

ESIEA LOURD

Software & Database Security

Final Report

Shannon Dsouza
10-18-2020

Index

Synthesis	2
Vulnerabilities	3
Vulnerability 1: Master Password stored in application source code.	3
Vulnerability 2: Database login information available in config file.....	5
Vulnerability 3: Application's network traffic is unencrypted.	7
Vulnerability 4: Passwords are stored in plain text in the database.	9
Vulnerability 5: Lack of minimal requirements for passwords.	10
Vulnerability 6: Passwords are not masked when entered.....	11
Vulnerability 7: SQL injection is possible in multiple queries.	12
Vulnerability 8: Lack of user privilege separation.....	14
Vulnerability 9: Leakage of database information (table names, column names).	15
Vulnerability 10: System command injection.....	16
Vulnerability 11: Sessions never expire.....	17
Vulnerability 12: Credentials remain in memory after use.....	18
Vulnerability 13: Tampering of SQL queries.	19

Synthesis

Scope: The testing of vulnerabilities in the Esica Lourd application and database. All methods of exploiting the application are explored.

Vulnerability categories:

There are 5 critical vulnerabilities, 4 medium impact vulnerabilities, and 3 low impact vulnerabilities discovered.

The critical vulnerabilities can impact the business significantly is exploited. These vulnerabilities usually allow unfettered access to the database or to the system.

The medium impact vulnerabilities can impact the business but not as significantly as the critical vulnerabilities. These usually reveal information that a particular user should not have access to.

The low impact vulnerabilities have minimal impact on the business and minor security breaches.

Recommendations:

- 1) The primary recommendation is to alter the system architecture from a 2-tier model to a 3-tier model. Should this be too complex or expensive, implement a simulated 3-tier architecture using citrix.
- 2) Remove all locally stored passwords and credentials from the application and files, only store credentials on the server and perform all authentications at the server.
- 3) Encrypt all network traffic between the application and the database/server.
- 4) Do not allow the application to run system commands through it.
- 5) Do not overlook general safety precautions such as masking passwords on input, use of strong passwords, and purging passwords from memory after use.

Vulnerabilities

Vulnerability 1: Master Password stored in application source code.

Criticality: Medium

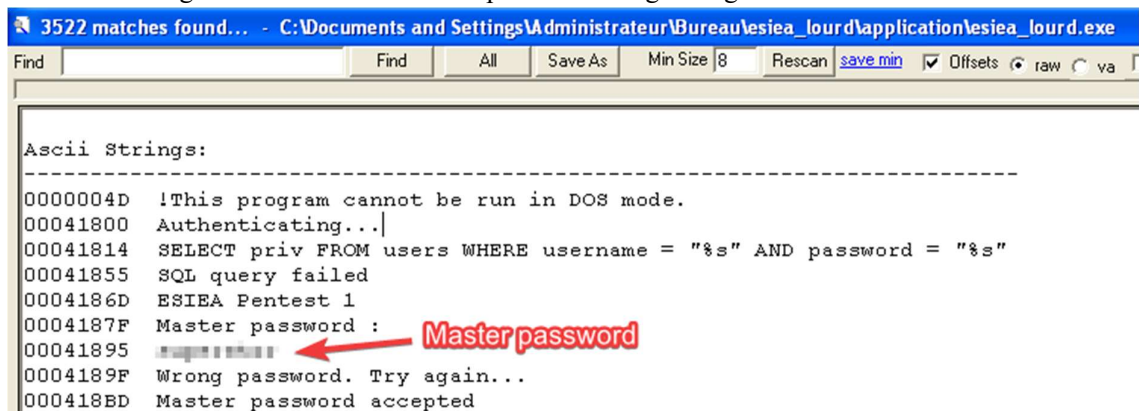
- a) Business impact: The ease with which the application can be unlocked raises concern but as user logins are required to use any of the functions of the application, the business impact is medium.
- b) Exploitability: The vulnerability is extremely easy to exploit as the master password is stored right in the source code and is readable by anyone that has access to the application.
- c) Remediation: The vulnerability can be fixed quite easily. The logon protocol must be altered to use server-side authentication just like the user logins so that the master password does not have to be stored in the application. The remediation will not require a large increase in budget or a lot of manpower as the system is already in place for user logins.

Description:

The master password is stored in the application. It can easily be read by anyone by reading the source code. It requires minimal effort to find the master password and allows anyone to unlock the application. However, as the functions of the applications are locked behind user logins, the impact is not very high as the attacker cannot use the application with the master password alone.

Exploitation:

- a) Either open the application in a text editor or use a software such as 'strings' to quickly identify strings in the code.
- b) Search the strings for secret information in plain-text using 'strings'.

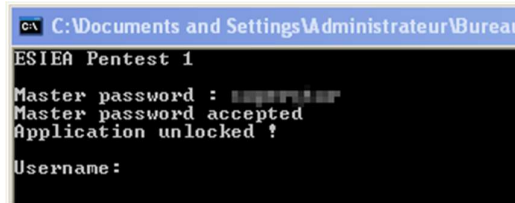


```
3522 matches found... - C:\Documents and Settings\Administrateur\Bureau\esiea_lourd\application\esiea_lourd.exe
Find Find All Save As Min Size 8 Rescan save min Offsets raw va

Ascii Strings:
-----
0000004D !This program cannot be run in DOS mode.
00041800 Authenticating...|
00041814 SELECT priv FROM users WHERE username = "%s" AND password = "%s"
00041855 SQL query failed
0004186D ESIEA Pentest 1
0004187F Master password :
00041895 
0004189F Wrong password. Try again...
000418BD Master password accepted
```

The password stored in plain-text in the source code

c) Unlock the application with the password.

A screenshot of a Windows command prompt window. The title bar shows the path 'C:\Documents and Settings\Administrateur\Bureau'. The command prompt displays the following text: 'ESIEA Pentest 1', 'Master password : [REDACTED]', 'Master password accepted', 'Application unlocked !', and 'Username:'.

```
C:\Documents and Settings\Administrateur\Bureau>
ESIEA Pentest 1
Master password : [REDACTED]
Master password accepted
Application unlocked !
Username:
```

Unlocking the application with the master password

Remediation:

Storing passwords in the application will always face the same problem. Instead you must use the same method as used for the user-logins and only store the passwords on the server.

Reference:

https://owasp.org/www-community/vulnerabilities/Use_of_hard-coded_password

Vulnerability 2: Database login information available in config file.

Criticality: High

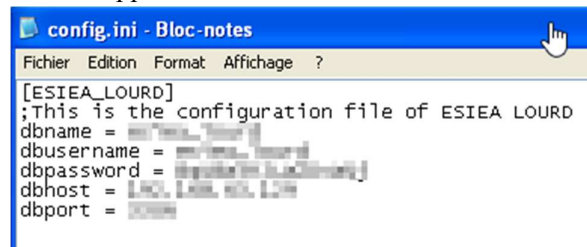
- a) **Business impact:** The config file contains all the information required to directly access the database and view and edit all the information stored in it. The business impact of such unfettered access is high.
- b) **Exploitability:** The database login information is present in clear text in the config file and it is all anyone needs to connect to the database. This vulnerability is very critical and easy that it could cause data loss even by accident by someone not trying to do anything malicious.
- c) **Remediation:** The entire application must be rebuilt as a 3-tier system. This can be expensive and will require a lot of manpower as it is almost the same as rebuilding the entire system.

