SOFTWARE AND DATABASE SECURITY AUDIT REPORT

ESIEA_LOURD THICK CLIENT

KUNAL PATIL

MSc Artificial Intelligence Systems

EPITA 2020

26/12/2020

Table of Contents

SYNTHESIS

This report aims to test the vulnerabilities in the Esiea_Lourd Thick Client application. Basically, this application is used for exploiting different possible vulnerabilities within application and related environment.

Severity of vulnerabilities can be classified as Critical, Medium and Low. (Refer NVD for more details) In this analysis there are total 11 vulnerabilities found among which 8 vulnerabilities can be considered as critical and 3 as medium.

The analysis done with consideration of the severity, business impact and scope of the vulnerability followed by its description in detail and detailed Exploitation.

After performing exploitation for each vulnerability, the remediation is mentioned to overcome the particular vulnerability and secure the system to prevent from attackers.

VULNERABILITIES

Vulnerability 1: Use of Hard-coded Credentials

Severity: Critical

a. **Business Impact**:
   If hard-coded passwords are used, it is almost certain that malicious users will gain access to the account in question.
   This weakness can lead to the exposure of resources or functionality to unintended actors, possibly providing attackers with sensitive information or even execute arbitrary code.
b. **Exploitability**:
   This vulnerability is so easy to exploit even without any specific tool.
   Someone without specific technical knowledge can exploit it simply by using notepad.
c. **Remediation**:
   Rather than hard-code a default username and password, key, or other authentication credentials for first time logins, utilize a "first login" mode that requires the user to enter a unique strong password or key.
   Apply strong one-way hashes to passwords and store those hashes in a configuration file or database with appropriate access control.

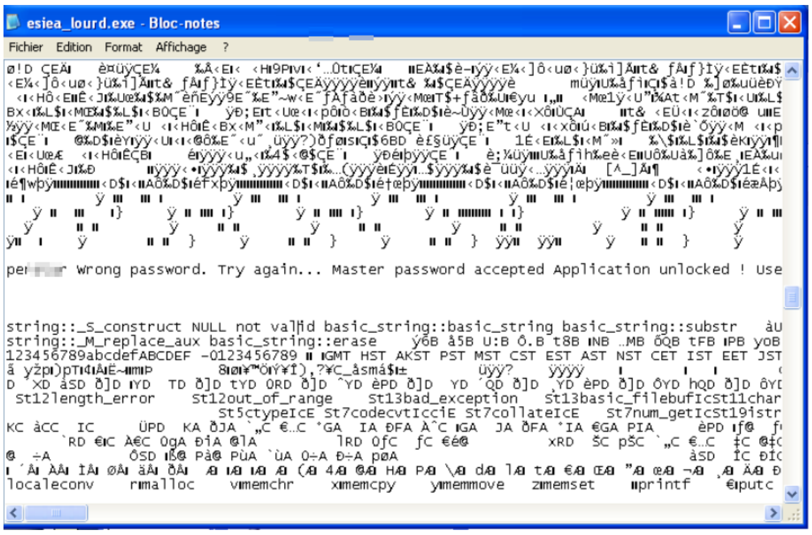
Description:

The software contains hard-coded credentials, such as a password or cryptographic key, which it uses for its own Inbound authentication. Hard-coded credentials typically create a significant hole that allows an attacker to bypass the authentication that has been configured by the software administrator.

Inbound Authentication: the software contains an authentication mechanism that checks the input credentials against a hard-coded set of credentials.

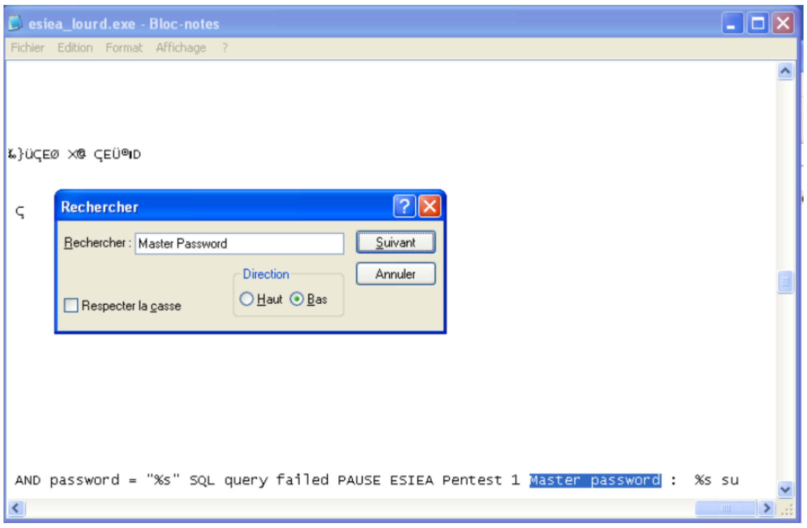
Exploitation:

a. Open esiea_lourd.exe in Notepad.


esiea_lourd.exe - Bloc-notes
Fichier Edition Format Affichage ?

```
per[redacted] Wrong password. Try again... Master password accepted Application unlocked ! Use


string::_S_construct NULL not valid basic_string::basic_string basic_string::substr
string::_M_replace_aux basic_string::erase
123456789abcdefABCDEF -0123456789   GMT HST AKST PST MST CST EST AST NST CET IST EET JST
 St12length_error    St12out_of_range    St13bad_exception   St13basic_filebufIcSt11char
                        St5ctypeIcE St7codecvtIcciE St7collateIcE   St7num_getIcSt19istr
localeconv    malloc    memchr    memcpy    memmove    memset    printf    putc
```

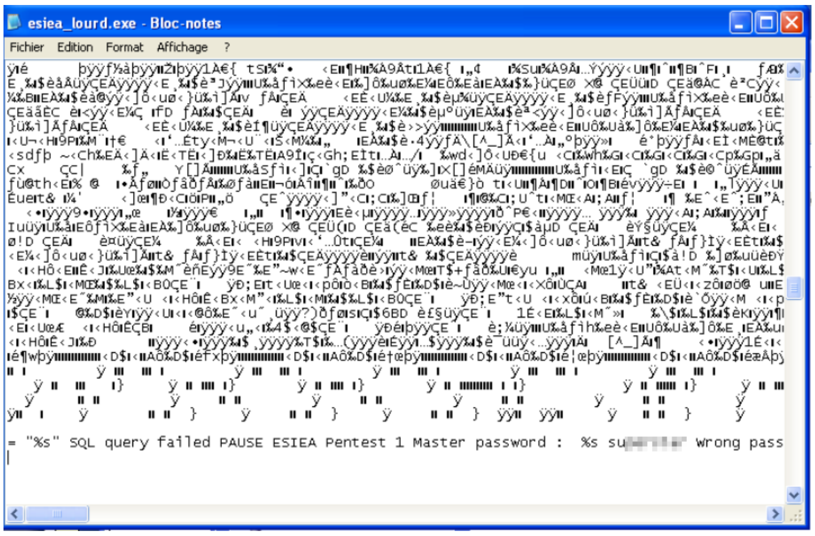
b. Search for keyword “Master Password” .


