KLEAP

CYBERSECURITY

Thick Client Application Security Report

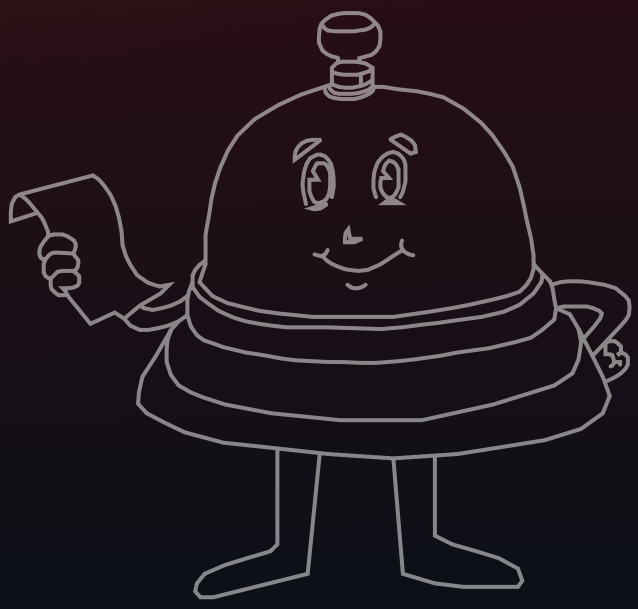


Table of Contents

Statement of Confidentiality

This pentest report contains confidential and proprietary information belonging to KLEAP Technologies Pvt. Ltd. and Client. It is intended solely for the use of the Client and KLEAP Technologies Pvt. Ltd. The information provided within this report should not be disclosed, distributed, or shared with any third parties without the explicit written consent of both KLEAP Technologies Pvt. Ltd. and Client. Any unauthorized use or disclosure of this information is strictly prohibited and may result in legal action.

Executive Summary

Client engaged KLEAP Technologies Pvt. Ltd. to perform penetration testing of the APIs. The primary goal of this API penetration testing project was to identify any potential areas of concern associated with the API in its current state and determine the extent to which the system may be breached by an attacker possessing a particular skill and motivation. The assessment was performed in accordance with the “best-in-class” practices as defined by ISECOM's Open Source Security Testing Methodology Manual (OSSTMM) and Open Web Application Security Project (OWASP).

KLEAP Technologies Pvt. Ltd. conducted the penetration testing during the period of March 20th, 2024 to April 11th, 2024. All testing activities were performed on the staging environment provided by the customer and completely isolated from the production data. While performing the testing activities, KLEAP Technologies Pvt. Ltd. emulated an external attacker without prior knowledge of the environment. To test the user-authenticated area and privilege escalation vulnerabilities, the customer supplied KLEAP Technologies Pvt. Ltd. credentials for several registered user and admin accounts.

Scoping and Time Limitations

Scoping during the engagement did not permit denial of service or social engineering across all testing components.

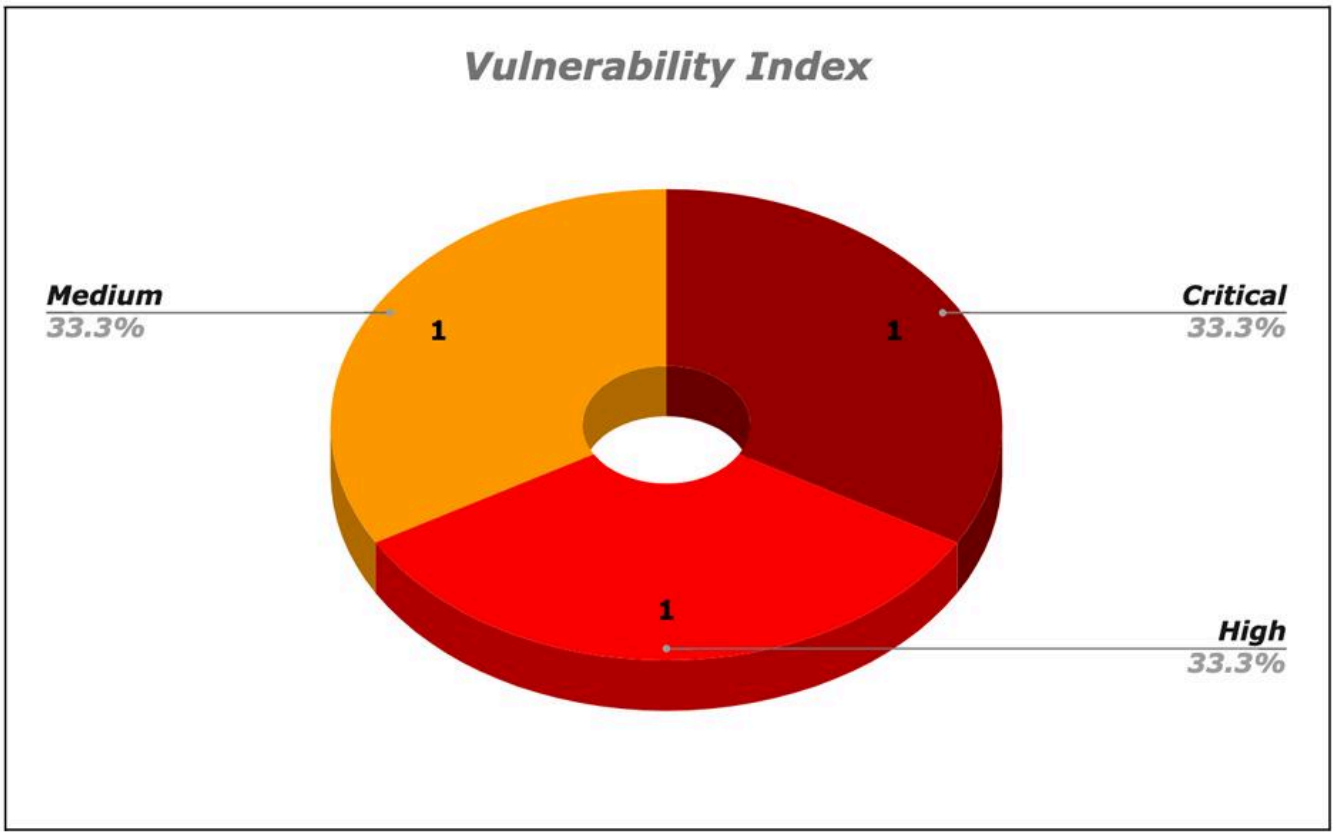
Time limitations were in place for testing. API penetration testing was permitted for seven (7) business days.