Description:

The credentials required to access the database are stored in plain text in the config.ini file. It contains the login credentials, database server IP address, and port. This is all the information required for anyone to use any available database tool and connect directly to the database.

Exploitation:

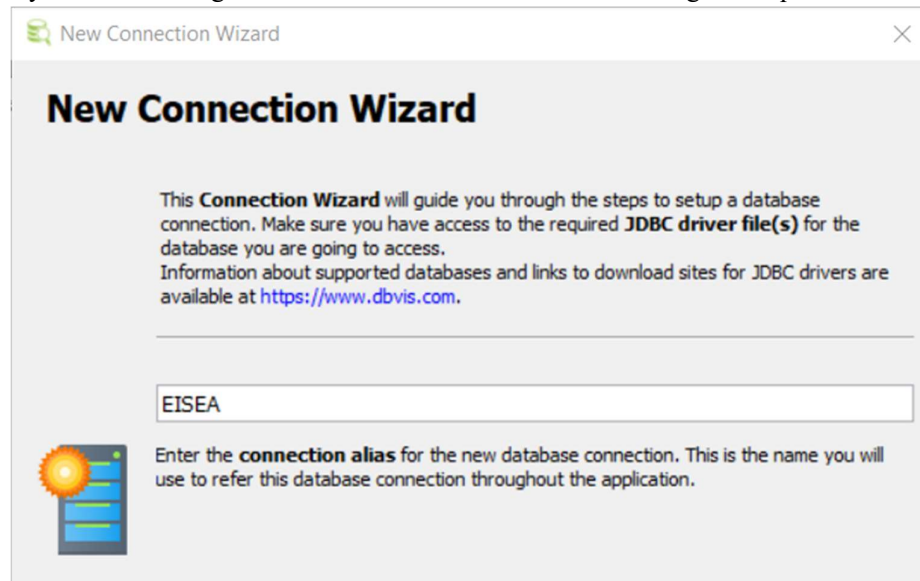
- a) Open the config.ini file in the application folder to reveal all the database credentials.



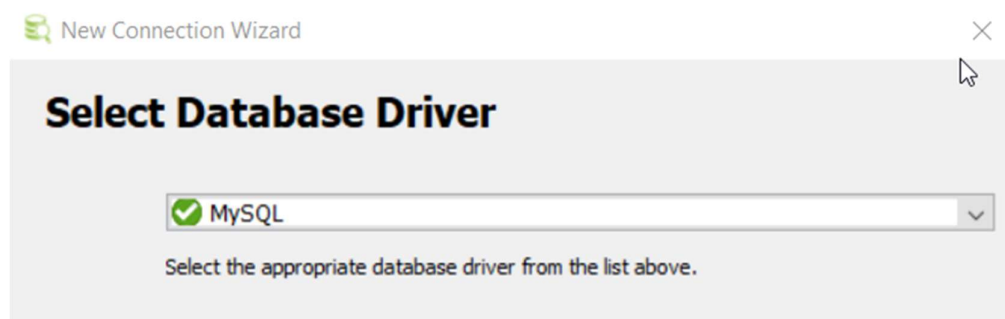
```
[ESIEA_LOURD]
;This is the configuration file of ESIEA LOURD
dbname = ESIEA_LOURD
dbusername = ESIEA_LOURD
dbpassword = ESIEA_LOURD
dbhost = 192.168.1.101
dbport = 1521
```

The config.ini file containing the database credentials.

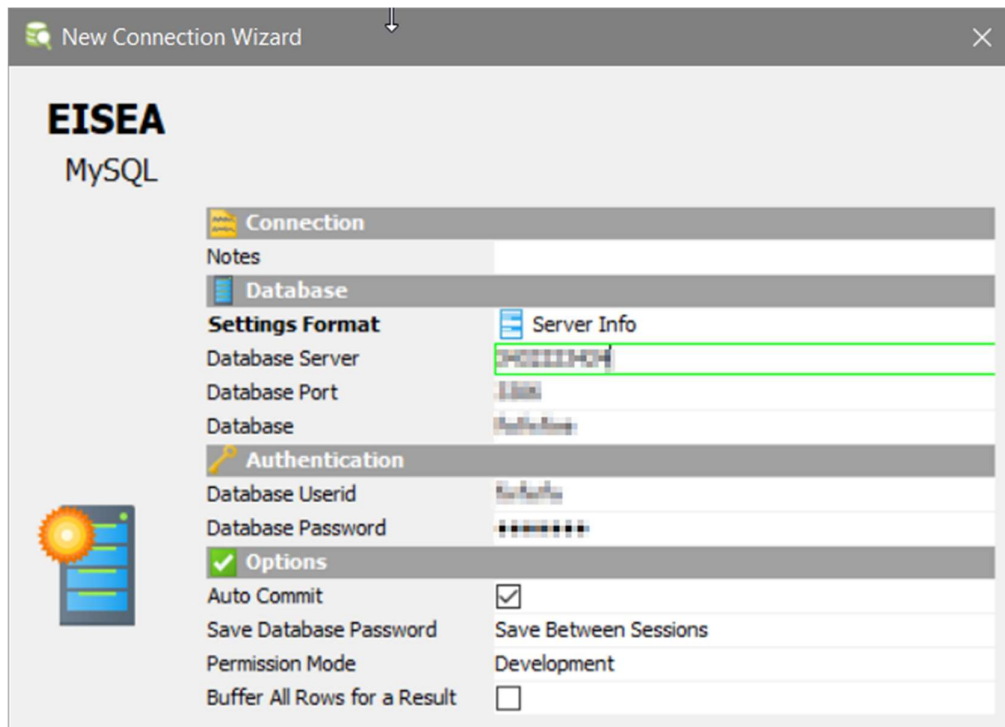
- b) Use any database management tool to connect to the database using the acquired credentials.



Using DbVisualizer to connect to the database



Selecting the Database Driver



Enter the acquired credentials to gain full access to the database

Remediation:

The database credentials must not be available in plain text as a separate file but instead the application must be redesigned to use a 3-tier system where the application communicates with a server, which in turn contains all the credentials and communicates with the database. With such a system, none of the credentials or secrets need to be with the client or the application.

Alternatively, a system such as citrix which simulates 3-tier architecture could be used for the same result as well.

Reference:

<https://www.sciencedirect.com/topics/computer-science/credential-store>

Vulnerability 3: Application's network traffic is unencrypted.