esiea_lourd.exe - Bloc-notes
Fichier Edition Format Affichage ?
Rechercher
Suivant
Annuler
Direction
Haut
Bas
Respecter la casse

Rechercher : Master Password

```
AND password = "%s" SQL query failed PAUSE ESIEA Pentest 1 Master password : %s su
```

c. Search for password in the same line.
The password is hardcoded and visible in plain-text


esiea_lourd.exe - Bloc-notes
Fichier Edition Format Affichage ?

```
= "%s" SQL query failed PAUSE ESIEA Pentest 1 Master password : %s su[redacted] Wrong pass
```

Remediation:

1. Consider using login authentication with strong passwords or keys instead of hardcoding them.
2. If hardcoded credentials cannot be removed, perform access control checks and limit accessed resources.
3. Consider using outbound authentication, where storing passwords and keys outside of the code, in mostly strongly-protected and encrypted storage which should be safe from outsiders as well as local users.

Reference:

1. https://cwe.mitre.org/data/definitions/798.html
2. https://www.appmarq.com/public/tqi,8222,CWE-798-Use-of-Hard-coded-Credentials

Vulnerability 2: Password changing feature not working

Severity: Medium

a. **Business Impact**:
   Gives the false indicator while updating an password for an user which actually upon restarting application sets back to older one which is hardcoded within the application.
b. **Exploitability**:
   This vulnerability is so easy to exploit even without any specific tool. Though it needs an clear observation and mindset of QA tester for an application as functionality.
c. **Remediation**:
   Implementation of updating database schema upon restarting the application. Including save changes for a column value for the password and updating it into database immediately and make it persistent forever.

Description:

This weakness can be categorized as a bug or missing feature for an application which may be caused during implementation of an architectural security tactic.

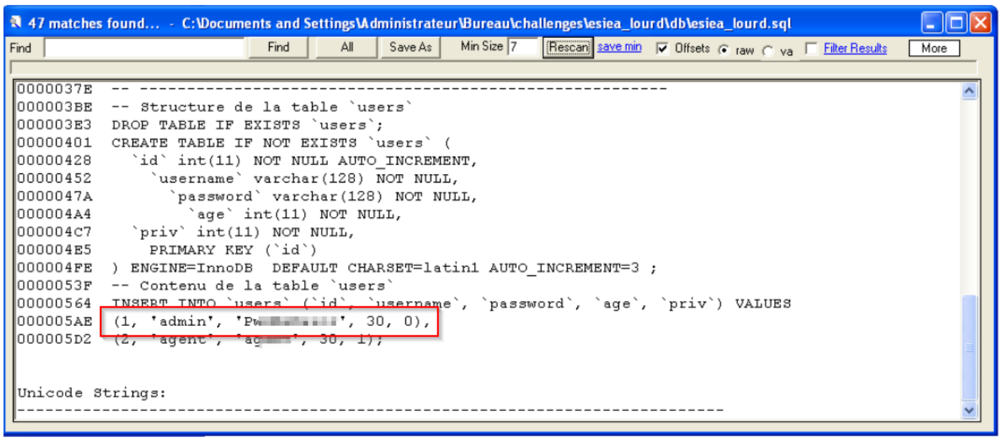
Exploitation:

a. Launch esiea_lourd.exe and enter Master password.

b. Login using admin credentials →

   To Find password for admin:

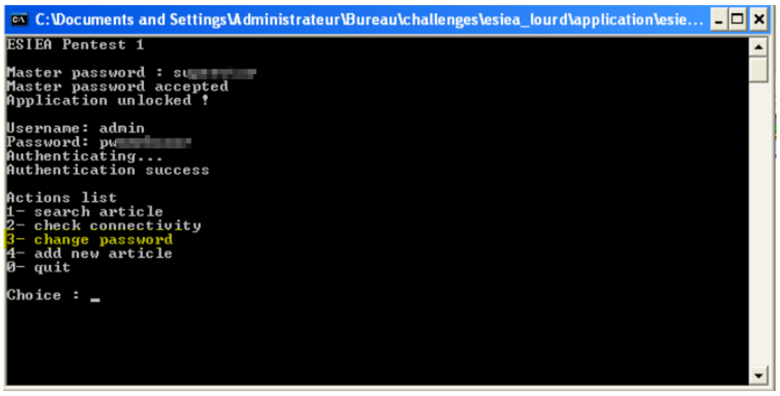
   Open application sql dump given with the application executable with strings.



```
0000037E  -- --------------------------------------------------------
000003BE  -- Structure de la table `users`
000003E3  DROP TABLE IF EXISTS `users`;
00000401  CREATE TABLE IF NOT EXISTS `users` (
00000428    `id` int(11) NOT NULL AUTO_INCREMENT,
00000452      `username` varchar(128) NOT NULL,
0000047A        `password` varchar(128) NOT NULL,
000004A4          `age` int(11) NOT NULL,
000004C7    `priv` int(11) NOT NULL,
000004E5      PRIMARY KEY (`id`)
000004FE  ) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=3 ;
0000053F  -- Contenu de la table `users`
00000564  INSERT INTO `users` (`id`, `username`, `password`, `age`, `priv`) VALUES
000005AE  (1, 'admin', 'Pw[redacted]', 30, 0),
000005D2  (2, 'agent', 'ag[redacted]', 30, 1);


Unicode Strings:
---------------------------------------------------------------------------
```

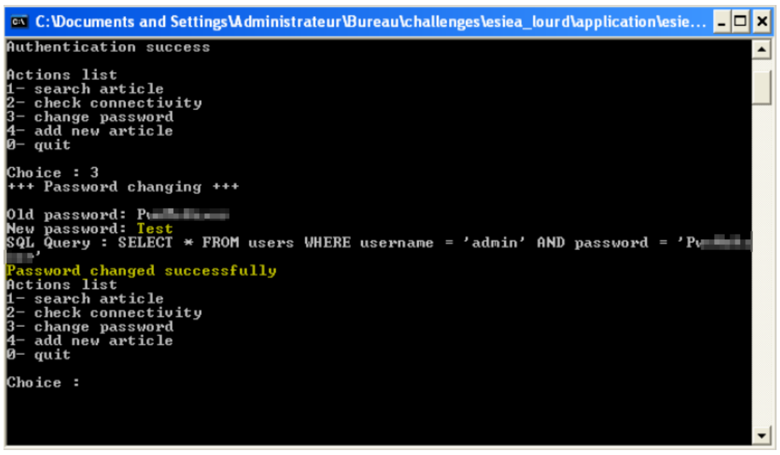
47 matches found... - C:\Documents and Settings\Administrateur\Bureau\challenges\esiea_lourd\db\esiea_lourd.sql
Find
Find
All
Save As
Min Size 7
Rescan
save min
Offsets
raw
va
Filter Results
More