Testing Summary

(Overall Summary of the findings)

Scope	Critical	High	Medium	Low	Info	Total
Client AD	1	1	1	0	0	3

Table 1: Findings per asset



Vulnerability Index

Vulnerability Status

Sr. No.	Vulnerability	Severity	Status
1	SQL Injection	CRITICAL	OPEN
2	Side Channel Data Leaks	HIGH	OPEN
3	Dumping connection string from memory	MEDIUM	OPEN

Scope Summary

In-Scope Assets

The following assets were considered explicitly in-scope for testing:

Assets In-Scope	Hostname / CIDR / IP
Thick Client Application	Vulnerable Thick Client Application

Out-of-Scope Assets

(If any)

Assets Out-of-Scope	Hostname / CIDR / IP
N/A	N/A

Methodology

The pentest methodology employed by KLEAP Technologies Pvt. Ltd. follows a systematic approach to assess the security posture of client systems.

Our Penetration Testing Methodology is based on following guidelines and standards:

- Penetration Testing Execution Standard (PTES)
- Open Source Security Testing Methodology Manual (OSSTMM)
- OWASP Testing Guide
- OWASP Top 10 Application Security Risks

Engagement Phases

1. Reconnaissance

The objective of the reconnaissance phase is to gather information about the target thick client application through passive reconnaissance and OSINT techniques. This includes identifying the application's functionality, components, and dependencies. The goal is to gain a better understanding of the target's infrastructure, potential vulnerabilities, and attack surface. Actions include identifying the application's environment, such as operating systems, libraries, and frameworks used, and collecting publicly available information about the application and its developers.

2. Scanning and Enumeration

The objective of the scanning and enumeration phase is to conduct active scanning to identify potential entry points and gather detailed information about the application. This involves using static analysis tools to scan the application's binary files and source code (if available), performing dynamic analysis by running the application in a controlled environment and monitoring its behavior, and identifying and enumerating services, libraries, and components used by the application. Additionally, gathering information about the application's configuration, such as file paths, registry entries, and network communications, is crucial. Tools like Dependency Walker, CFF Explorer, Procmon, and Wireshark can be employed in this phase.

3. Vulnerability Assessment

The objective of the vulnerability assessment phase is to perform a comprehensive vulnerability assessment to identify specific vulnerabilities in the application. This involves using automated tools and manual techniques to identify common vulnerabilities and misconfigurations. The scan results are manually reviewed to validate and prioritize the identified vulnerabilities based on their severity and potential impact. This phase helps in identifying specific vulnerabilities that can be exploited to gain unauthorized access or compromise the target systems. Tools such as Ghidra, IDA Pro and Burp Suite can be utilized in this phase.

4. Exploitation

The objective of the exploitation phase is to attempt to exploit identified vulnerabilities to demonstrate their potential impact. Ethical hacking techniques are utilized to simulate real-world attack scenarios while ensuring no harm is caused to the target systems. The pentester may use various tools, scripts, or custom exploits to exploit the identified vulnerabilities. The goal is to demonstrate the potential impact of the vulnerabilities and assess the effectiveness of the target's security controls. Documenting successful exploits and their impact is crucial in this phase.

5. Reporting

In this final phase, the pentester compiles all findings, categorizes them based on severity levels, and provides detailed explanations, proof-of-concept demonstrations, and prioritized recommendations for remediation. The report includes a summary of the pentest engagement, an overview of the methodology used, and a comprehensive analysis of the vulnerabilities discovered. It also includes actionable recommendations to mitigate the identified vulnerabilities and improve the overall security posture of the target systems. The report serves as a valuable resource for the client to understand the security risks and take appropriate measures to address them.

Vulnerability Classification & Severity

To categorize vulnerabilities according to a commonly understood vulnerability taxonomy, KLEAP Technologies Pvt. Ltd. uses the industry-standard Common Weakness Enumeration (CWE). CWE is a community-developed taxonomy of common software security weaknesses. It serves as a common language, a measuring stick for software security tools, and as a baseline for weakness identification, mitigation, and prevention efforts.

To rate the severity of vulnerabilities, KLEAP Technologies Pvt. Ltd. uses the industry standard Common Vulnerability Scoring System (CVSS) to calculate severity for each identified security vulnerability. CVSS provides a way to capture the principal characteristics of a vulnerability, and produce a numerical score reflecting its severity, as well as a textual representation of that score.

To help prioritize vulnerabilities and assist vulnerability management processes, KLEAP Technologies Pvt. Ltd. translates the numerical CVSS rating to a qualitative representation (such as low, medium, high and critical):

CVSS Score v3.1	
Severity	**Score**
Critical	9.0 - 10.0
High	7.0 - 8.9
Medium	4.0 - 6.9
Low	0.1 - 3.9
Informational	0.0

Findings Summary

Findings are sorted by their severity and grouped by the asset and CWE classification. Each asset section will contain a summary. Table 1 in the executive summary contains the total number of identified security vulnerabilities per asset per risk indication.

Findings Overview

During the engagement, 3 unique vulnerabilities were found across 2 different vulnerability categories. Vulnerabilities of the following kinds were identified:

- Injection
- Sensitive Data Exposure

Exploring the findings further by their actual vulnerability type as defined by CWE, Table 3 shows the number of individual findings and its distribution of severity.