Criticality: High

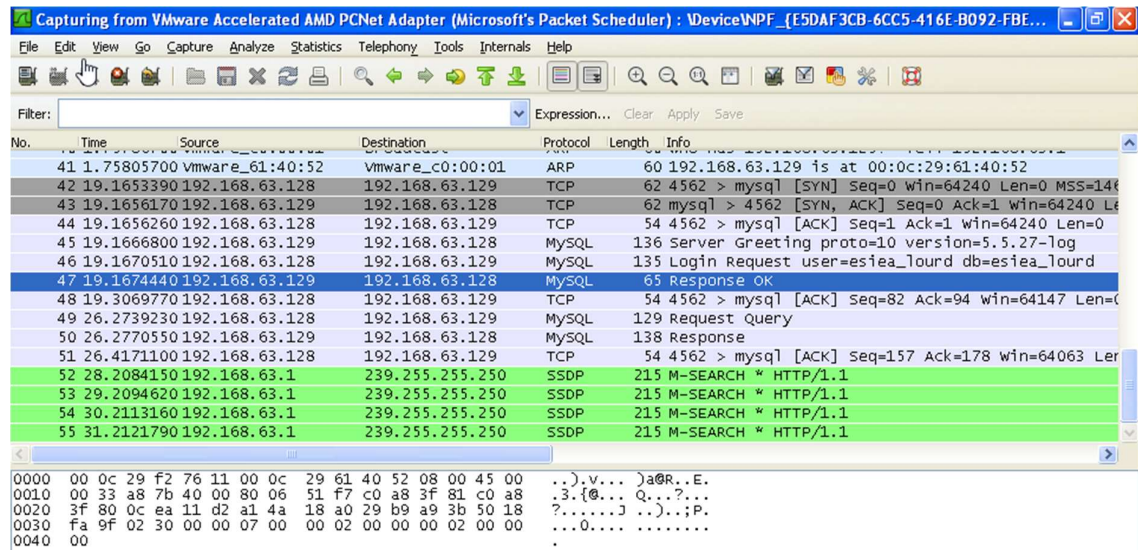
- Business impact:** All the network traffic between the application and the database is unencrypted. Anyone sniffing the network can read all the information, sensitive or not, that passes between them. This means the business impact of this vulnerability is very high.
- Exploitability:** This vulnerability is extremely easy to exploit. Anyone with a network sniffing tool like Wireshark can see every message communicated between the application and the database. It requires no special skills or tools, just access to the network.
- Remediation:** The network traffic must be encrypted. This bears additional cost and manpower to add encryption capabilities to the software.

Description:

All the network traffic is unencrypted and visible in plain text to anyone sniffing the network. This is a critical vulnerability as logins, sensitive information, and insensitive information are accessible to everyone.

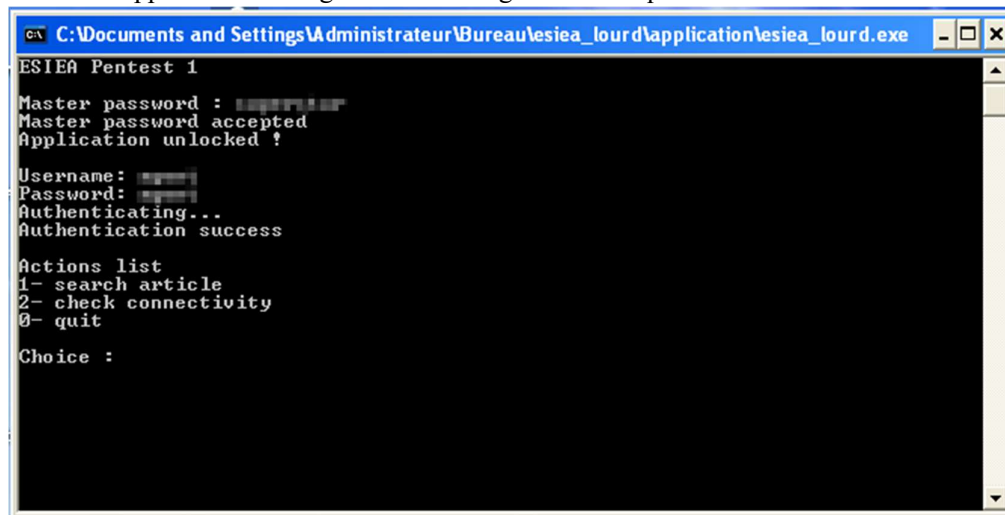
Exploitation:

- Launch Wireshark to sniff the network traffic.



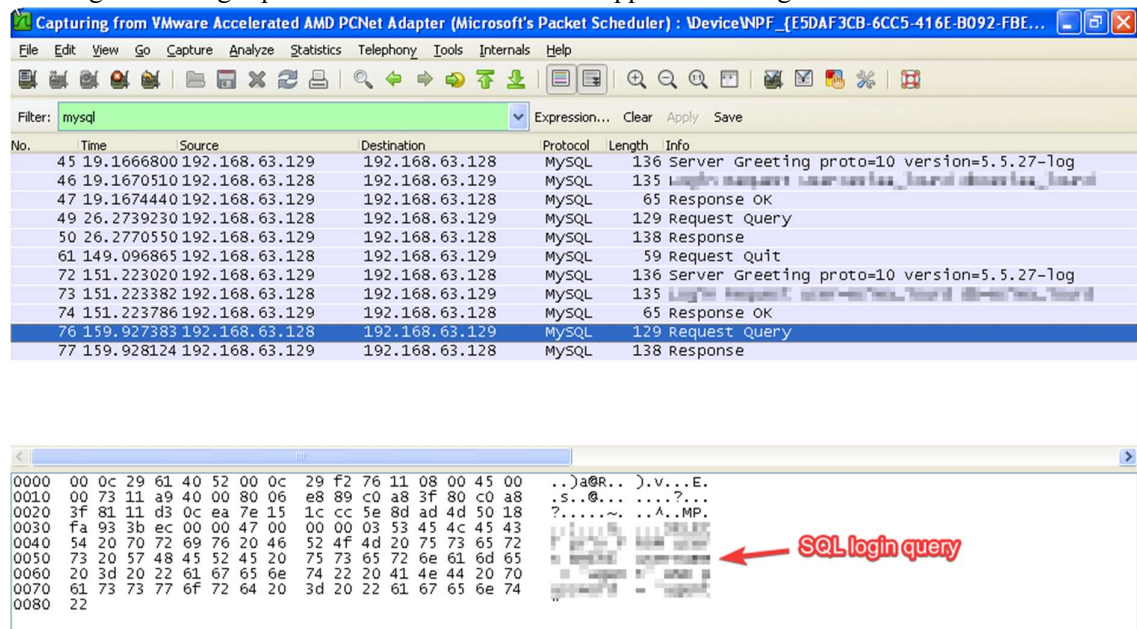
Wireshark sniffing network traffic

- b) Launch the application and log in as a user to generate a request to the database.



Connecting to the database from the application

- c) Reading the messages passed between the server and application using Wireshark.



