilize the HxD application by dragging the thick client local.exe. By performing a search Master Password within the file, the password can be seen.





1.5 Recommendation:

- Avoid storing passwords directly within the executable code
- Implementing the secure methods for storing and retrieving passwords, such as using strong encryption algorithms.

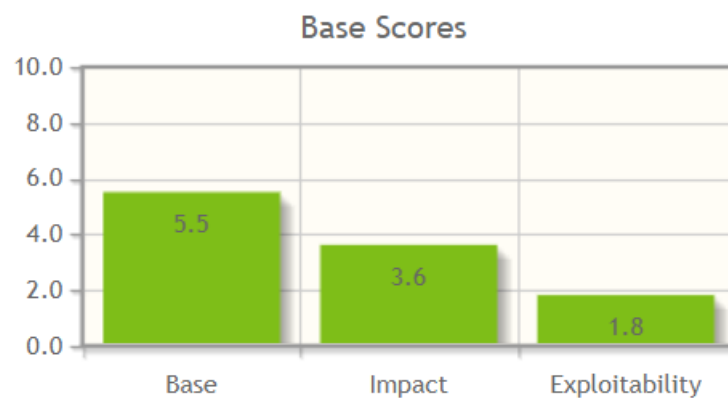
2) password_shown_whilest_typed

2.1 CWE-549: Missing Password Field Masking

<https://cwe.mitre.org/data/definitions/549.html>

2.2 CVSS 3.1 Base Score Metrics:

- <https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:L/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N&version=3.1>
- **Score:**



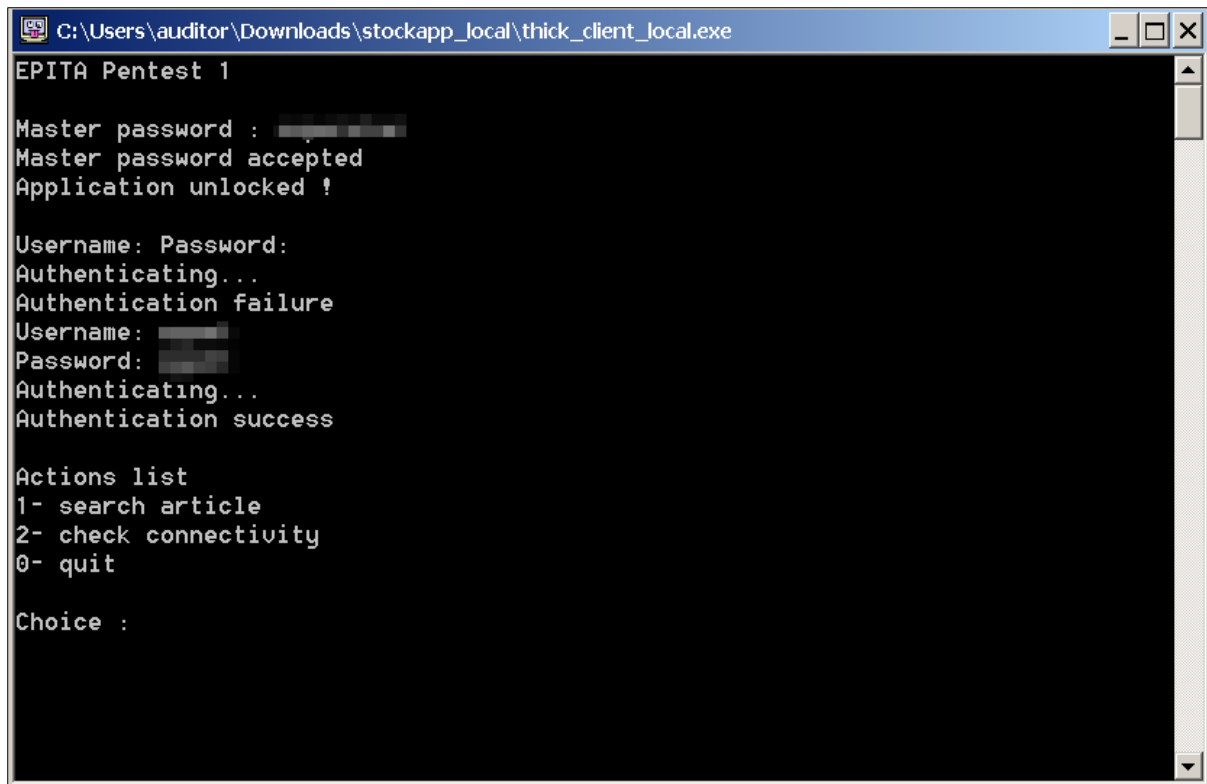
- **Risk Level: Medium**

2.3 Description:

The Product does not mask passwords during entry, increasing the potential for attackers to observe and capture passwords.

2.4 Exploitation:

- When opening the thick client local executable, it prompts the user to enter a password. However, due to a lack of password masking, attackers can observe the password as it is being typed, compromised its visibility and security.



```
C:\Users\auditor\Downloads\stockapp_local\thick_client_local.exe
EPITA Pentest 1

Master password : 
Master password accepted
Application unlocked !

Username: Password:
Authenticating...
Authentication failure
Username: 
Password: 
Authenticating...
Authentication success

Actions list
1- search article
2- check connectivity
0- quit

Choice :
```

2.5 Recommendations:

- a) Modify the thick client application to implement password masking functionality. This ensure that passwords are displayed as asterisks or dots while being typed, preventing visual observation by attackers.
- b) Use complex passwords and following strong password policies can help to prevent it.