Vulnerabilities	Critical	High	Medium	Low	Info
SQL Injection	1	0	0	0	0
Side Channel Data Leaks	0	1	0	0	0
Dumping connection string from memory	0	0	1	1	0
	1	1	1	0	0

Table 3: Severity distribution across vulnerability types

Findings Overview as per OWASP Standards

During the engagement, 5 unique vulnerabilities were found across mainly 3 different vulnerability categories. The most common vulnerability types identified were:

- Broken Object Level Authorization
- Injection
- Security Misconfiguration

Exploring the findings further by their actual vulnerability type as defined by CWE, Table 3 shows the number of individual findings and its distribution of severity.

OWASP Top 10	Results	Findings
DA1 - Injections	Fail	1
DA2 - Broken Authentication & Session Management	Pass	0
DA3 - Sensitive Data Exposure	Fail	2
DA4 - Improper Cryptography Usage	Pass	0
DA5 - Improper Authorization	Pass	0
DA6 - Security Misconfiguration	Pass	0
DA7 - Insecure Communication	Pass	0
DA8 - Poor Code Quality	Pass	0
DA9 - Using Components with Known Vulnerabilities	Pass	0
DA10 - Insufficient Logging & Monitoring	Pass	0

Technical Findings Details

01: Sensitive Data Disclosure

Vulnerability Severity	CWE ID
Critical	89
OWASP Category	CVSS Score
DA1 - Injections	9.1

Vulnerability Description

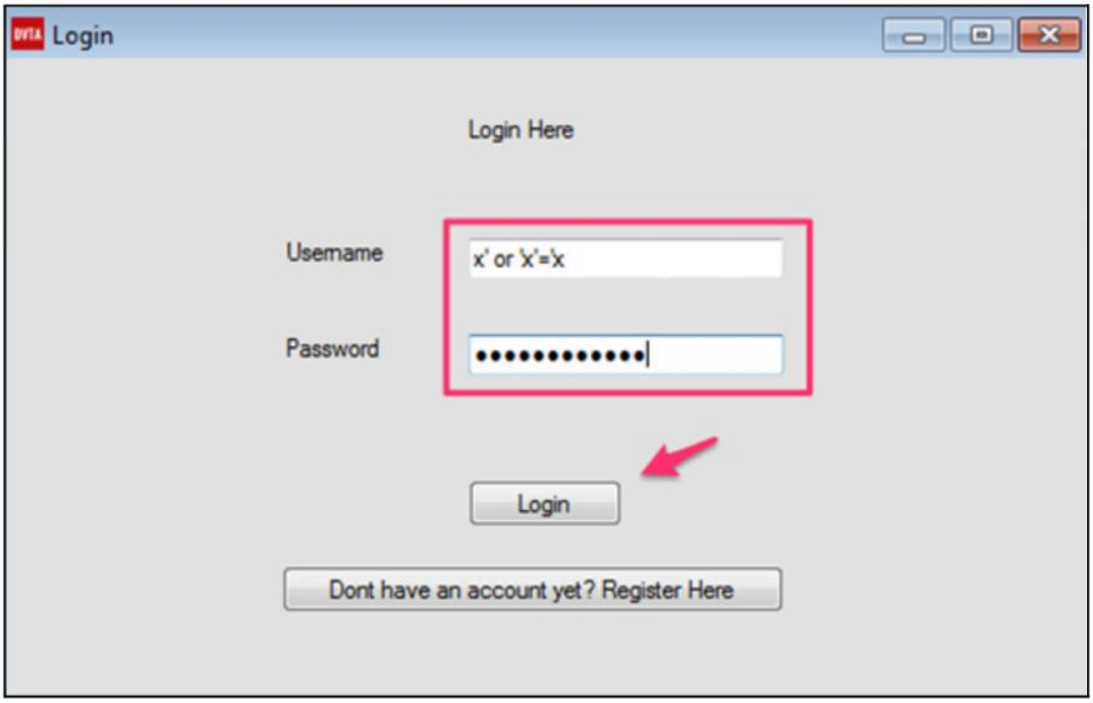
SQL injection, also known as SQLI, is a common attack vector that uses malicious SQL code for backend database manipulation to access information that was not intended to be displayed.

Impact

An attacker can execute arbitrary SQL queries, potentially leading to data exfiltration or unauthorized access.