   We can see the sql query written for the user admin, use the password for authentication.



```
C:\Documents and Settings\Administrateur\Bureau\challenges\esiea_lourd\application\esie...
ESIEA Pentest 1

Master password : su[redacted]
Master password accepted
Application unlocked !

Username: admin
Password: pw[redacted]
Authenticating...
Authentication success

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

Choice : _
```

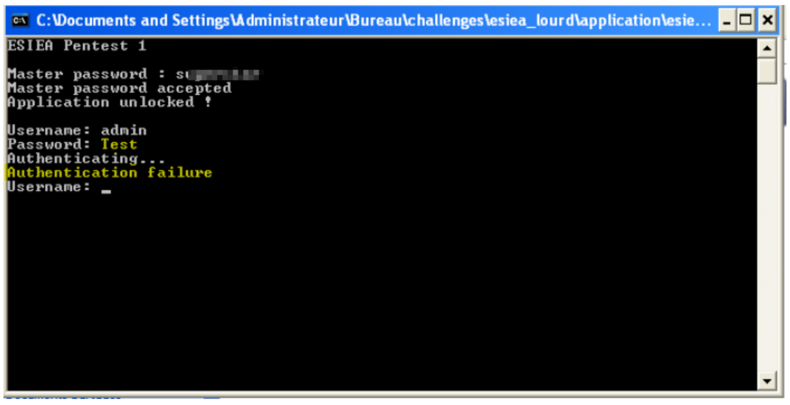
c. Select option 3 to change password.



```
C:\Documents and Settings\Administrateur\Bureau\challenges\esiea_lourd\application\esie...
Authentication success

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

Choice : 3
+++ Password changing +++

Old password: Pw[redacted]
New password: Test
SQL Query : SELECT * FROM users WHERE username = 'admin' AND password = 'Pw[redacted]
[redacted]'
Password changed successfully
Actions list
1- search article
2- check connectivity
3- change password
4- add new article
0- quit

Choice :
```

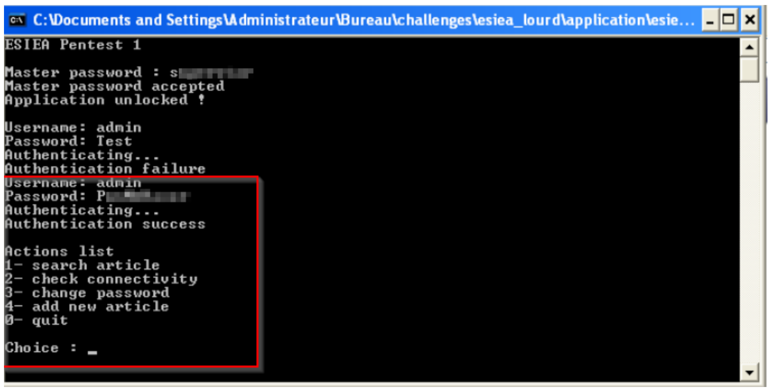
Upon changing password, the application gives the response “Password changed successfully” which gives the indication to user that old password has been overwritten.

d. Close the application and try again login with admin now with new password.



```
C:\Documents and Settings\Administrateur\Bureau\challenges\esiea_lourd\application\esie...
ESIEA Pentest 1

Master password : su[redacted]
Master password accepted
Application unlocked !

Username: admin
Password: Test
Authenticating...
Authentication failure
Username: _
```

e. Try login with old password.



```
C:\Documents and Settings\Administrateur\Bureau\challenges\esiea_lourd\application\esie...
ESIEA Pentest 1

Master password : s[redacted]
Master password accepted
Application unlocked !

Username: admin
Password: Test
Authenticating...
Authentication failure
Username: admin
Password: P[redacted]
Authenticating...
Authentication success

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

Choice : _
```

Login with old password is successful which shows that password is not updating after restarting the application.

Remediation:

1. Implementation of password change function is a sensitive function and requires protection policies within the application.
2. SQL query is hardcoded within the application which every time drop the 'users' table if present and creates a new one with the values hardcoded itself in the schema dump. This can be resolve by removing the drop query written in the dump to avoid deletion of users table upon restarting the application

Reference:

1. https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/04-Authentication_Testing/09-Testing_for_Weak_Password_Change_or_Reset_Functionalities
2. https://cwe.mitre.org/data/definitions/620.html

Vulnerability 3: Exposure of Sensitive Technical Information

Severity: Critical

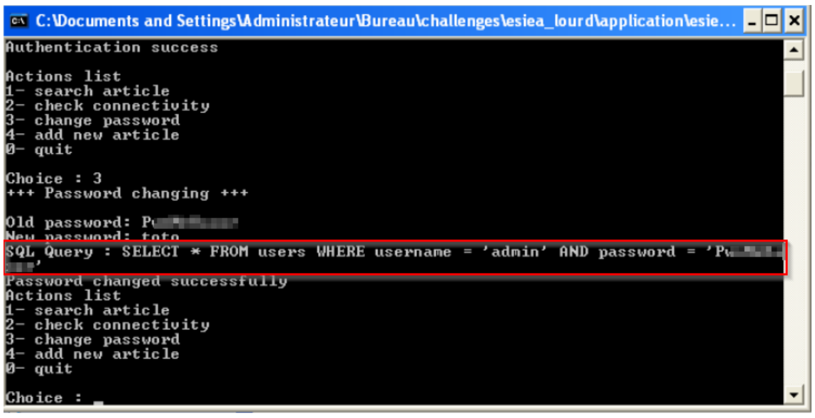
a. **Business Impact**:
   Scope: Loss of confidentiality.
   Impact: Reading the application data.
b. **Exploitability**:
   This vulnerability is so easy to exploit even without any specific tool. Upon changing the password for the user, it prints the SQL query which reveals the credentials for admin which is not a safe practice for implementation.
c. **Remediation**:
   Creation of safe area to define trust boundaries. Do not allow sensitive data to go out of the trust boundaries.
   Usually as a developer while debugging code often use the practice of print statement to analyse the code output. In such cases he/she ensures to remove print statements if any included while promoting the dev code to the production environment.