Plain text SQL query containing the user id and password on Wireshark

Remediation:

Use SSL to encrypt all network traffic between the application and database. There is no other way to fix this vulnerability besides using encryption. It is also good practice to use encryption when sending information over a network.

Reference:

<https://www.informit.com/articles/article.aspx?p=169578&seqNum=4>

Vulnerability 4: Passwords are stored in plain text in the database.

Criticality: High

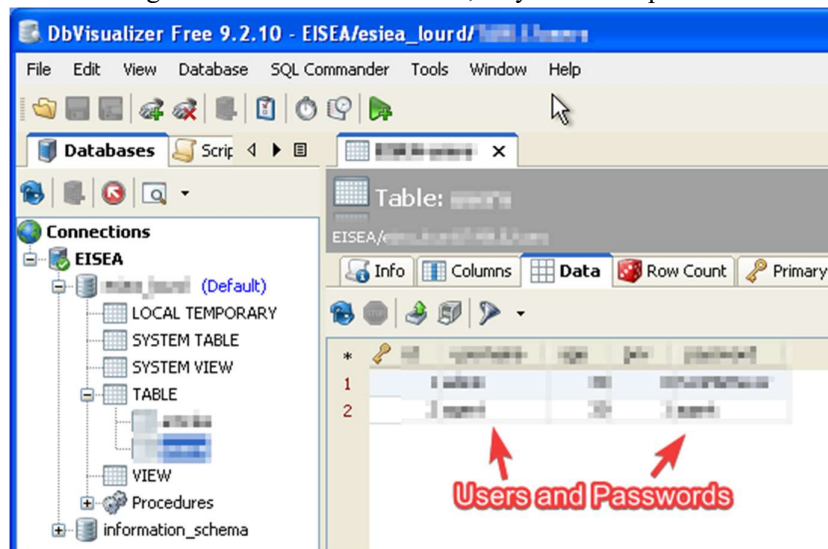
- a) Business impact: Should the database be breached; the passwords of every user will be stolen. The attacker would then have access to every user's account, the impact of this is high.
- b) Exploitability: This vulnerability is not directly exploitable but depends on an attacker gaining access to the database before exploiting this vulnerability. If sufficient measures are in place to secure database access, this vulnerability may never be exploited.
- c) Remediation: The passwords must be stored in the database as hashes and not in plain text. The cost and additional work required would be minimal.

Description:

The passwords are stored in plain text in the database. If an attacker were to gain access to the database, they would have access to every password and consequently every account. They may also try these login credentials for different services to test if passwords were reused.

Exploitation:

- a) Once the attacker has gained access to the database, they check the passwords column.



The usernames and passwords are visible in plain text.

Remediation:

Store the passwords as hashes in the database. Never store them in plain text. Using a salt provides additional security that even the hashes will not be crackable should they be stolen.

Reference:

<https://www.vaadata.com/blog/how-to-securely-store-passwords-in-database/>

Vulnerability 5: Lack of minimal requirements for passwords.

Criticality: Medium

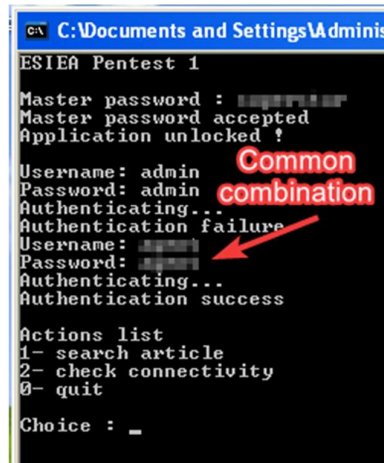
- a) Business impact: The lack of minimal requirements means that users may use extremely simple passwords such as '123' or 'qwerty'. If access is gained to systems through these passwords there would be some impact on the business.
- b) Exploitability: Weak passwords can easily be brute-forced and therefore this vulnerability is easily exploitable.
- c) Remediation: Introducing minimal requirements for passwords will only mean adding a few rules users must satisfy when adding passwords. This will not require an increased budget or much manpower.

Description:

Simple passwords are easily cracked especially if they are common words or phrases. A brute-force attack is very simple to perform and therefore this vulnerability is easily exploitable. Once the attacker has successfully gained access to an account he can do as he likes with that user's privileges.

Exploitation:

- a) Attempt to access accounts with common and unsafe combination such as 'admin/admin'.



```
CA C:\Documents and Settings\Adminis
ESIEA Pentest 1
Master password : 
Master password accepted
Application unlocked !
Username: admin
Password: admin
Authenticating...
Authentication failure
Username: admin
Password: admin
Authenticating...
Authentication success

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

Choice : _
```

The screenshot shows a terminal window titled 'CA C:\Documents and Settings\Adminis'. It displays the output of a program called 'ESIEA Pentest 1'. The program prompts for a 'Master password', which is accepted, and then 'Application unlocked !'. It then prompts for 'Username' and 'Password'. The first attempt with 'admin' for both fails with 'Authentication failure'. The second attempt, also with 'admin' for both, succeeds with 'Authentication success'. Below this, an 'Actions list' is shown with options: '1- search article', '2- check connectivity', and '0- quit'. The prompt 'Choice : _' is at the bottom. A red arrow points to the second successful login attempt, and the text 'Common combination' is written in red next to it.

Common User/Password combination is accepted

Remediation:

Set up rules when forming new passwords. Rules such as minimal length for passwords along with the use of uppercase letters, lowercase letters and special characters is recommended.

Reference:

[https://techblog.topdesk.com/security/developers-need-know-password-guidelines/#:~:text=This%20refers%20to%20those%20rules,are%20not%20uppercase%20or%20lowercase\)%2C](https://techblog.topdesk.com/security/developers-need-know-password-guidelines/#:~:text=This%20refers%20to%20those%20rules,are%20not%20uppercase%20or%20lowercase)%2C)

Vulnerability 6: Passwords are not masked when entered.

Criticality: Low

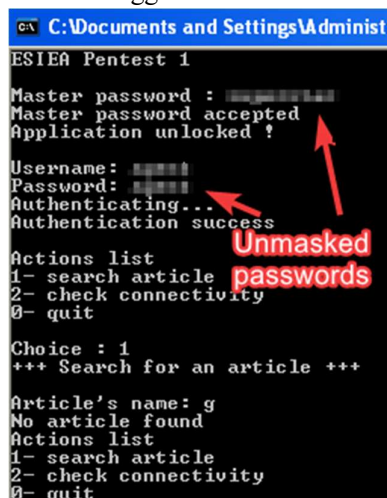
- a) Business impact: If passwords are stolen by shoulder surfing, the attacker may gain access to that user's account and cause harm. This could have some impact on the business.
- b) Exploitability: Reading passwords that are unmasked on a system requires the attacker to have access to that system. This physical requirement makes the vulnerability quite hard to exploit.
- c) Remediation: Mask passwords as they are entered into the application. This will not require an increase in budget or more manpower.

Description:

Unmasked passwords remaining in the application interface is a vulnerability as anyone that gains access to a system that is logged in can read and steal the user's passwords.