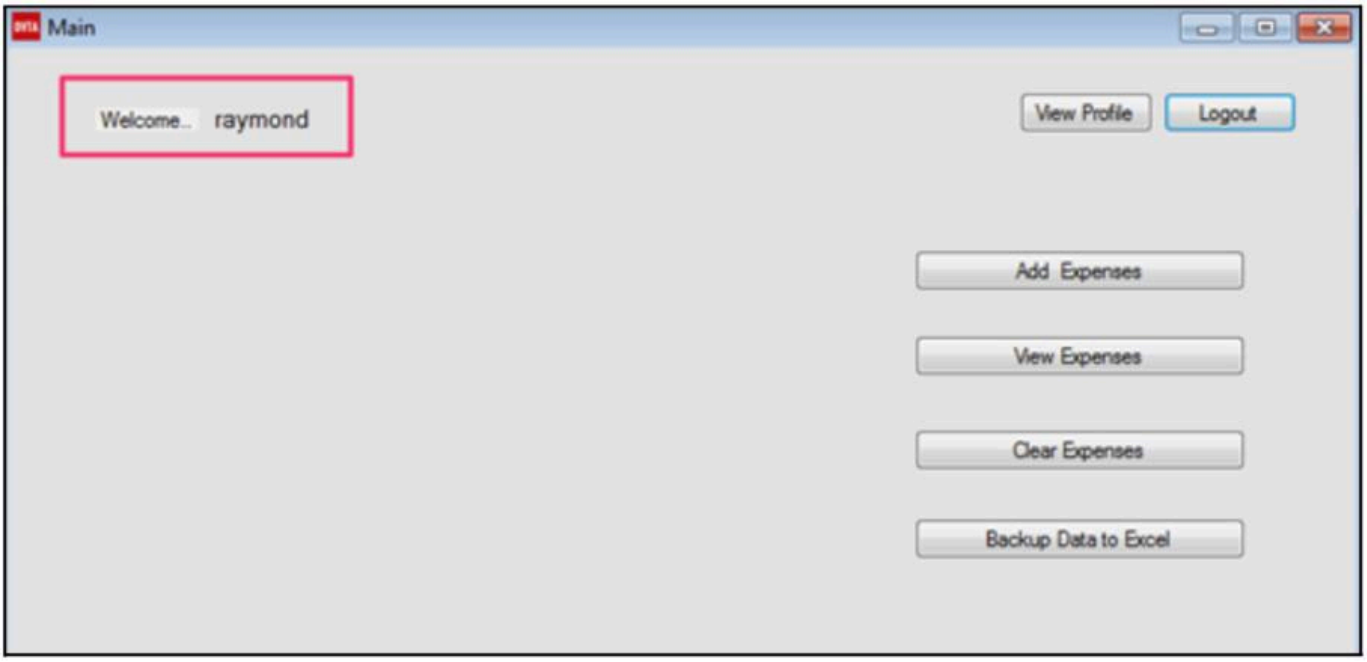
Steps to Reporduce

1. Enter x' or 'x'='x in both the username and password fields as shown in the following figure and then click Login.



Steps to Reporduce

2. Observe that we are logged in as Raymond.



Remediation

- Implement prepared statements and parameterized queries to prevent SQL injection.
- Validate and sanitize all user inputs.

02: Side Channel Data Leaks

Vulnerability Severity	CWE ID
High	200
CVE ID	CVSS Score
DA3 - Sensitive Data Exposure	8.7

Vulnerability Description

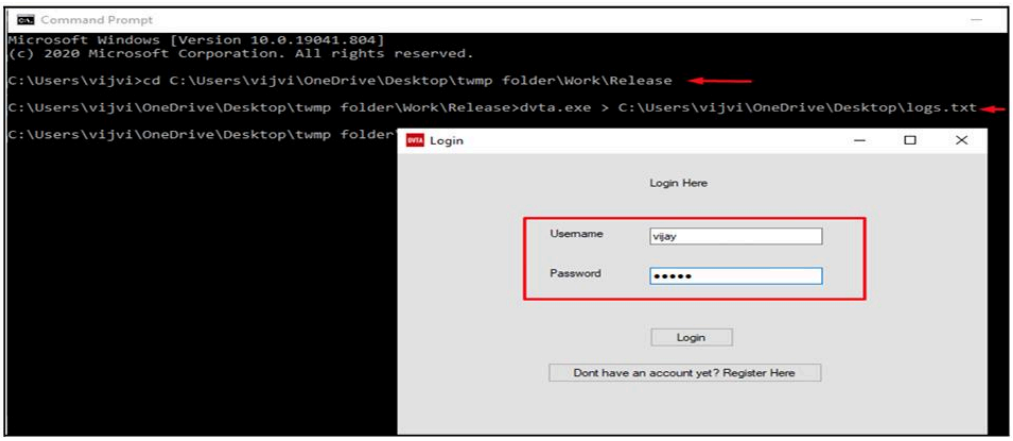
Side-channel data leakage, also known as unintended data leakage, is a vulnerability that occurs when sensitive information is unintentionally exposed due to how software or hardware is used. This can happen when information is placed in an insecure location on a device, such as due to caching, logging, or browser cookies.

Impact

- **Sensitive Data Exposure:** Leaks can reveal passwords, encryption keys or other critical information.
- **System Compromise:** Enables attackers to gain unauthorized access.
- **Privacy Violations:** Exposes personal or confidential information.
- **Financial Loss:** Leads to fraud or unauthorized transactions.
- **Reputation Damage:** Erodes trust and harms the organization's image.

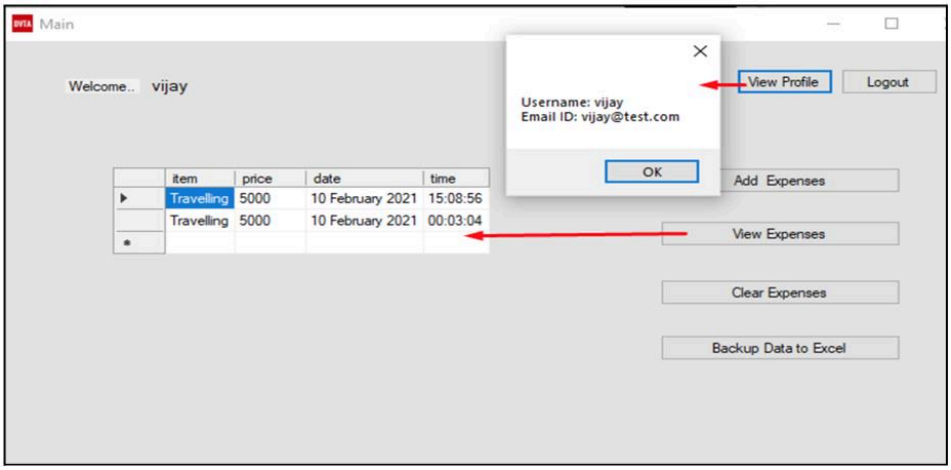
Step to Reproduce (Evidences)