Description:

The product exposes sensitive information to an actor that is not explicitly authorized to have access to that information. This vulnerability exposes the product's own code or internal state.

Exploitation:

Try to change the password for admin.



```
C:\Documents and Settings\Administrateur\Bureau\challenges\esiea_lourd\application\esie...
Authentication success

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

Choice : 3
+++ Password changing +++

Old password: P[redacted]
New password: toto
SQL Query : SELECT * FROM users WHERE username = 'admin' AND password = 'P[redacted]'
Password changed successfully
Actions list
1- search article
2- check connectivity
3- change password
4- add new article
0- quit

Choice : _
```

Upon changing the password, SQL query return with the output which is revealing information about application to the attacker. As a attacker may approach for application structure and database to get more information and break the application.

Remediation:

Remove the return response as a SQL query as a output to the user within application.

Reference:

https://cwe.mitre.org/data/definitions/200.html

Vulnerability 4: Improper Handling of Case Sensitivity

Severity: Critical

a. **Business Impact**:
   Scope: Access control
   Impact: Bypass protection mechanism
b. **Exploitability**:
   This vulnerability is so easy to exploit even without any specific tool. Upon entering the password in any case, it allows user to login.
c. **Remediation**:
   Implement strict check input validation

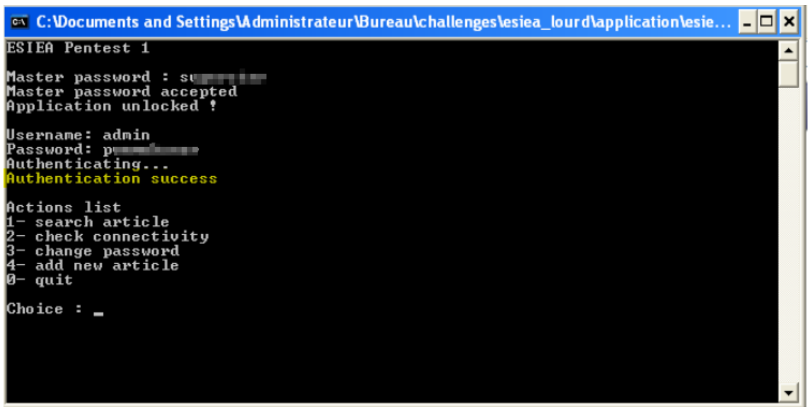
Description:

The software does not properly account for differences in case sensitivity when accessing or determining the properties of a resource, leading to inconsistent results.
Improperly handled case sensitive data can lead to several possible consequences, including:

- case-insensitive passwords reducing the size of the key space, making brute force attacks easier
- bypassing filters or access controls using alternate names
- multiple interpretation errors using alternate names.

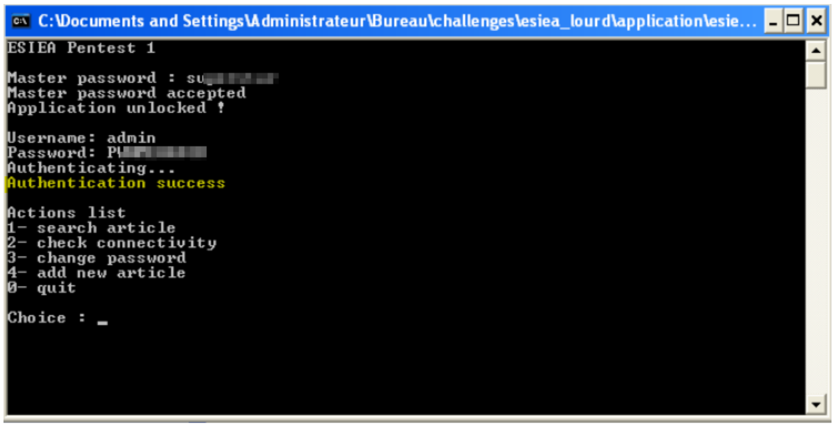
Exploitation:

a. Try to enter password for admin in minuscule.



```
C:\Documents and Settings\Administrateur\Bureau\challenges\esiea_lourd\application\esie...
ESIEA Pentest 1

Master password : su[redacted]
Master password accepted
Application unlocked !

Username: admin
Password: p[redacted]
Authenticating...
Authentication success

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

Choice : _
```

b. Try to enter password for admin in majuscule.



```
C:\Documents and Settings\Administrateur\Bureau\challenges\esiea_lourd\application\esie...
ESIEA Pentest 1

Master password : su[redacted]
Master password accepted
Application unlocked !

Username: admin
Password: P[redacted]
Authenticating...
Authentication success

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

Choice : _
```

In both cases, the authentication succeeds. Which means password accepted is case-insensitive.

Remediation:

1. Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a list of acceptable inputs that strictly conform to specifications.
2. Reject any input that does not strictly conform to specifications, or transform it into something that does, like case insensitive cases.
3. When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules.

Reference:

https://cwe.mitre.org/data/definitions/178.html

Vulnerability 5: Insufficiently Protected Credentials

Severity: Medium

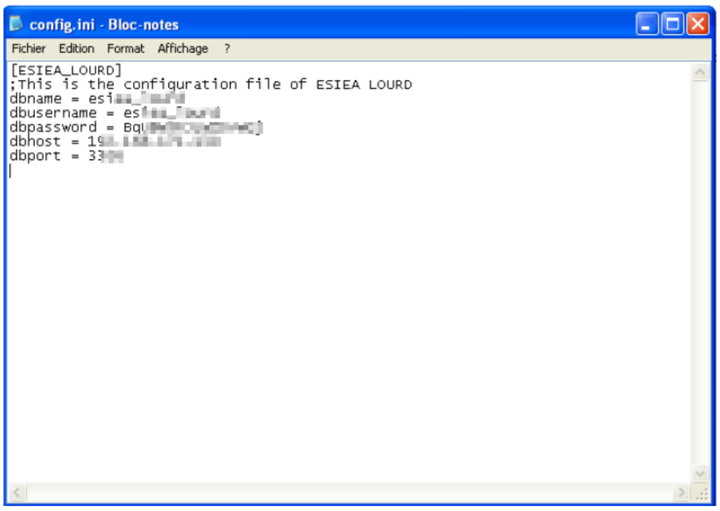
a. **Business Impact**:
   Scope: Access control
   Impact: Gain Privileges or Assume Identity
   An attacker could gain access to user accounts and access sensitive data used by the user accounts.
b. **Exploitability**:
   Supporting initializer file includes credentials in plaintext for connecting to the database.
   Attacker can use any widely available database clients, such as db visualizer.
c. **Remediation**:
   Implement appropriate security mechanism to protect the credentials.