3) passwords_in_plaintext_in_config_file

3.1 CWE-312: Cleartext Storage of Sensitive Information

<https://cwe.mitre.org/data/definitions/312.html>

3.2 CVSS 3.1 Base Score Metrics:

- <https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:L/AC:L/PR:N/UI:N/S:C/C:H/I:L/A:N/E:X/RL:X/RC:C&version=3.1>
- **Score**



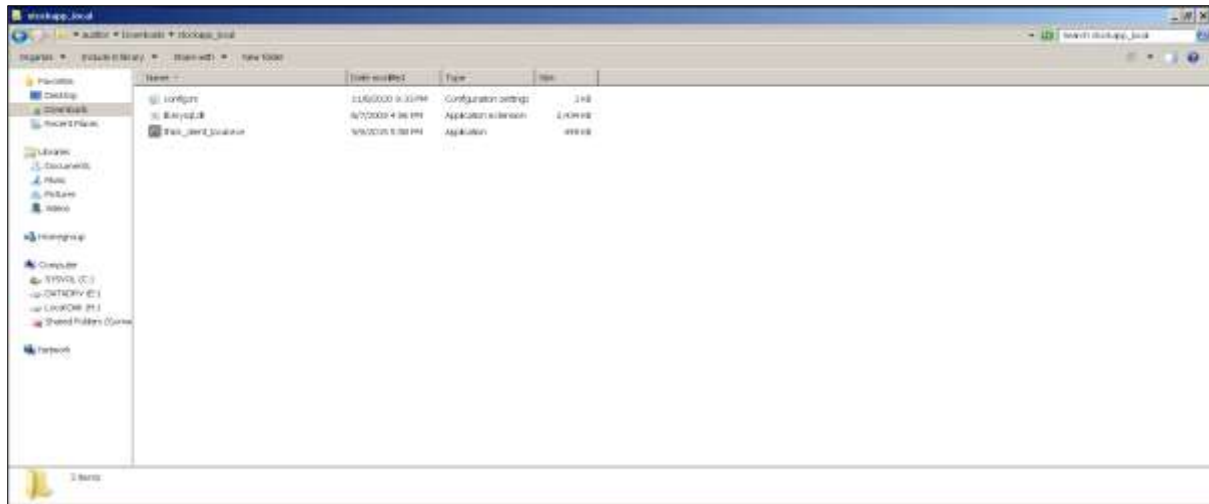
- **Risk Level : High**

3.3 Description:

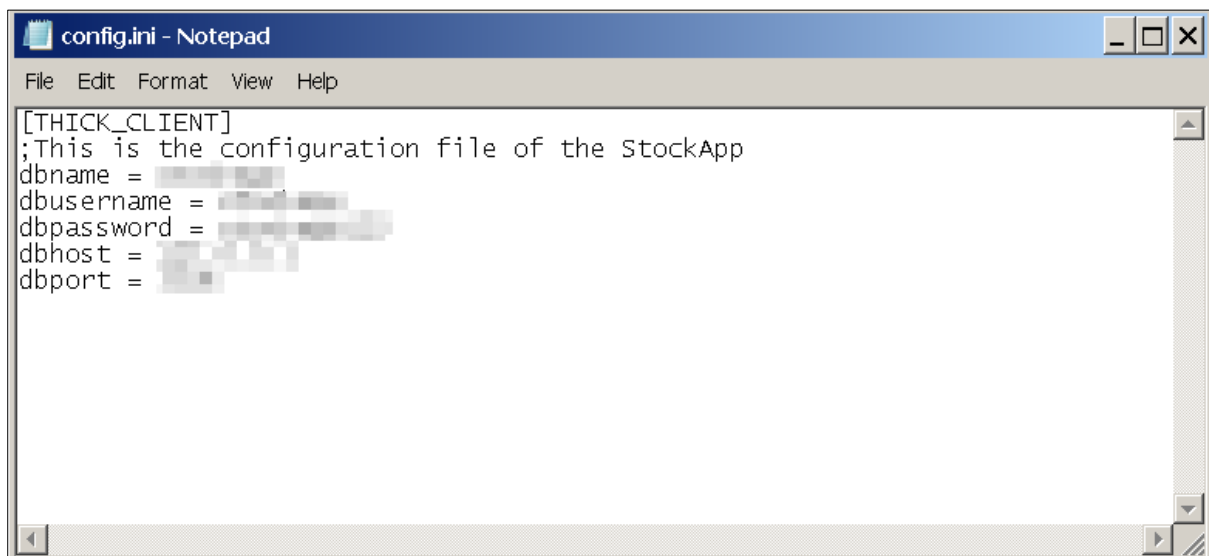
Storing the credentials /sensitive information in the configuration file is a serious vulnerability that allows anyone can access the file to view the sensitive information. It impacts the unauthorized access and data breaches. The attackers can exploit the unauthorized access , escalate the privileges , perform credentials stuffing or can do brute force attacks.

3.4 Exploitation:

- a) Once the stock app has been downloaded and extracted, it is possible to view the configuration file.



- b) Opening the config.ini file one can see the database credentials, including the username, password, host, and port details in plain text.



3.5 Recommendation :

- Implementing the encryption mechanism to protect sensitive configuration data. Encrypting the data ensures that even if an attacker gains access to the configuration file, the information will remain unreadable and unusable.
- Implement validation mechanism to ensure that the configuration file does not contain any plain text sensitive data before deployment. Automated checks and code analysis tools can help identify and prevent the inclusion of sensitive information in configuration files.
- Employ the configuration management tools that provide secure storage options for the sensitive data. These tools often feature such as encryption, access control and auditing to ensure the confidentiality of configuration information.
- Reference: <https://www.kamilgrzybek.com/blog/posts/how-to-store-sensitive-configuration-data>

4) passwords_stored_in_plaintext_in_database

4.1 CWE-256: Plaintext Storage of a Password

<https://cwe.mitre.org/data/definitions/256.html>

4.2 CVSS 3.1 Base Score Metrics :

- <https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:N/AC:H/PR:H/UI:N/S:C/C:H/I:H/A:H&version=3.1>