1. Login to the user account "vijay" as shown in the POC.

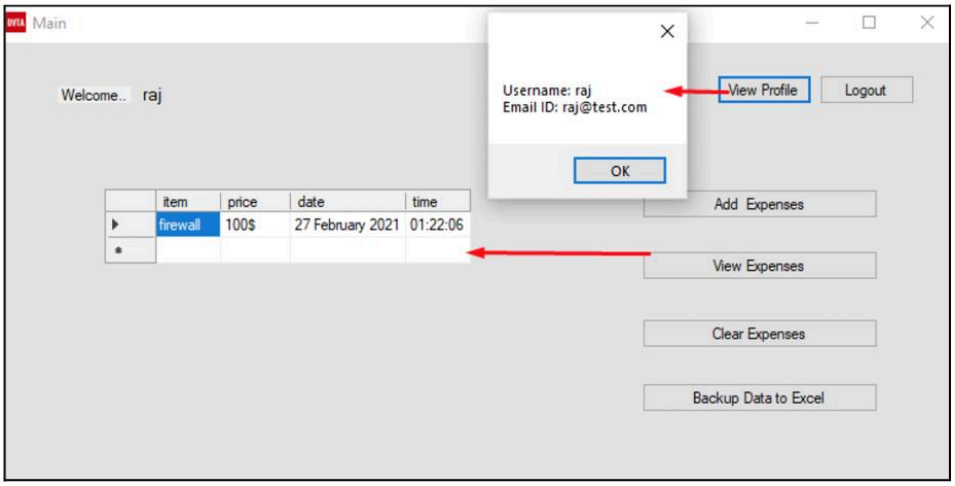


2. After logging in to the application, click on "View Expenses" then click on "View Profile" and finally log out.

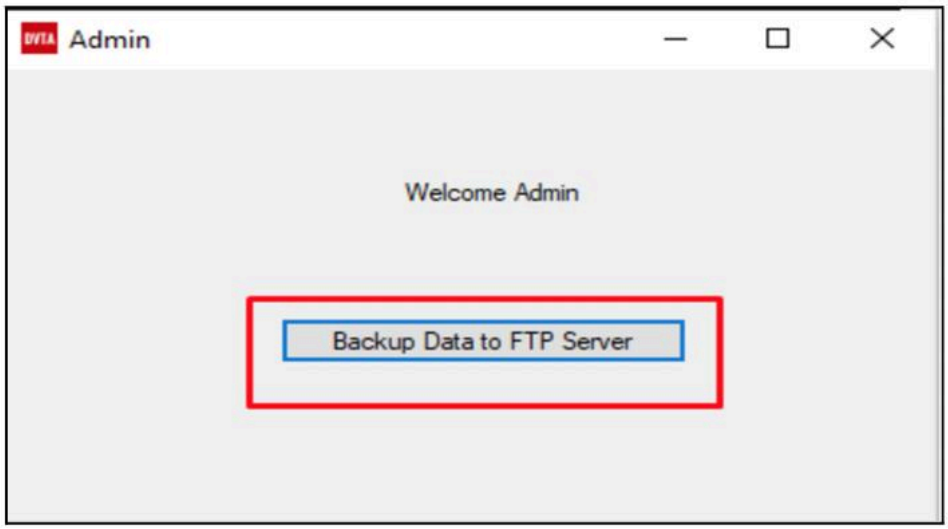
Step to Reproduce (Evidences)