Description:

The product transmits or stores authentication credentials, but it uses an insecure method that is susceptible to unauthorized interception and/or retrieval.

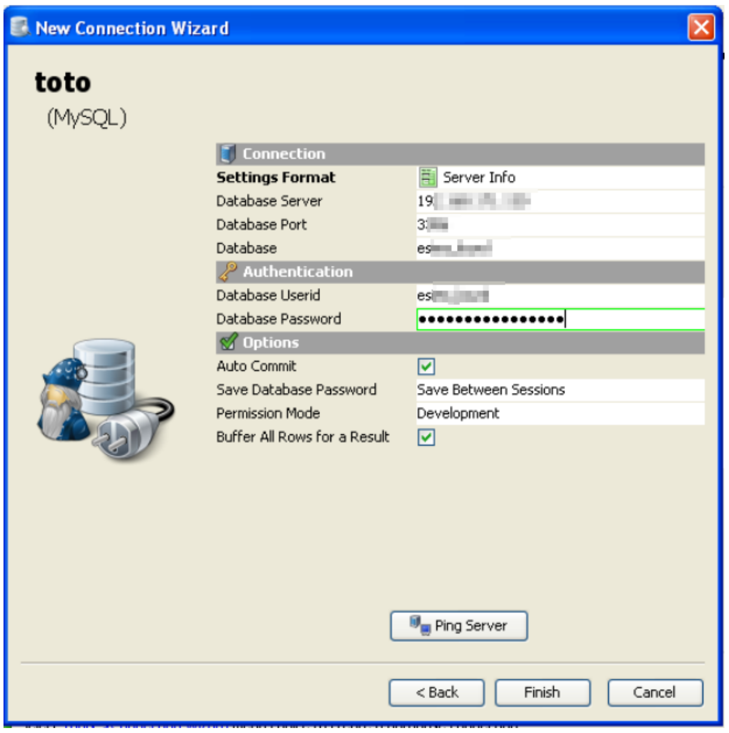
Exploitation:

a. Open config.ini file and observe the details



```
[ESIEA_LOURD]
;This is the configuration file of ESIEA LOURD
dbname = esi[redacted]
dbusername = es[redacted]
dbpassword = Bq[redacted]
dbhost = 19[redacted]
dbport = 33[redacted]
```

b. Check if these details can be use to connect to database and access the database of application.



New Connection Wizard

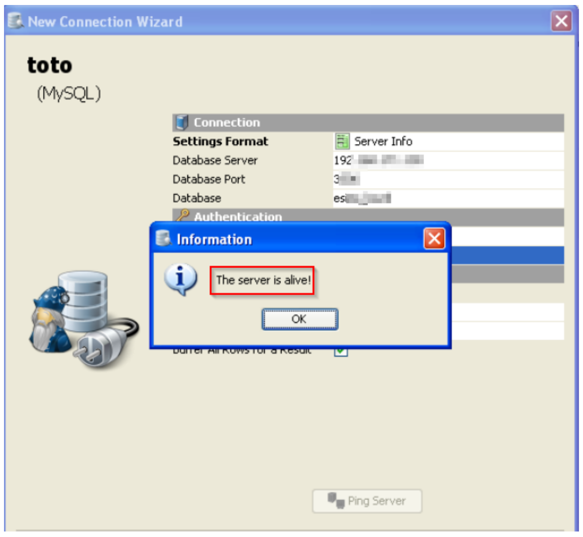
toto

(MySQL)

Connection	
Settings Format	Server Info
Database Server	19[redacted]
Database Port	3[redacted]
Database	es[redacted]
Authentication	
Database Userid	es[redacted]
Database Password	●●●●●●●●●●●●●●●●
Options	
Auto Commit	✓
Save Database Password	Save Between Sessions
Permission Mode	Development
Buffer All Rows for a Result	✓

Here by observing port number we can verify which is this database, It is MySql instance here.

c. Try ping to server.



New Connection Wizard

toto

(MySQL)

Connection	
Settings Format	Server Info
Database Server	192[redacted]
Database Port	3[redacted]
Database	es[redacted]
Authentication	

Information

The server is alive!

The ping is successful that means this file includes all the data necessary to connect to database used and access the content.

Remediation:

1. Use an appropriate security mechanism to protect the credentials.
2. Do not store credentials in plaintext in some file like .ini, .config, .txt etc.
3. Make appropriate use of cryptography to protect the credentials.
4. Use industry standards to protect the credentials.

Reference:

1. https://nvd.nist.gov/vuln/detail/CVE-2018-11079
2. https://cwe.mitre.org/data/definitions/522.html

Vulnerability 6: Cleartext Storage of Sensitive Information

Severity: Critical

a. **Business Impact**:
   Scope: Confidentiality
   Impact: Read application data
   An attacker with access to the system could read sensitive information stored in cleartext.

b. **Exploitability**:
   Using database visualizer tools can give a structure of database used by the application given we have all access credentials

c. **Remediation**:
   Implementation of appropriate security mechanism to protect the data.
   Use of encryption or use of techniques of cryptography.

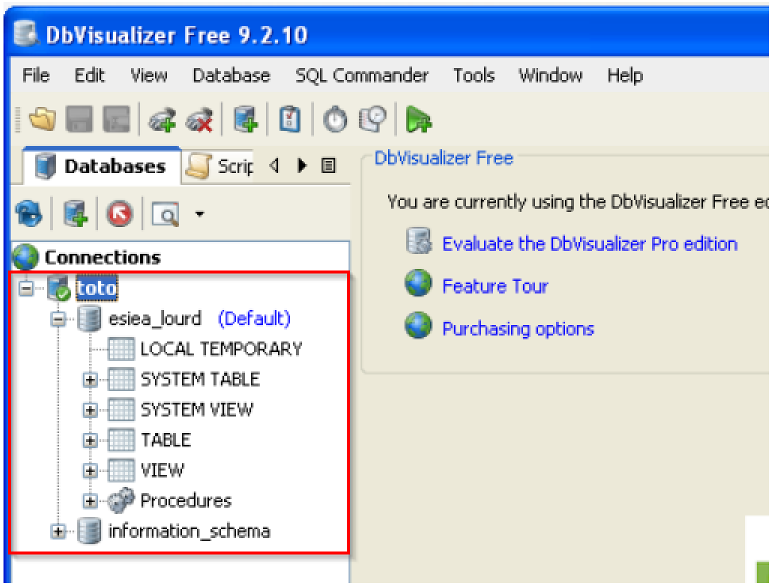
Description:

The application stores sensitive information in cleartext within a resource that might be accessible to another control sphere.