Exploitation:

- a) Access the application after a user has logged in and look for their credentials.



The screenshot shows a terminal window titled 'C:\Documents and Settings\Administr...'. The application 'ESIEA Pentest 1' is running. It prompts for a 'Master password' which is masked with asterisks. After acceptance, it asks for 'Username' and 'Password'. The 'Password' is also masked with asterisks. However, the next line shows 'Authentication...' followed by 'Authentication success'. A red arrow points to the masked password input, and another red arrow points to the 'Authentication success' message. A red text label 'Unmasked passwords' is overlaid on the terminal output, indicating that the password is visible in the interface.

```
C:\Documents and Settings\Administr...
ESIEA Pentest 1
Master password : *****
Master password accepted
Application unlocked ?

Username: *****
Password: *****
Authentication...
Authentication success

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

Choice : 1
+++ Search for an article +++

Article's name: g
No article found
Actions list
1- search article
2- check connectivity
0- quit
```

Passwords remain unmasked in the application

Remediation:

Mask the passwords as soon as they are entered into the application. Passwords must not remain visible for anyone to read it by shoulder-surfing or by gaining access to a logged in system.

Reference:

<https://medium.com/@yulianaz/better-password-masking-for-sign-up-forms-ceedbef09b657>

Vulnerability 7: SQL injection is possible in multiple queries.

Criticality: High

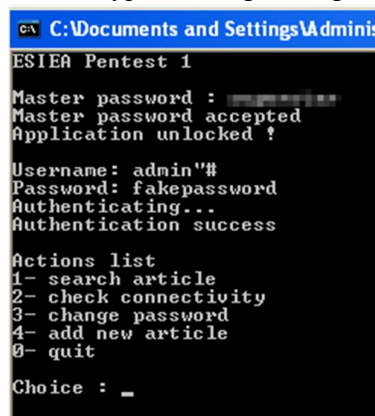
- a) Business impact: SQL injections allow attackers to access data they are not permitted to from the database. This can have a very high impact as sensitive information can be accessed by anyone that has logged in.
- b) Exploitability: SQL injections require knowledge of SQL and certain skills to perform, they are not readily exploitable.
- c) Remediation: Filter SQL queries for special characters, use input validation or escaping to prevent SQL injection. These would require a slight increase in manpower to develop.

Description:

SQL injection is performed by adding an SQL query into a predefined query so that it returns more data than it was supposed to.

Exploitation:

- a) After logging in, use SQL injection to bypass the login and gain access to any account.



```
C:\Documents and Settings\Adminis
ESIEA Pentest 1
Master password : 
Master password accepted
Application unlocked !

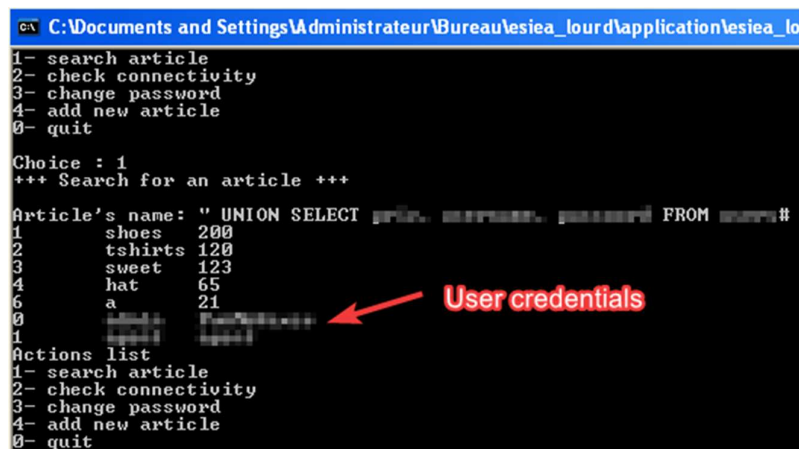
Username: admin"#
Password: fakepassword
Authenticating...
Authentication success

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

Choice : _
```

By adding the “#” after the username, the remainder of the SQL query is ignored, i.e, the password is ignored and the user is authenticated. In this case, admin access is acquired.

- b) By using the search article option another SQL injection is possible, this injection dumps the user credentials using the UNION operator.



```
C:\Documents and Settings\Administrateur\Bureau\esiea_lourd\application\esiea_lo
1- search article
2- check connectivity
3- change password
4- add new article
0- quit

Choice : 1
+++ Search for an article +++

Article's name: " UNION SELECT user, password FROM users#
1 shoes 200
2 tshirts 120
3 sweet 123
4 hat 65
6 a 21
0 
1 
Actions list
1- search article
2- check connectivity
3- change password
4- add new article
0- quit
```

User credentials

By using the query " UNION SELECT x, y, z FROM abc# the column of user credentials are returned as well.

Remediation:

Use input validation or escaping to prevent SQL injection from occurring. These can be implemented quite easily using a php script and SQL injections would no longer work.

Reference:

<https://www.hacksplaining.com/prevention/sql-injection>

Vulnerability 8: Lack of user privilege separation.

Criticality: Medium

- a) Business impact: Users can access functions they are not permitted to and this could harm the business depending on what information is accessed.
- b) Exploitability: Users must be logged in to exploit this vulnerability but the vulnerability itself is exceptionally easy to exploit.
- c) Remediation: Implement proper user privilege separation. This would require some manpower to implement but no new technologies would need to be purchased.

Description:

As user privilege separations are not implemented, a user with lower privileges can access functions which they are not permitted to access by just choosing said function.

Exploitation:

- a) A user logged in with lower privileges tries to access an option not available to them and gains access to it.



```
C:\Documents and Settings\Administrateur\ESIEA Pentest 1
Master password : 
Master password accepted
Application unlocked ?

Username: 
Password: 
Authenticating...
Authentication success

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

Choice : 3
+++ Password changing +++

Old password: xxx
New password: xxx
Old password invalid !
Actions list
1- search article
2- check connectivity
0- quit

Choice : 4
+++ New article addition +++

Article:
```

The user only has permissions to access options 1 and 2. However, the user can access option 3 (change passwords) and option 4 (add new article) by simply choosing those options.

Remediation:

Implement proper user privilege separation instead of simply hiding the options that users should not have access to.

Reference:

<https://www.lepide.com/blog/how-does-privilege-separation-improve-it-security/>

Vulnerability 9: Leakage of database information (table names, column names).