3. Next, log in to different accounts such as “Raj”, explore the application and then log out.

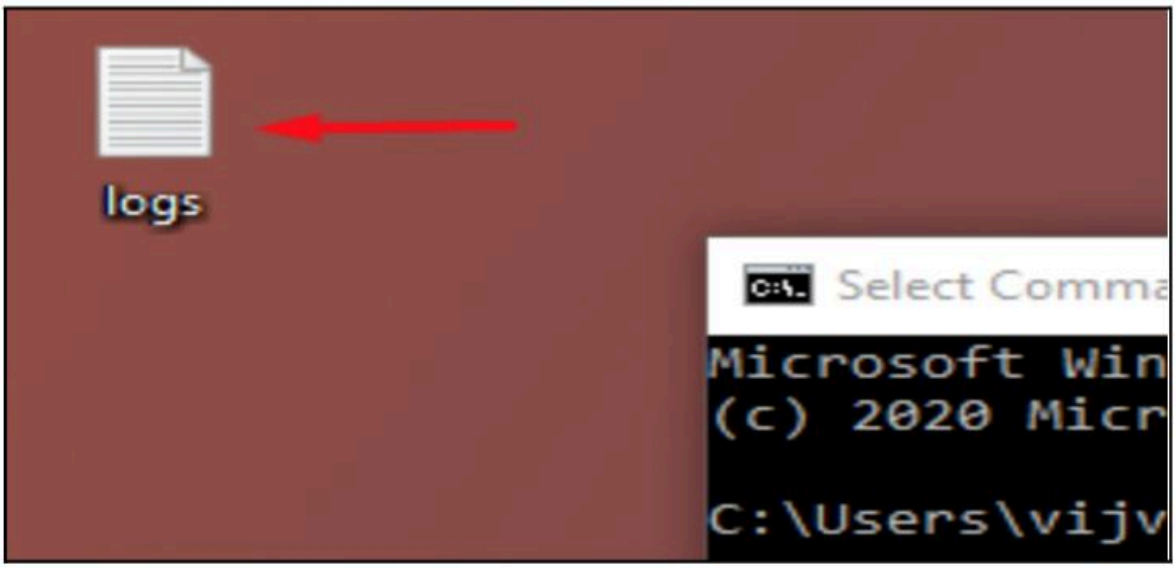


4. Similarly, log in as an admin user and click on “Backup Data to FTP Server”.

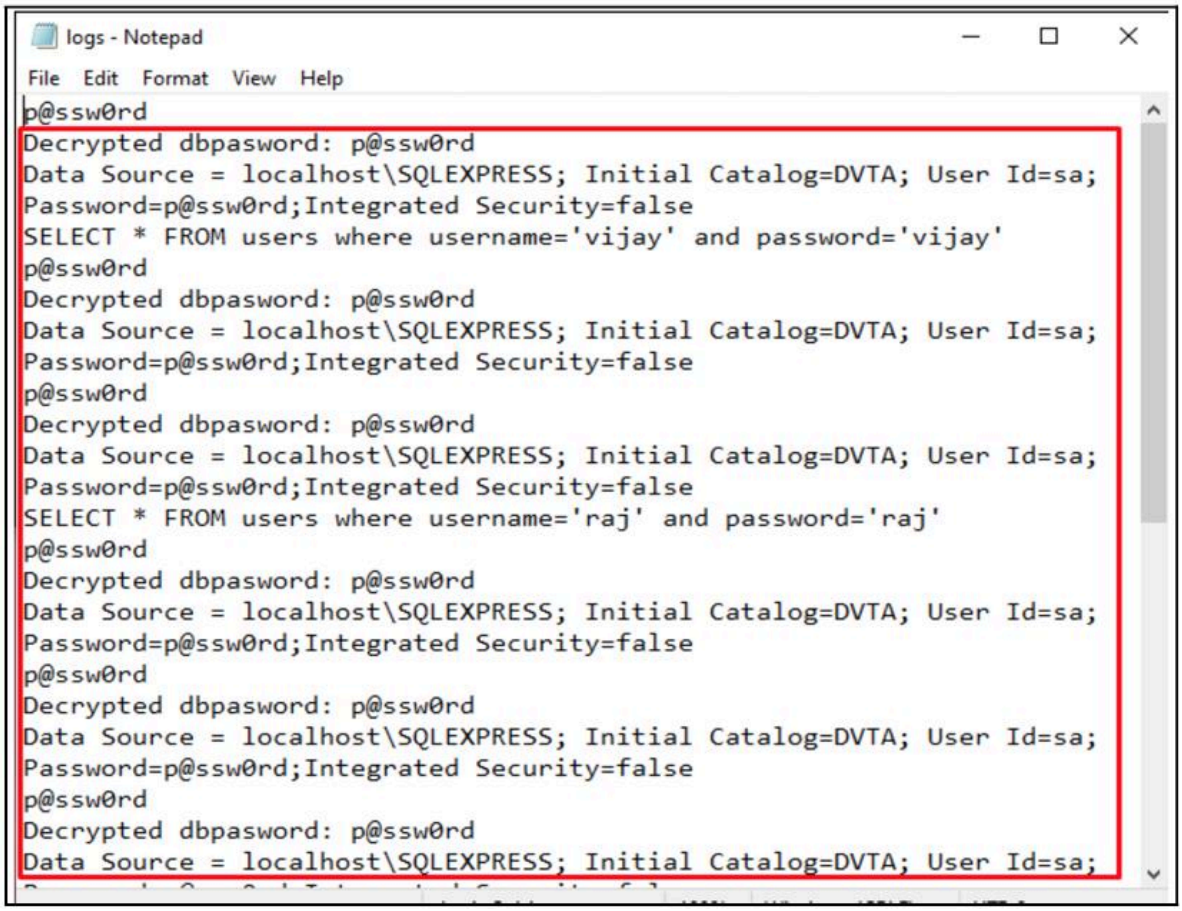


5. We're simply browsing the application to generate logs, which will create a log file on the desktop.

Step to Reproduce (Evidences)



6. Open the file in Notepad where you can see details like database password, connection string, and SQL queries.



Remediation

- Remove or sanitize sensitive information from logs.
- Implement proper logging practices to prevent data leaks.

03: Dumping connection string from memory

Vulnerability Severity	CWE ID
Medium	316
CVE ID	CVSS Score
DA3 - Sensitive Data Exposure	5.6

Vulnerability Description

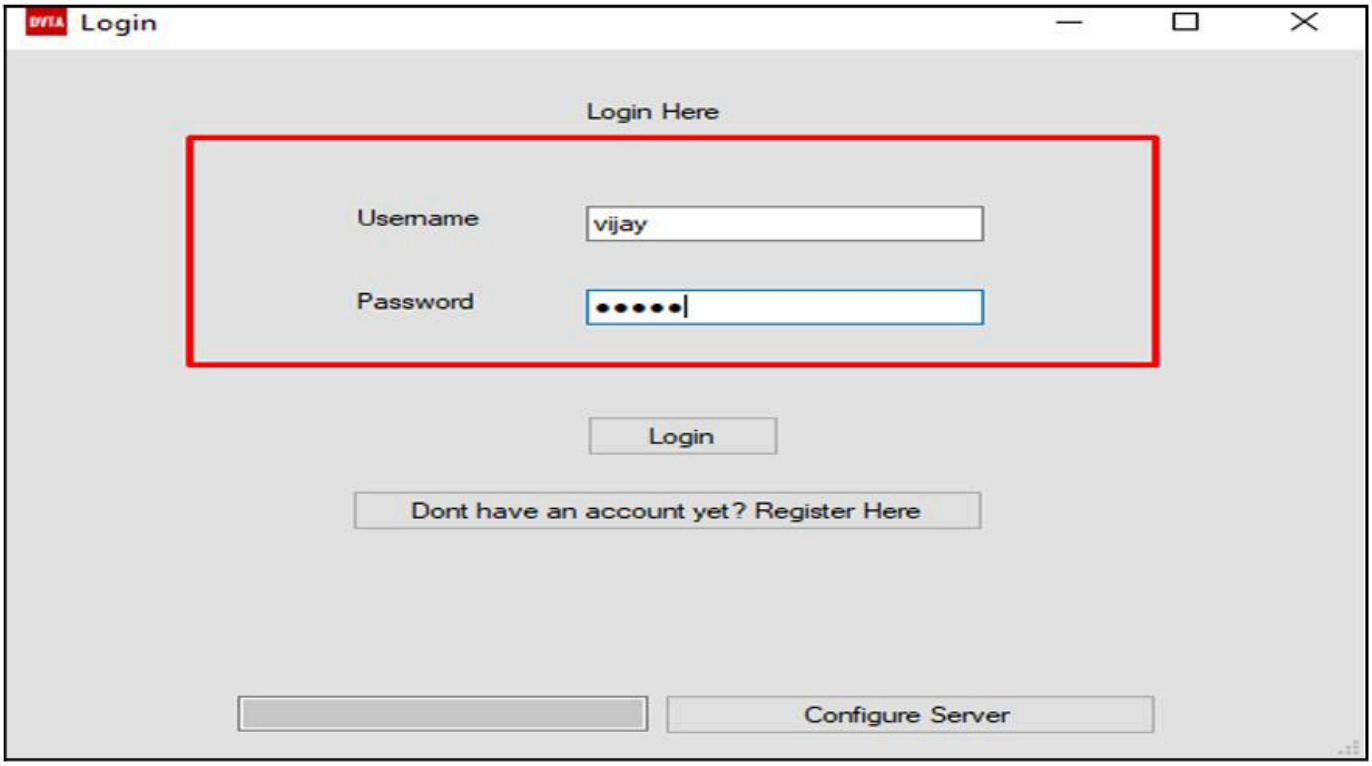
The data storage issue that is predominantly seen in the 2-tier application is finding database connection strings in memory.