- **Score :**



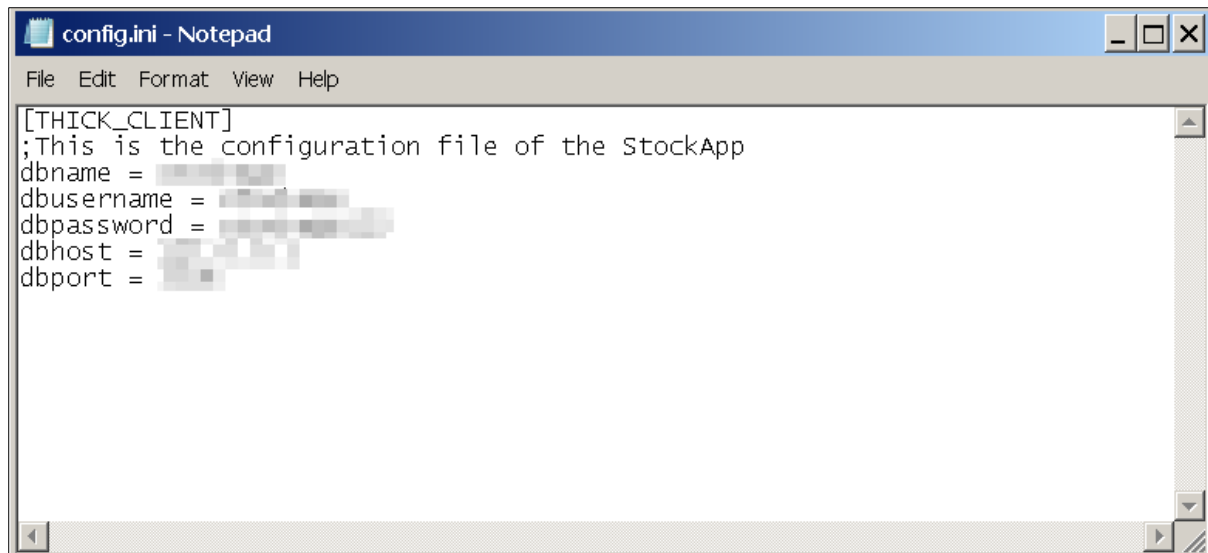
- **Risk Level : High**

4.3 Description:

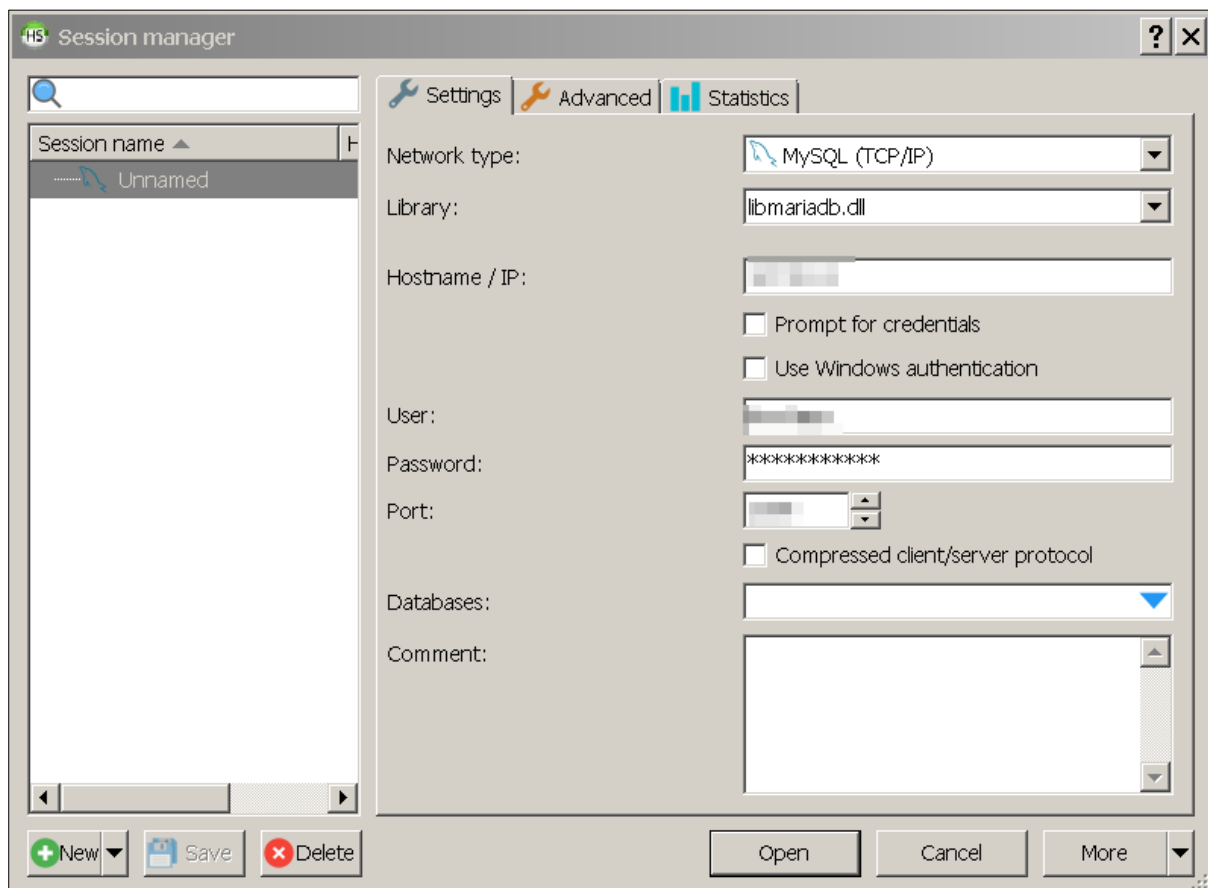
The Vulnerability of storing the password in the plaintext inside the database without encrypting can leads attackers to gain the unauthorized access and can misuse the password by modifying it.