Criticality: Low

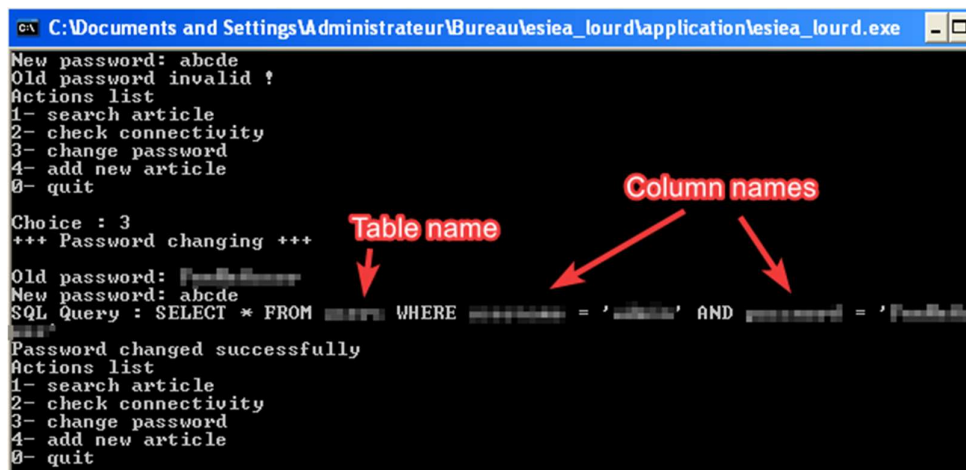
- a) Business impact: If an attacker learns about the database, he can use this information to execute further attacks. The leakage of database information alone will not do any damage to the business.
- b) Exploitability: The vulnerability is very easy to exploit but requires admin privileges.
- c) Remediation: Fix the source-code so that sensitive information is never displayed to the user. It requires minimal manpower to implement and no new technologies need to be purchased.

Description:

This vulnerability displays technical information (the table name and column names) to the user when a particular option is selected. This information can be used by attackers to perform further attacks such as a UNION SQL injection.

Exploitation:

- a) Users must login as admin and attempt to change their password, this return an SQL query which reveals a table name and two column names.



```
C:\Documents and Settings\Administrateur\Bureau\esia_lourd\application\esia_lourd.exe
New password: abcde
Old password invalid !
Actions list
1- search article
2- check connectivity
3- change password
4- add new article
0- quit

Choice : 3
+++ Password changing +++
Old password: 
New password: abcde
SQL Query : SELECT * FROM users WHERE username = 'admin' AND password = 'abcde'
Password changed successfully
Actions list
1- search article
2- check connectivity
3- change password
4- add new article
0- quit
```

The screenshot shows a command prompt window with a blue title bar. The text inside shows a menu-driven application. When option 3 (change password) is selected, it displays an SQL query: `SELECT * FROM users WHERE username = 'admin' AND password = 'abcde'`. Red arrows point from the text "Table name" to the word "users" in the FROM clause, and from the text "Column names" to the asterisk (*) in the SELECT clause.

SQL query revealing the database information

Remediation:

Find where in the source code this leak occurs and edit it so sensitive database information is never displayed to the user.

Reference:

<https://www.cybertec-postgresql.com/en/security-matters-hiding-a-table-column-for-a-quick-fix/>

Vulnerability 10: System command injection.

Criticality: High

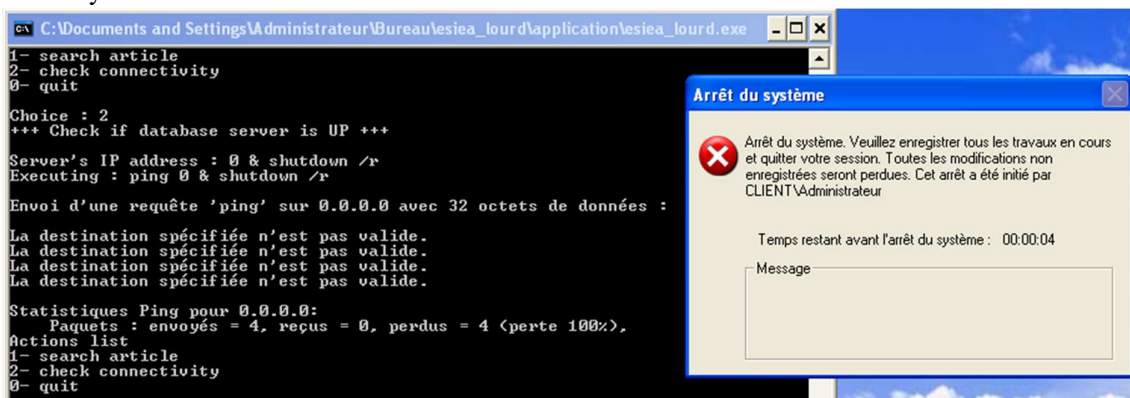
- a) Business impact: An attacker can use this vulnerability to acquire total control of the system. The business impact of this vulnerability is high.
- b) Exploitability: This vulnerability while not obvious requires minimal effort to exploit.
- c) Remediation: Prevent users from running system commands on the system. This does not require an increase in budget or manpower or new technologies.

Description:

This vulnerability allows the user to run system commands directly on the system. It is possible using this vulnerability alone to take total control of the system and run any application on it.

Exploitation:

- a) After logging in, choose option 2 which runs a ping command on the inputted IP address. By adding additional system commands after the IP address, the attacker can run any application on the system.



By adding the & shutdown /r causes the system to restart

Remediation:

Remove the option to run system commands from within the application. This will prevent arbitrary system command injection.

Reference:

<https://portswigger.net/web-security/os-command-injection>

Vulnerability 11: Sessions never expire.

Criticality: Low

- a) Business impact: Sessions that do not expire mean that a logged in user will remain logged in until they explicitly log out. This could provide an attacker with access if a user forgets to log out and in turn the attacker could damage the business.
- b) Exploitability: This vulnerability is exploitable only with access to a system that is left logged in. This makes the vulnerability unlikely to be exploited.
- c) Remediation: Add session expiration as a feature so that each user session expires after a certain duration of inactivity. This will require some manpower to implement.

Description:

This vulnerability can be exploited by an attacker that has access to a logged in system. As the user sessions never expire, the user will remain logged in even if they left the system without logging out days ago. Such a system could be exploited by an attacker.

Exploitation:

- a) A user forgets to log out and leaves the system. The system remains logged in several hours later as there is no feature to log them out.
- b) An attacker happens by the logged in system and begins to access all the data in the database as additional authentication is not required in a log in system.

Remediation:

Add a session expiration feature that automatically logs users out after a pre-defined duration of inactivity.

Reference:

https://docstore.mik.ua/orelly/webprog/webdb/ch08_05.htm

Vulnerability 12: Credentials remain in memory after use.