Impact

Hard-coded secrets are a security risk because they are often stored in plain text, making it easy for attackers to extract them from the source code. They can also be inadvertently disclosed or exposed through other security vulnerabilities, such as code injection or data leaks.

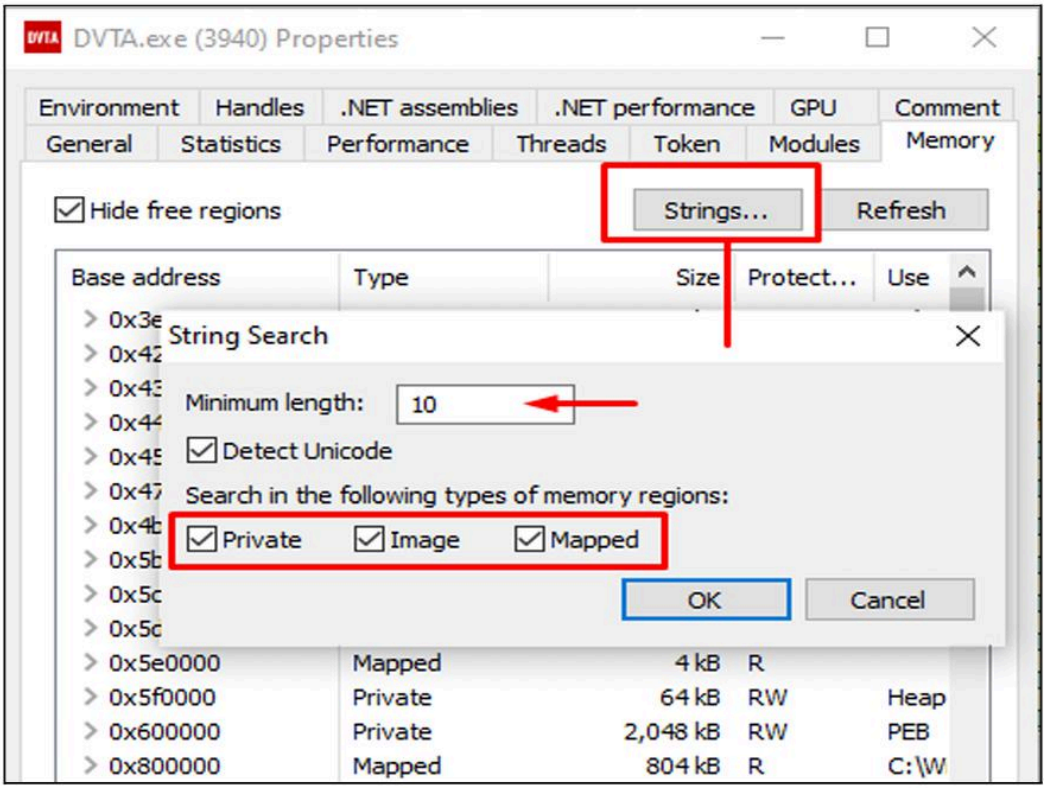
Step to Reproduce (Evidences)

1. Log in to the DVTA program after opening it.

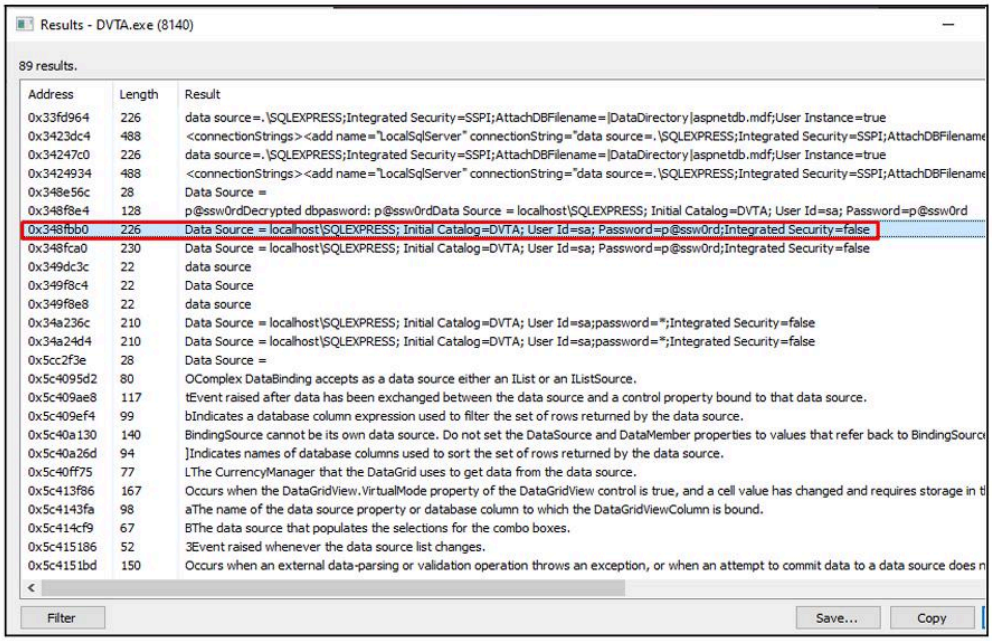


Step to Reproduce (Evidences)

2. Open the properties of DVTA.exe using Process Manager, and then select the Memory tab. After clicking on the strings, select “mapped” and “image”.