4.4 Exploitation:

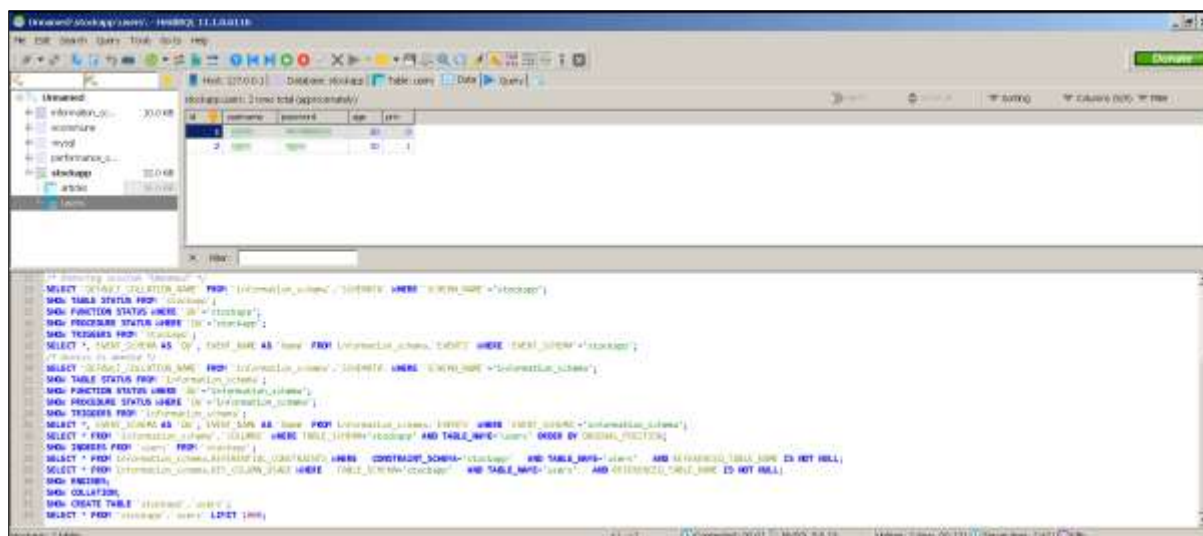
- a) After obtaining the plain text database credentials from the config.ini file, an attacker can use this information to access the database.



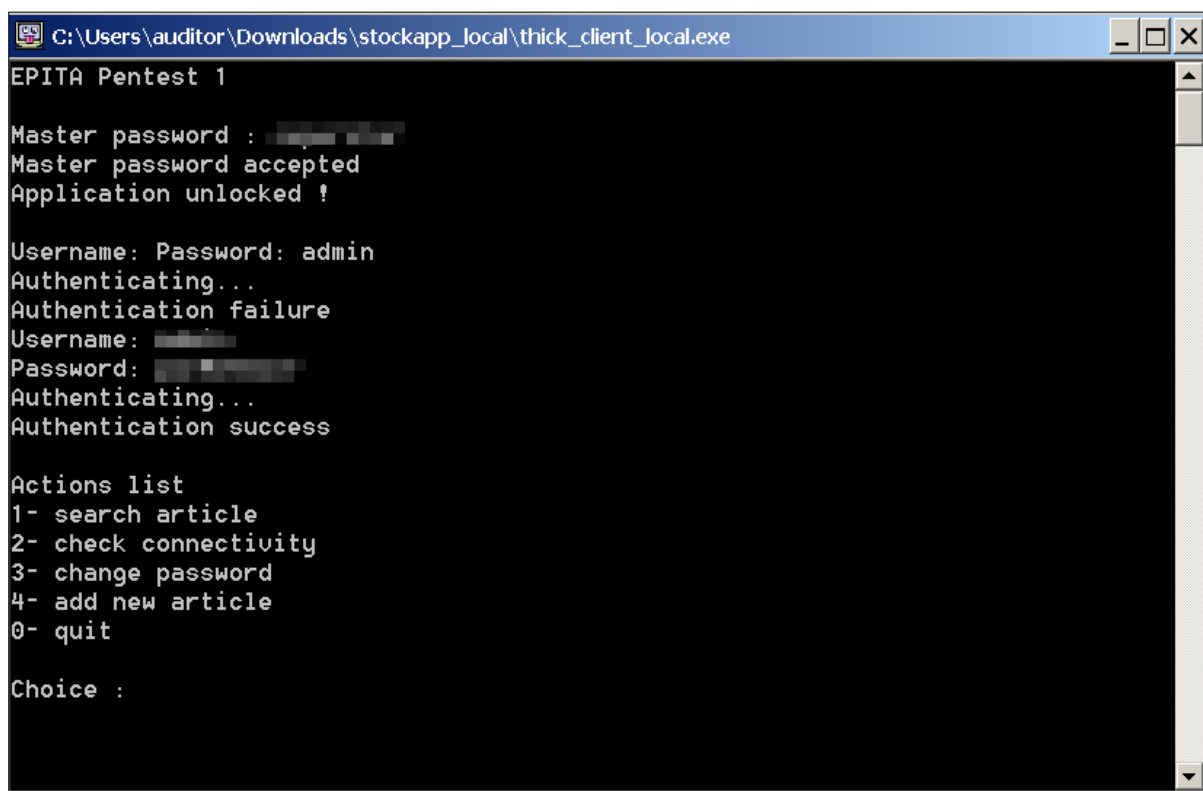
b) To access the database the attackers can use tools like Heidi SQL application.



c) Once login the database with the provided information from the config.ini file, by navigating the stock app section under users the hackers can able to see the users information like username, password.



d) Using the username and password, can be able to login into the thick client local executable.



4.5 Recommendations :

- Instead of storing the plain text credentials in the config.ini file, employ encryption techniques to protect sensitive information.
- Adjust the file permissions for the config.ini file to ensure that only authorized users or processes have read access.
- Avoid using high-privileged accounts or granting unnecessary permissions to the application's database users.

- d) Ensure that communication between database and application layer is secured using encryptions. Such as SSL/TLS. This protects the credentials and data in the transit.
- e) Implementing the layered architecture, separate the application into distinct layers. Ensure that the sensitive information's like database credentials is not directly accessible or expose in the presentation layer.