Criticality: Medium

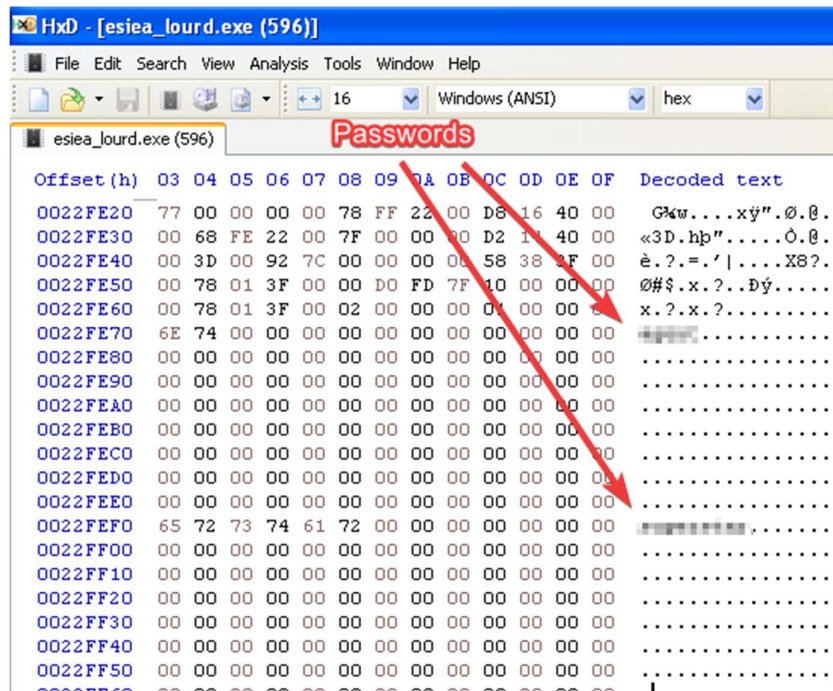
- a) Business impact: Sessions that do not expire mean that a logged in user will remain logged in until they explicitly log out. This could provide an attacker with access if a user forgets to log out and in turn the attacker could damage the business.
- b) Exploitability: This vulnerability is exploitable only with access to a system that is left logged in. This makes the vulnerability unlikely to be exploited.
- c) Remediation: Add session expiration as a feature so that each user session expires after a certain duration of inactivity. This will require some manpower to implement.

Description:

This vulnerability can be exploited by an attacker that has access to a logged in system. As the user sessions never expire, the user will remain logged in even if they left the system without logging out days ago. Such a system could be exploited by an attacker.

Exploitation:

- a) After logging into the application, use a tool such as HxD to read the memory occupied by this application.



Passwords are visible in the memory occupied by the application

Remediation:

Implement a purge of passwords from memory once authentication is completed. Leaving the passwords in memory is a security risk as attackers could scan the memory and find the passwords.

Reference:

<https://www.sjoerdlangkemper.nl/2016/05/22/should-passwords-be-cleared-from-memory/>

Vulnerability 13: Tampering of SQL queries.

Criticality: High

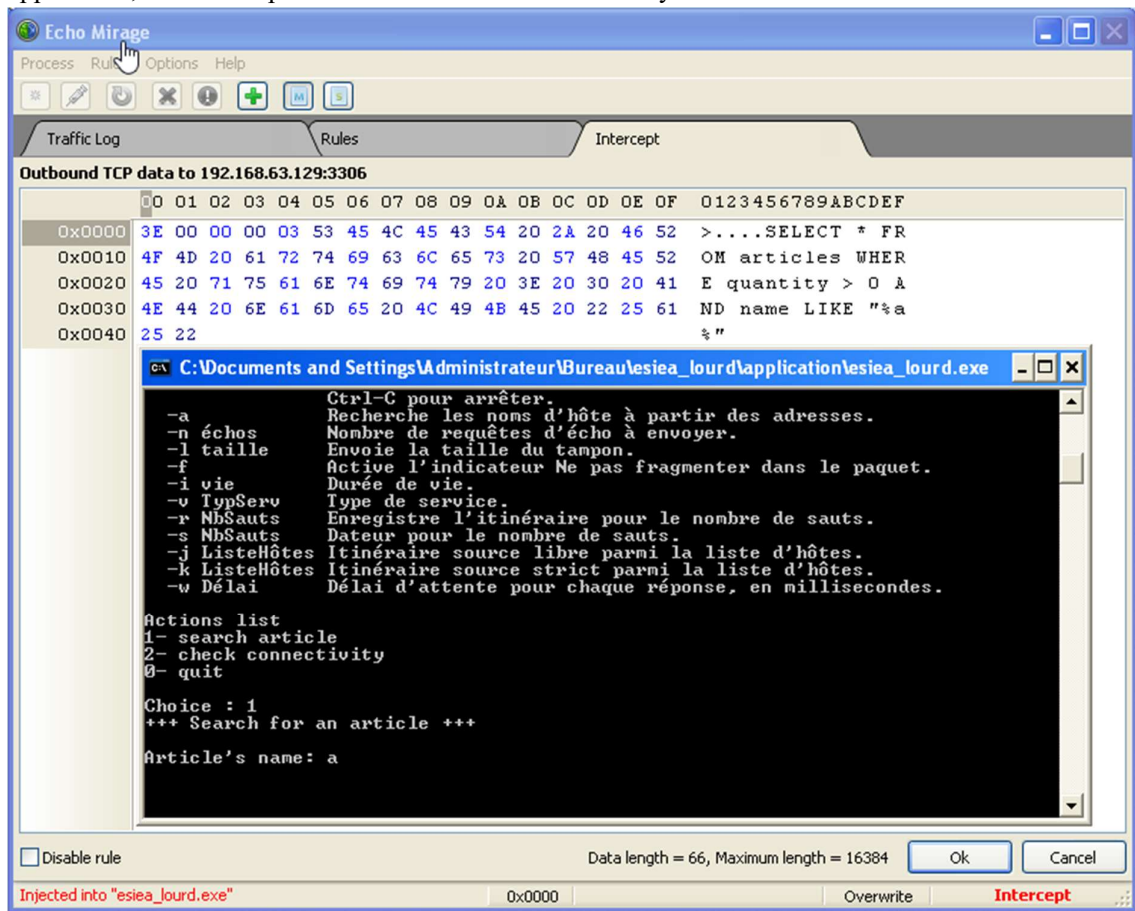
- a) Business impact: Tampering with SQL queries can give an access to all the information in the database. This could have a high business impact.
- b) Exploitability: This vulnerability requires particular skills and knowledge of SQL to exploit. It is not easily exploitable.
- c) Remediation: The vulnerability is an unavoidable in 2-tier systems. The only way to avoid this vulnerability is to switch to a 3-tier system or using software such as Citrix to simulate a 3-tier system. Additional manpower and budget is required.

Description:

This vulnerability can be exploited by an attacker that has access to a logged in system. As the user sessions never expire, the user will remain logged in even if they left the system without logging out days ago. Such a system could be exploited by an attacker.