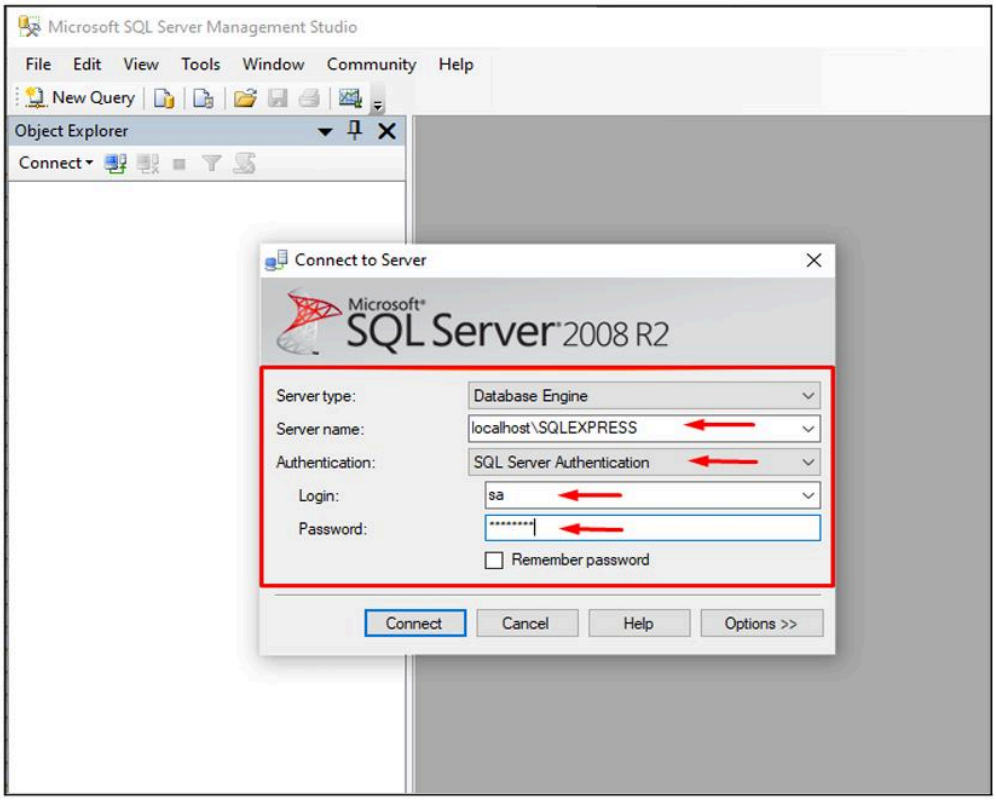


3. Additionally, it must locate every string in this process's memory. Therefore, we will apply the filter using the keyword “Data source”, which is commonly seen in database connection strings. Next, we discovered SQL Cred. in unformatted form.

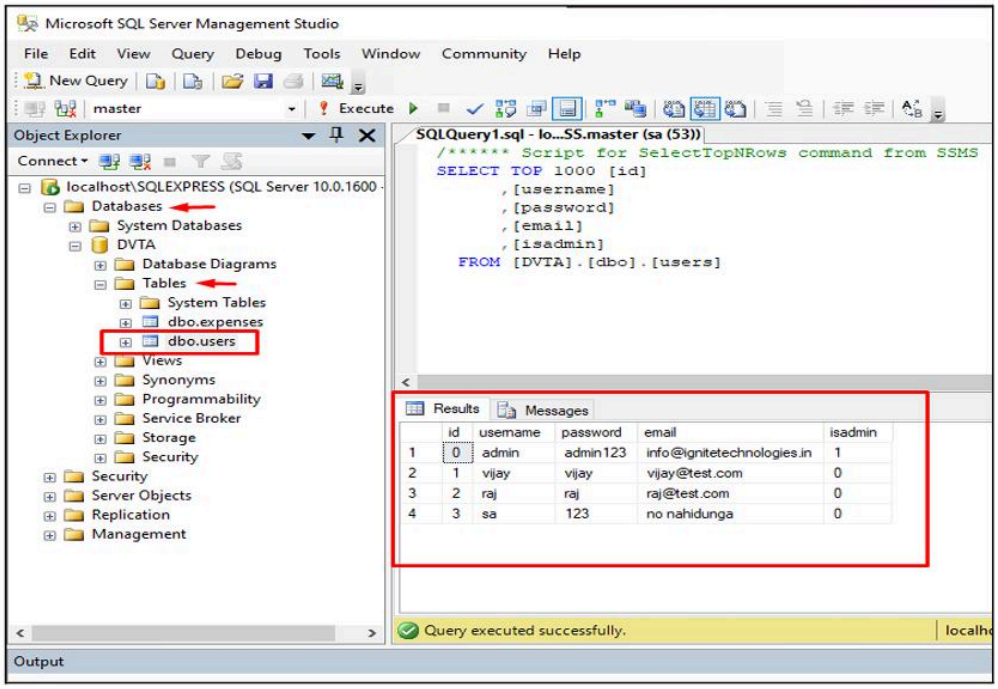


4. To access the hardcoded credentials, we utilize “SQL Server Management Studio” to log in.

Step to Reproduce (Evidences)



5. As you can see, we have a connection to the database and may now browse its tables or extract sensitive information from it.



Remediation

- Ensure that sensitive information is not stored in memory in plain text.
- Implement encryption and proper memory management practices to protect sensitive data.

KLEAP

CYBERSECURITY

https://kleapcybersecurity.com/

info@kleapcybersecurity.com

4111, Briargrove Circle, Raleigh,
North Carolina, 27607, USA