5) database_user_over_privilege

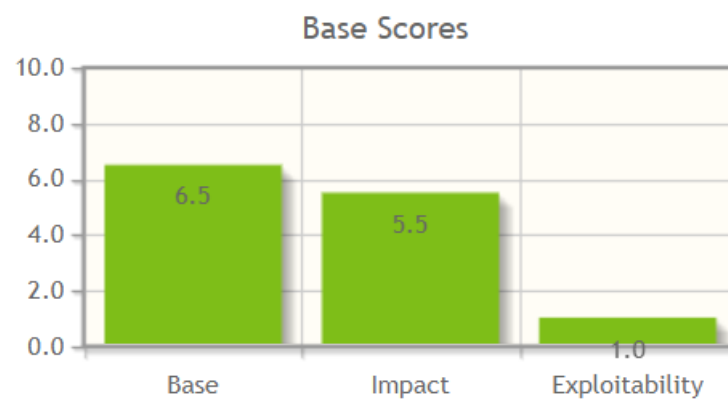
5.1 CWE-274: Improper Handling of Insufficient Privileges

<https://cwe.mitre.org/data/definitions/274.html>

5.2 CVSS 3.1 Base Score Metrics:

- <https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:N/AC:H/PR:H/UI:N/S:C/C:H/I:H/A:H&version=3.1>

- **Score:**



- **Risk Level:** High

5.3 Description:

The database_user_over_privileged vulnerability in the stockapp is identified through config.ini file, allows attackers to exploit an over-privileged database user. This can result in unauthorized access , data manipulation and business disruption.

5.4 Exploitation :

- a) The information's obtained from the config file and accessing the database go to tools > user management to see the privilege access.

6) network traffic not encrypted

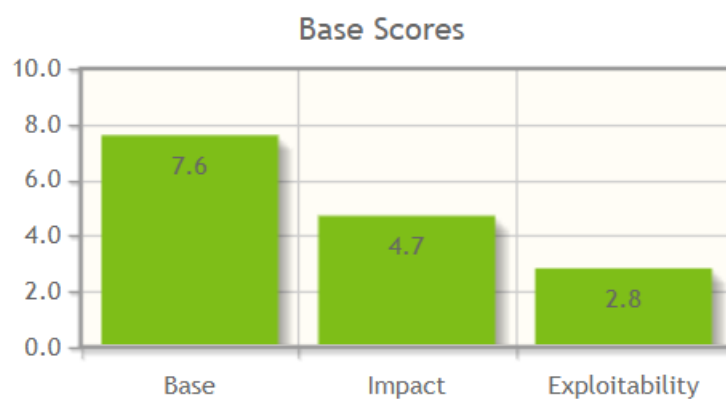
6.1 CWE-319: Cleartext Transmission of Sensitive Information

<https://cwe.mitre.org/data/definitions/319.html>

6.2 CVSS 3.1 Base Score Metrics :

- <https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:L/A:L/E:X/RL:X/RC:C&version=3.1>