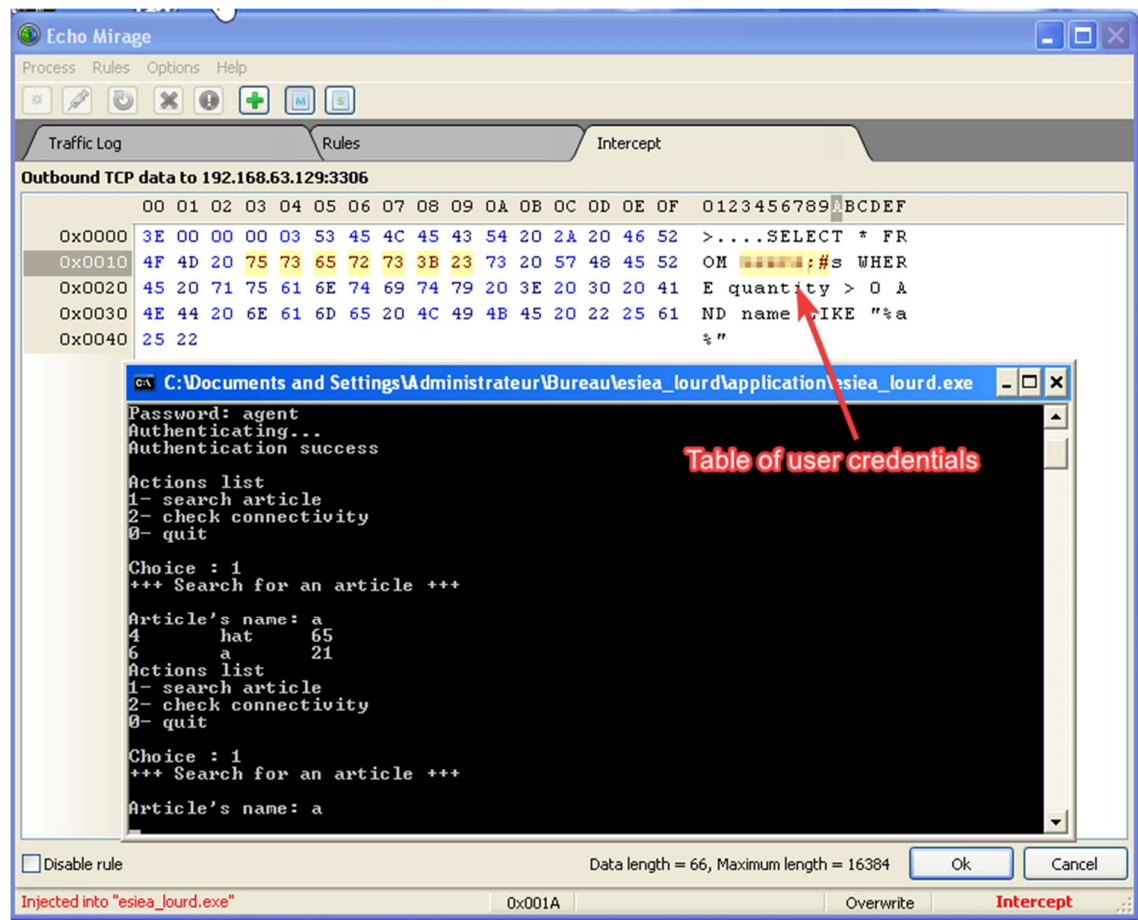
Exploitation:

- a) After logging into the application, open Echo Mirage and inject into the application. In the application, select the option to search articles and enter any text.



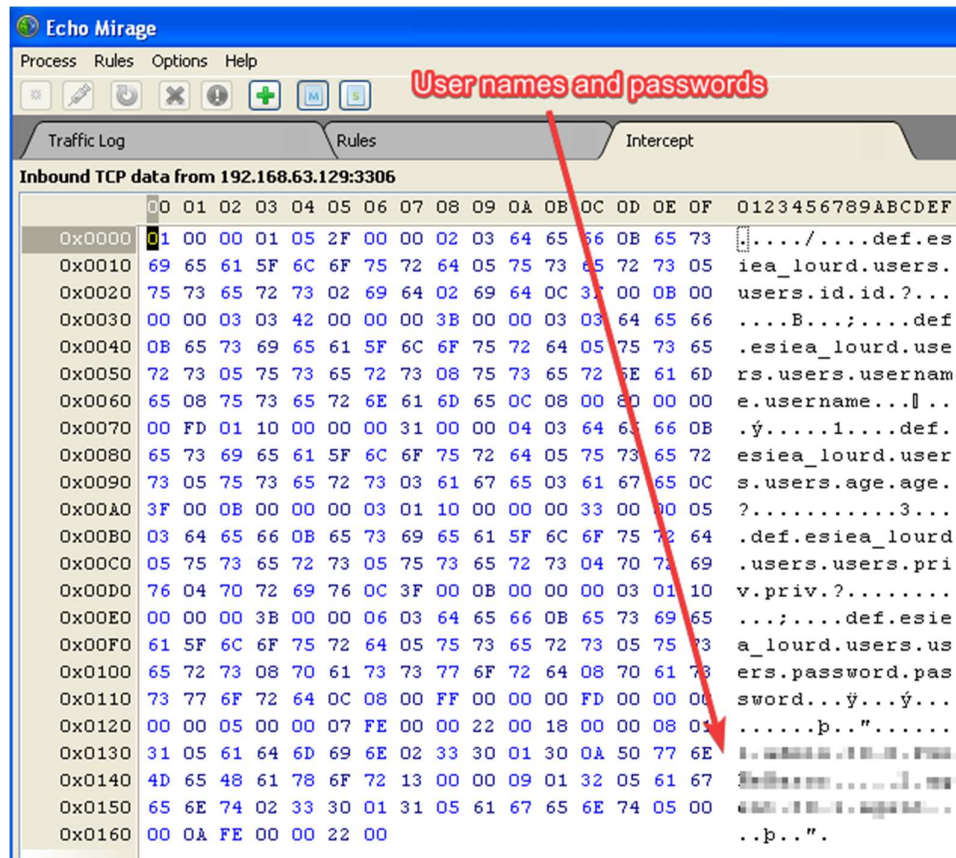
The SQL query is intercepted by Echo Mirage and ready to be altered

- b) Edit the SQL query in Echo Mirage to return the information you would like from the database.



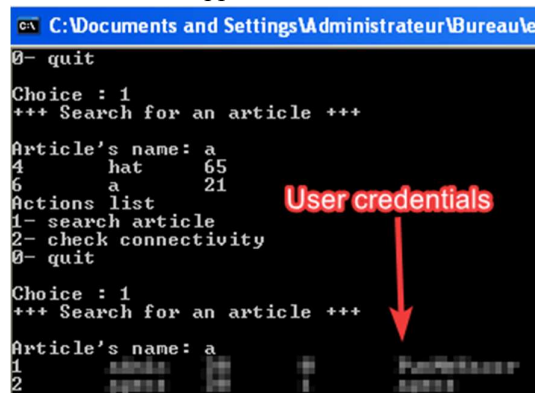
SQL query is modified in Echo Mirage to return user credentials

- c) The database returns user credentials and they're intercepted by Echo Mirage.



User credentials are returned from the database

- d) The user credentials are returned to the application.



The user credentials are returned the application instead of article information

Remediation:

Use of a 3-tier system or a simulation of one such as citrix is the only fix for this vulnerability. This is an inherent problem in all 2-tier systems and is a common vulnerability and therefore likely to be exploited by a motivated attacker.

Reference:

<https://portswigger.net/support/using-burp-to-detect-sql-injection-via-sql-specific-parameter-manipulation>