Because the information is stored in cleartext, attackers could potentially read it. Even if the information is encoded in a way that is not human-readable, certain techniques could determine which encoding is being used, then decode the information.

Exploitation:

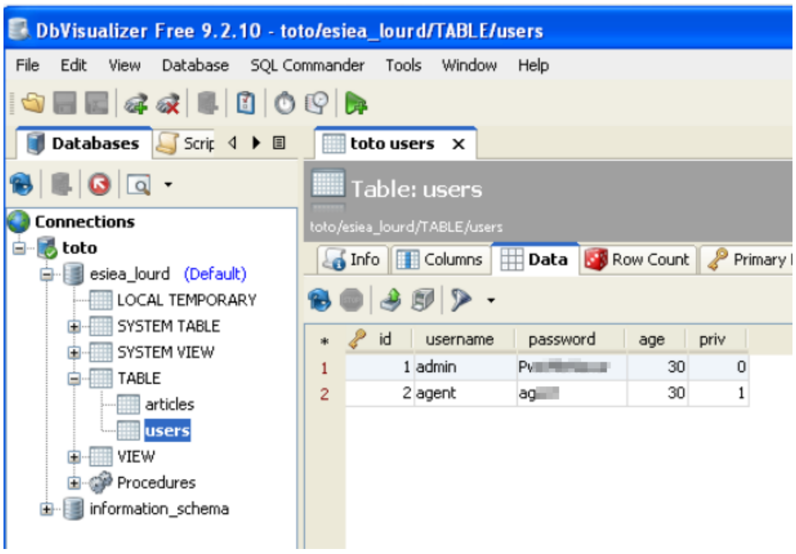
a. Connect to database using db visualiser



You are currently using the DbVisualizer Free ed

After connecting to the database, we can see the schema and the tables.

b. Check tables and information included.



	id	username	password	age	priv
1	1	admin	Pv[redacted]	30	0
2	2	agent	ag[redacted]	30	1

In users table, credentials are stored in plaintext.

Remediation:

1. Use an appropriate security mechanism to protect the credentials.
2. Using hashing techniques.
3. Implement encryption algorithms and ensure up-to-date and strong standard algorithms, protocols, and keys are in place; use proper key management.
4. Use Bcrypt
5. Set a reasonable work factor for the system.
6. Use a salt (modern algorithms do this automatically).
7. Consider using a pepper to provide an additional layer of security.

Reference:

1. https://cwe.mitre.org/data/definitions/312.html
2. https://nvd.nist.gov/vuln/detail/CVE-2009-0964
3. https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure
4. https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html

Vulnerability 7: Weak Password Requirements

Severity: Medium

a. **Business Impact**:

Scope: Access control

Impact: Gain Privileges or Assume Identity

An attacker could easily guess user passwords and gain access user accounts.

b. **Exploitability**:

Using simple common password such as test, agent, admin, toto, tata, etc which are easy to guess and most commonly used during dev phase of an application. This vulnerability can be simply exposed with opening strings for sql dump given with an application or by using db visualiser.

c. **Remediation**:

Recommended to use a strong password policy.

Avoid using test passwords and even if used be ensure to remove them while promoting an application from dev environment to production.

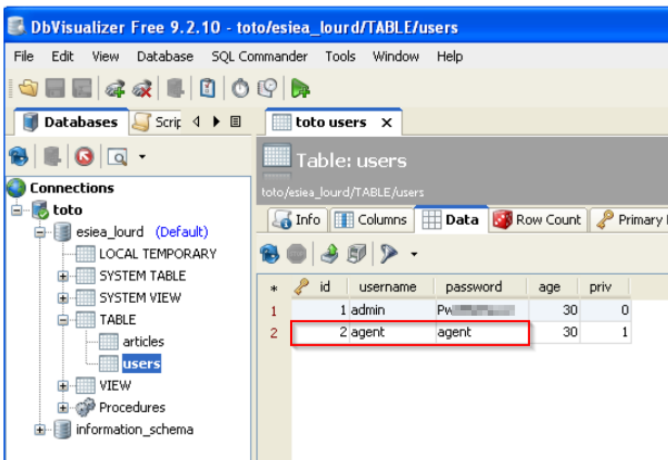
Description:

The product does not require that users should have strong passwords, which makes it easier for attackers to compromise user accounts.

Authentication mechanisms often rely on a memorized secret (also known as a password) to provide an assertion of identity for a user of a system. It is therefore important that this password be of sufficient complexity and impractical for an adversary to guess. The specific requirements around how complex a password needs to be depended on the type of system being protected. Selecting the correct password requirements and enforcing them through implementation are critical to the overall success of the authentication mechanism.

Exploitation:

a. Open database schema using db visualiser.



	id	username	password	age	priv
1	1	admin	Pw[redacted]	30	0
2	2	agent	agent	30	1

Here credentials agent/agent which is easy to guess by an attacker and shows implementation of weak password policies.

Remediation:

1. A product's design should require adherence to an appropriate password policy. Specific password requirements depend strongly on contextual factors, but it is recommended to contain the following attributes:
   a. Enforcement of a minimum and maximum length
   b. Restrictions against password reuse
   c. Restrictions against using common passwords
   d. Restrictions against using contextual string in the password (e.g., user id, app name)
2. Complex passwords requiring mixed character sets (alpha, numeric, special, mixed case)
3. Large Minimum Length (encouraging passphrases instead of passwords)
4. Randomly Chosen Secrets:
   Generating a password for the user can help make sure that length and complexity requirements are met, and can result in secure passwords being used.
5. Password Expiration:
   Requiring a periodic password change can reduce the time window that an adversary has to crack a password, while also limiting the damage caused by password exposures at other locations.
6. Consider implementing a password complexity meter to inform users when a chosen password meets the required attributes

Reference:

https://cwe.mitre.org/data/definitions/521.html

Vulnerability 8: Cleartext Transmission of Sensitive Information

Severity: Critical

a. **Business Impact**:

Scope: Integrity and Confidentiality

Impact: Read application data, modify files or directories

Anyone can read the information by gaining access to the channel being used for communication or just by shoulder surfing

b. **Exploitability**:

While typing the password within the application, the password in visible on screen which gives an insecure way to type a password in public.

c. **Remediation**:

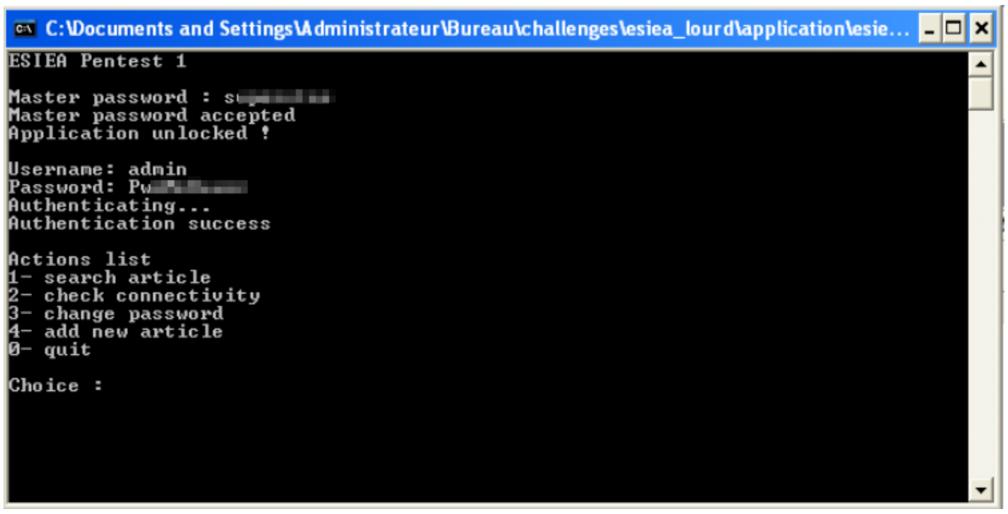
Preventing visibility of password text on application screen while login.

Description:

The software transmits sensitive or security-critical data in cleartext in a communication channel in public that can be sniffed by unauthorized actors.

Exploitation:

a. Login with credential within application.



C:\Documents and Settings\Administrateur\Bureau\challenges\esiea_lourd\application\esie...

```
ESIEA Pentest 1

Master password : s[redacted]
Master password accepted
Application unlocked !

Username: admin
Password: Pw[redacted]
Authenticating...
Authentication success

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

Choice :
```

The passwords are clearly visible on screen which can be easily sniffed by unauthorized users.

Remediation:

1. Encrypt the data with a reliable encryption scheme.
2. Improve the code by adding function tcsetattr() (implemented for unix terminals)

Reference:

1. https://cwe.mitre.org/data/definitions/319.html
2. https://security.stackexchange.com/questions/35133/masking-password-when-typing-vs-hiding-input-when-typing

Vulnerability 9: Missing Encryption of Sensitive Data

Severity: Critical

a. **Business Impact**:

Scope	Impact	Description
Confidentiality	Read application data	If the application does not use a secure channel, such as SSL, to exchange sensitive information, it is possible for an attacker with access to the network traffic to sniff packets from the connection and uncover the data. This attack is not technically difficult, but does require physical access to some portion of the network over which the sensitive data travels. This access is usually somewhere near where the user is connected to the network (such as a colleague on the company network) but can be anywhere along the path from the user to the end server.
Confidentiality **Integrity**	Modify application data	Omitting the use of encryption in any program which transfers data over a network of any kind should be considered on par with delivering the data sent to each user on the local networks of both the sender and receiver. Worse, this omission allows for the injection of data into a stream of communication between two parties -- with no means for the victims to separate valid data from invalid. In this day of widespread network attacks and password collection sniffers, it is an unnecessary risk to omit encryption from the design of any system which might benefit from it.

b. **Exploitability**:

This vulnerability can be exploited easily by having access to network.
Use of some sniffing tools like Wireshark

c. **Remediation**:

Use of appropriate encryption to avoid the human readability over the network

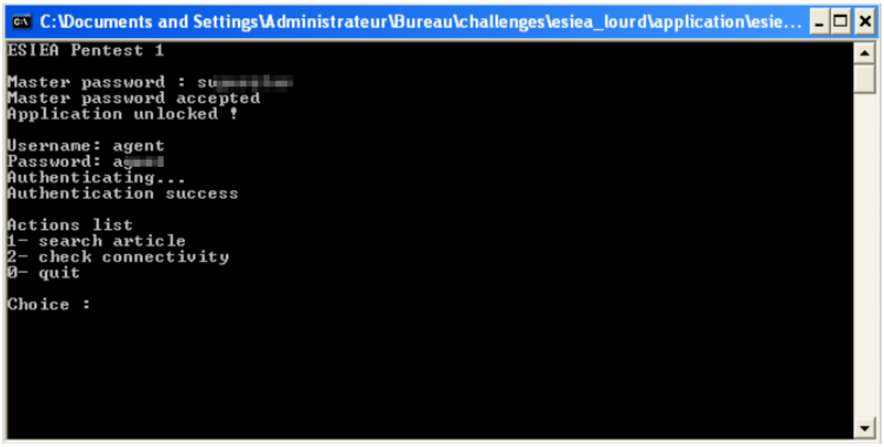
Description:

The software does not encrypt sensitive or critical information before storage or transmission.

The lack of proper data encryption passes up the guarantees of confidentiality, integrity, and accountability that properly implemented encryption conveys.