- **Score :**



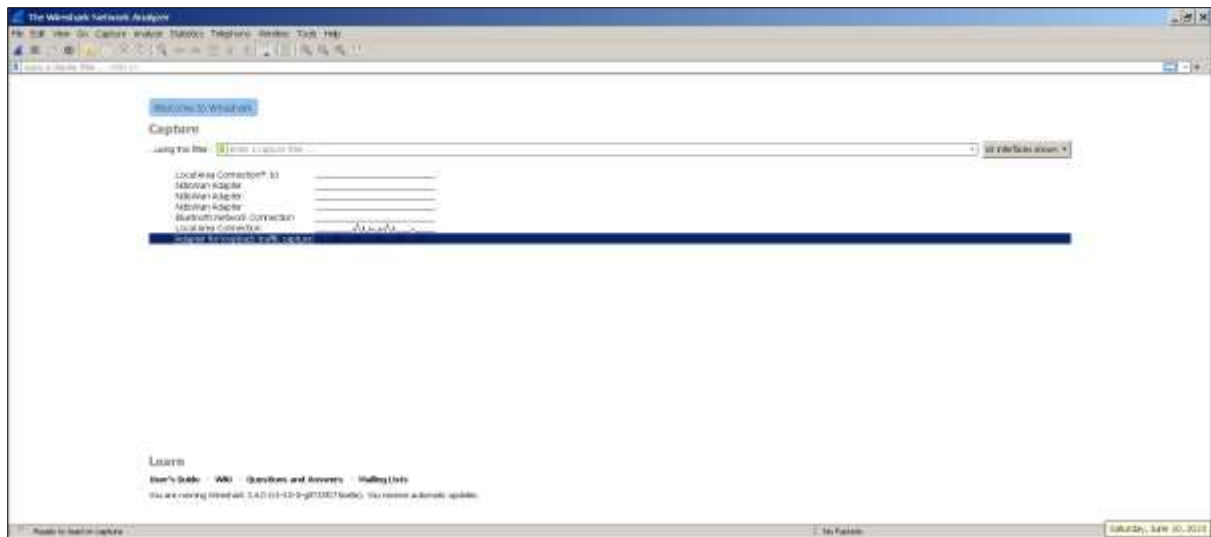
- **Risk Level : High**

6.3 Description:

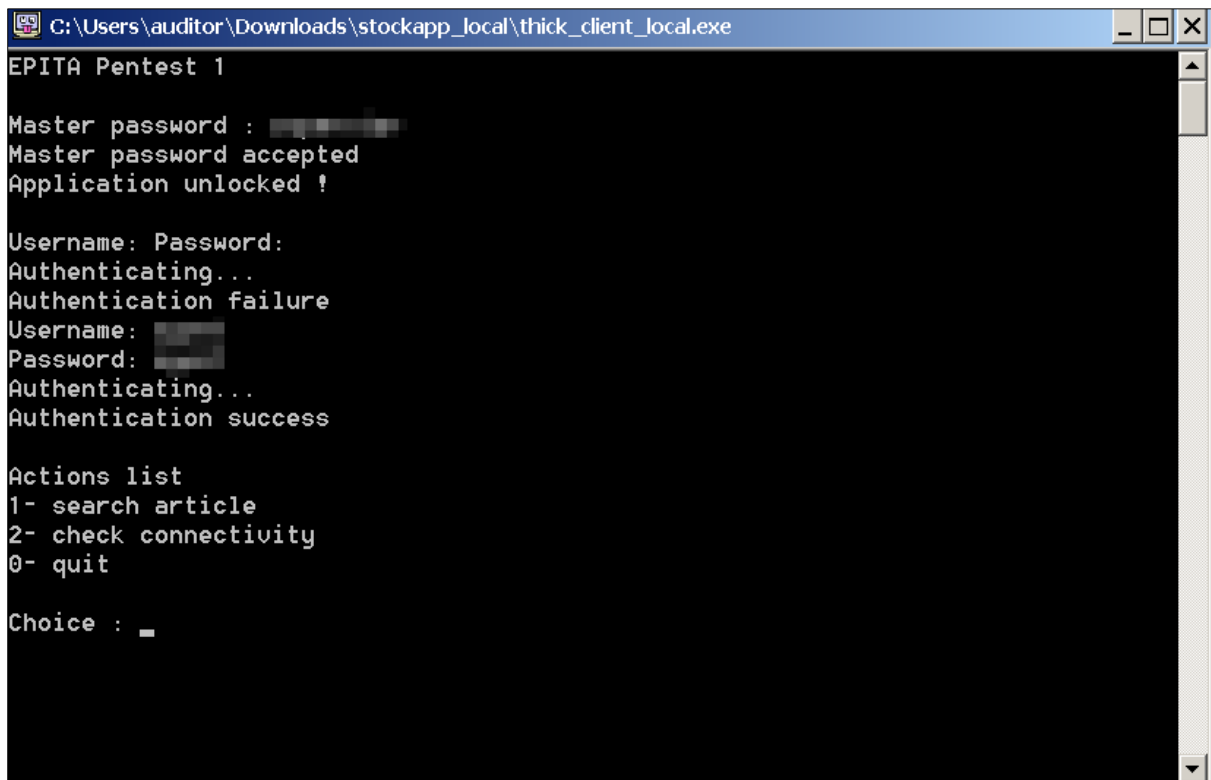
The Lack of network traffic encryption in the thick client executable allows attackers to capture and view plain-text passwords using tool like Wireshark. This puts users credentials at risk, leading to authorized access.

6.4 Exploitation:

- a) Open the wireshark application and click the option “Adapter for loopback traffic capture” to start capturing the communication between client and server.

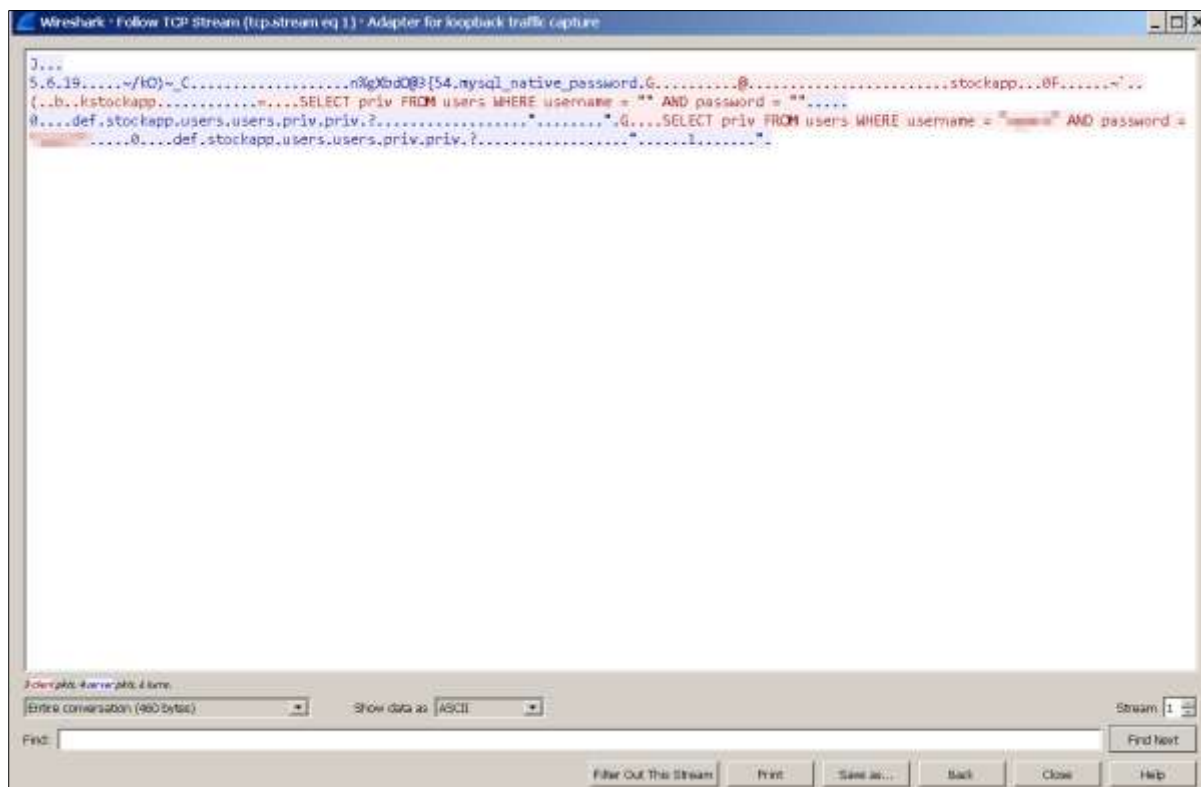


b) Then start the thick client local exe and it asks for the username and password for authentication.



c) When credentials are entered in the client(Thick client local exe) the communication between the client and server is captured and visible within Wireshark application.

- d) By selecting the request Query > follow > TCP stream, in the wireshark tool, the credentials entered in the client can be viewed in plain text.



6.5 Recommendations :

- Implement strong encryption protocols and algorithm to secure the communication channel. Use SSL/TLS (secure sockets layer/transport Layer security) to encrypt data transmission and ensure the confidentiality and integrity of the transmitted data.
- Implement proper Network segmentation to separate sensitive data traffic from other non-sensitive traffic. This helps restrict the exposure of sensitive data to unauthorized actors gain access to the storage system.
- Reference: https://portswigger.net/kb/issues/01000200_unencrypted-communications

7) Sql injection

7.1 CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

<https://cwe.mitre.org/data/definitions/89.html>

7.2 CVSS 3.1 Base Score Metrics:

- <https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:N/AC:H/PR:H/UI:N/S:C/C:H/I:H/A:H&version=3.1>

- **Score :**



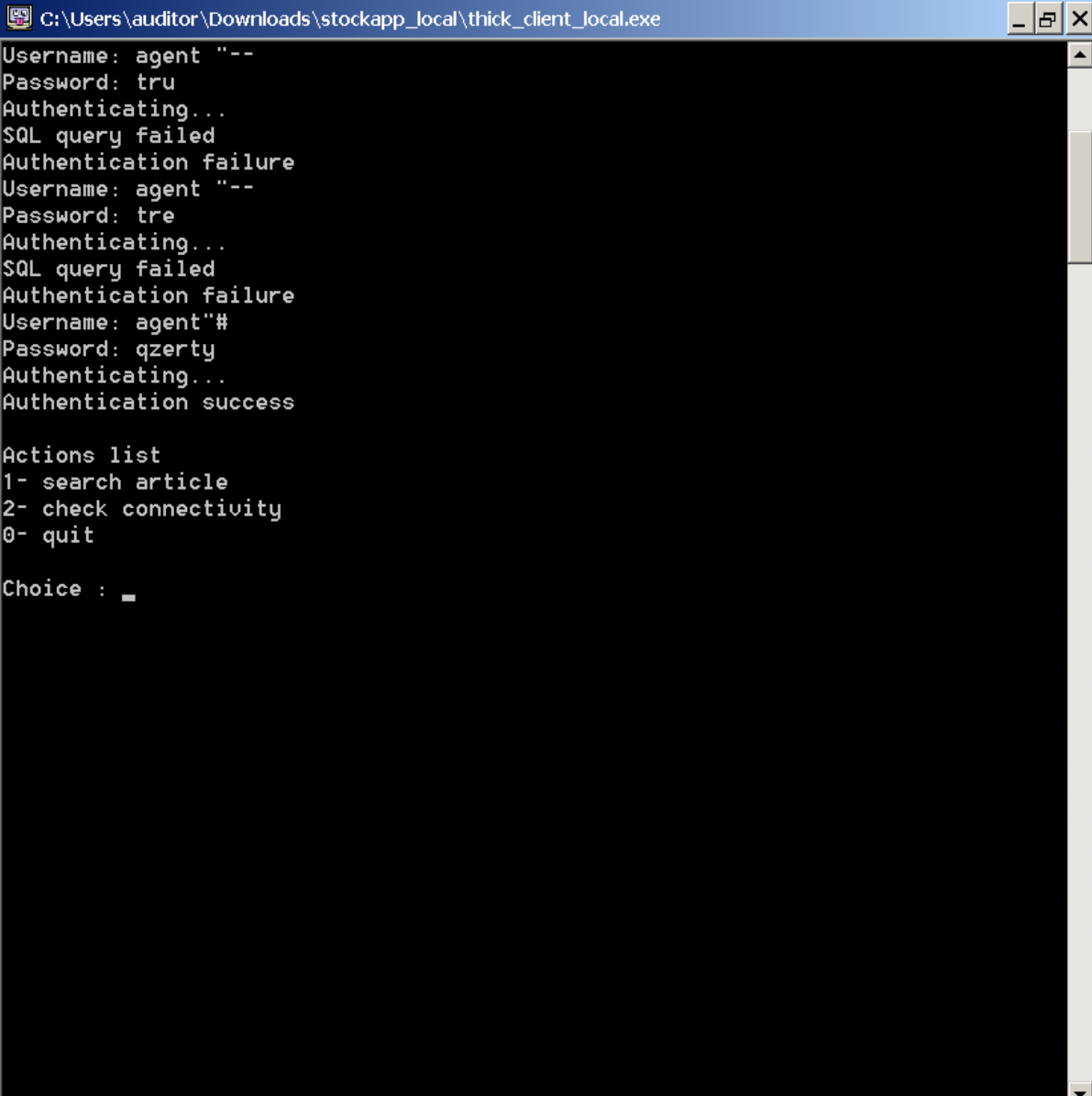
- **Risk Level : High**

7.3 Description:

The product constructs all or part of an SQL command using externally influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended SQL command when it is sent to a downstream component.

7.4 Exploitation:

- a) Open the Heidi SQL application, and connect the database using the information from the config.ini file and under stockapp column , can able to inject the SQL query.



```
C:\Users\auditor\Downloads\stockapp_local\thick_client_local.exe
Username: agent "--
Password: tru
Authenticating...
SQL query failed
Authentication failure
Username: agent "--
Password: tre
Authenticating...
SQL query failed
Authentication failure
Username: agent"#
Password: qzerty
Authenticating...
Authentication success

Actions list
1- search article
2- check connectivity
0- quit

Choice : _
```

7.5 Recommendations:

- a) Use parameterized SQL commands, or stored procedures, with parameters containing the untrusted input.
- b) Instead of constructing SQL commands dynamically, utilize prepared statements or parametrized queries. These methods separate the SQL logic from the data input, preventing malicious inputs from altering the structure of the SQL command.
- c) Limit the privileges and permissions granted to the database.
- d) Implement the Web application firewall as an additional layer of defence to monitor and filter incoming HTTP requests. WAF's can detect and blocks suspicious SQL injection attempts, providing an added layer of protection.
- e) Reference: <https://learn.microsoft.com/en-us/dotnet/fundamentals/code-analysis/quality-rules/ca3001>

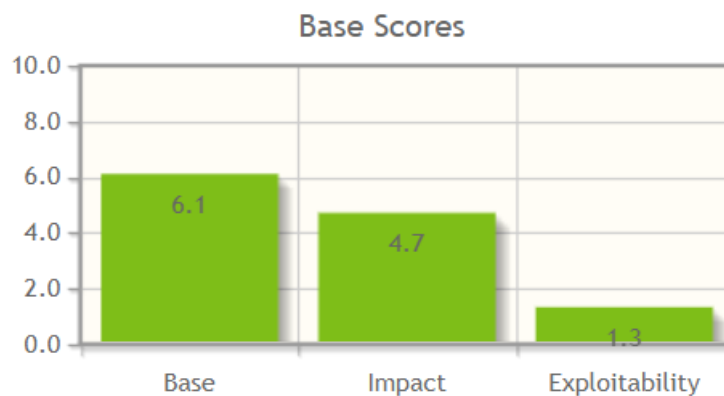
8) password_aging_not_implemented

8.1 CWE-262: Not Using Password Aging

<https://cwe.mitre.org/data/definitions/262.html>

8.2 CVSS 3.1 Base Score Metrics:

- <https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:L/AC:L/PR:L/UI:R/S:U/C:H/I:L/A:L&version=3.1>
- Score:



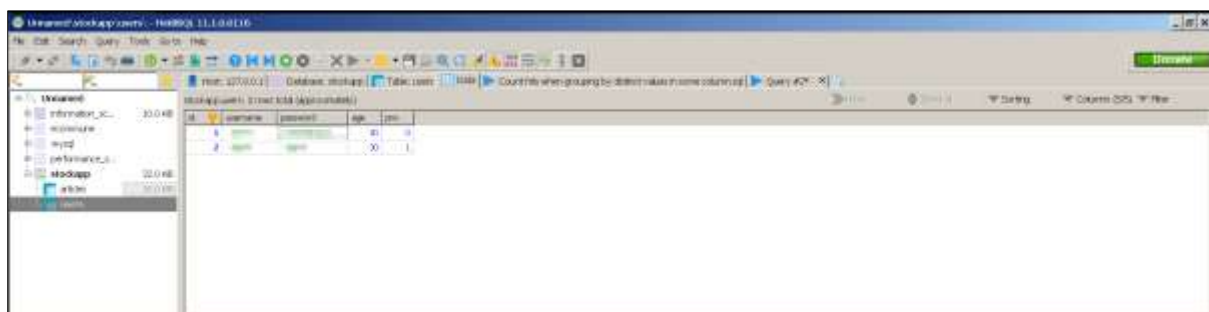
- Risk Level: High

8.3 Description:

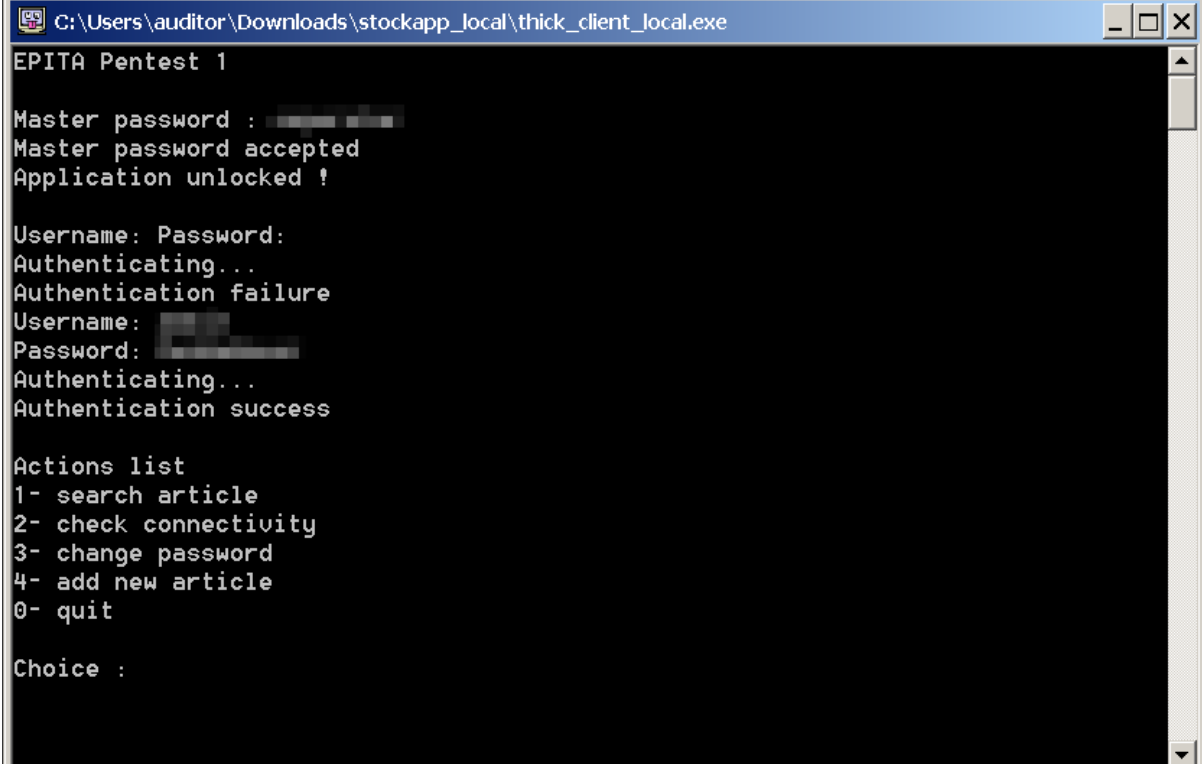
The thick client uses the config.ini file connected to the database and reads the few data from it. The client limits the amount of data that it will read to show you. The client will bypass the credentials and sprightly connect to the database. The passwords were already saved in our local computer and it will connect automatically in future.

8.4 Exploitation

- After connecting the database using the information's from the config.ini file it can connect to the Heidi SQL. And the user's information and password can be viewed.



- b) The password used to access the thick client application are stored locally on the computer, making it easier to log in to the application in the future. The application does not enforce an expiration rule for password.



```
C:\Users\auditor\Downloads\stockapp_local\thick_client_local.exe
EPITA Pentest 1

Master password : 
Master password accepted
Application unlocked !

Username: Password:
Authenticating...
Authentication failure
Username: 
Password: 
Authenticating...
Authentication success

Actions list
1- search article
2- check connectivity
3- change password
4- add new article
0- quit

Choice :
```

8.5 Recommendations:

- Implement 2FA as an additional layer of security to authenticate users. This can include using SMS codes, authenticator apps or biometrics verification.
- Set a policy for password expiration and enforce regular password rotations. This reduces the impact of compromised password by ensuring that they are regularly updated.
- Implementing the session time out mechanism to automatically logout the user after a period of inactivity.
- Reference: <https://learn.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/maximum-password-age>