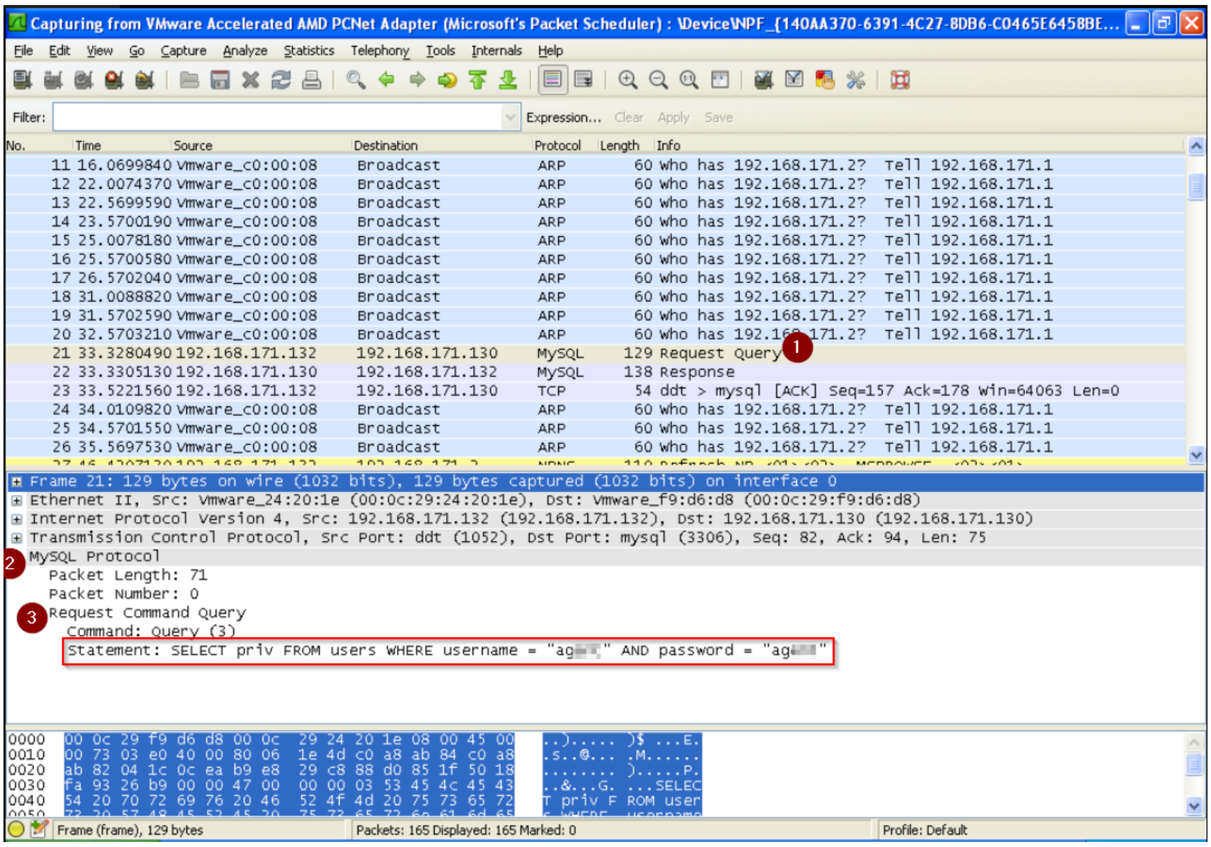
Exploitation:

a. Open wireshark and launch the application



```
C:\Documents and Settings\Administrateur\Bureau\challenges\esiea_lourd\application\esie...
ESIEA Pentest 1

Master password : su[redacted]
Master password accepted
Application unlocked !

Username: agent
Password: a[redacted]
Authenticating...
Authentication success

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

Choice :
```

b. Observe the network traffic in wireshark



Capturing from VMware Accelerated AMD PCNet Adapter (Microsoft's Packet Scheduler) : \Device\NPF_{140AA370-6391-4C27-8DB6-C0465E6458BE...

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
No.  Time        Source            Destination      Protocol Length Info
  11 16.0699840  Vmware_c0:00:08   Broadcast        ARP       60 Who has 192.168.171.2?  Tell 192.168.171.1
  12 22.0074370  Vmware_c0:00:08   Broadcast        ARP       60 Who has 192.168.171.2?  Tell 192.168.171.1
  13 22.5699590  Vmware_c0:00:08   Broadcast        ARP       60 Who has 192.168.171.2?  Tell 192.168.171.1
  14 23.5700190  Vmware_c0:00:08   Broadcast        ARP       60 Who has 192.168.171.2?  Tell 192.168.171.1
  15 25.0078180  Vmware_c0:00:08   Broadcast        ARP       60 Who has 192.168.171.2?  Tell 192.168.171.1
  16 25.5700580  Vmware_c0:00:08   Broadcast        ARP       60 Who has 192.168.171.2?  Tell 192.168.171.1
  17 26.5702040  Vmware_c0:00:08   Broadcast        ARP       60 Who has 192.168.171.2?  Tell 192.168.171.1
  18 31.0088820  Vmware_c0:00:08   Broadcast        ARP       60 Who has 192.168.171.2?  Tell 192.168.171.1
  19 31.5702590  Vmware_c0:00:08   Broadcast        ARP       60 Who has 192.168.171.2?  Tell 192.168.171.1
  20 32.5703210  Vmware_c0:00:08   Broadcast        ARP       60 Who has 192.168.171.2?  Tell 192.168.171.1
  21 33.3280490  192.168.171.132   192.168.171.130  MySQL    129 Request Query
  22 33.3305130  192.168.171.130   192.168.171.132  MySQL    138 Response
  23 33.5221560  192.168.171.132   192.168.171.130  TCP       54 ddt > mysql [ACK] Seq=157 Ack=178 Win=64063 Len=0
  24 34.0109820  Vmware_c0:00:08   Broadcast        ARP       60 Who has 192.168.171.2?  Tell 192.168.171.1
  25 34.5701550  Vmware_c0:00:08   Broadcast        ARP       60 Who has 192.168.171.2?  Tell 192.168.171.1
  26 35.5697530  Vmware_c0:00:08   Broadcast        ARP       60 Who has 192.168.171.2?  Tell 192.168.171.1

⊞ Frame 21: 129 bytes on wire (1032 bits), 129 bytes captured (1032 bits) on interface 0
⊞ Ethernet II, Src: Vmware_24:20:1e (00:0c:29:24:20:1e), Dst: Vmware_f9:d6:d8 (00:0c:29:f9:d6:d8)
⊞ Internet Protocol Version 4, Src: 192.168.171.132 (192.168.171.132), Dst: 192.168.171.130 (192.168.171.130)
⊞ Transmission Control Protocol, Src Port: ddt (1052), Dst Port: mysql (3306), Seq: 82, Ack: 94, Len: 75
  MySQL Protocol
    Packet Length: 71
    Packet Number: 0
    Request Command Query
      Command: Query (3)
      Statement: SELECT priv FROM users WHERE username = "ag[redacted]" AND password = "ag[redacted]"

0000  00 0c 29 f9 d6 d8 00 0c  29 24 20 1e 08 00 45 00   ..)..... )$ ...E.
0010  00 73 03 e0 40 00 80 06  1e 4d c0 a8 ab 84 c0 a8   .s..@... .M......
0020  ab 82 04 1c 0c ea b9 e8  29 c8 88 d0 85 1f 50 18   ........ ).....P.
0030  fa 93 26 b9 00 00 47 00  00 00 03 53 45 4c 45 43   ..&...G. ...SELEC
0040  54 20 70 72 69 76 20 46  52 4f 4d 20 75 73 65 72   T priv F ROM user

Frame (frame), 129 bytes    Packets: 165 Displayed: 165 Marked: 0    Profile: Default
```

The sql query is clearly visible in plaintext which is fetched from database the value once user login in application.

Remediation:

1. Clearly specify which data or resources are valuable enough that they should be protected by encryption. Require that any transmission or storage of this data/resource should use well-vetted encryption algorithms.
2. Ensure that encryption is properly integrated into the system design, including but not necessarily limited to:
    a. Encryption that is needed to store or transmit private data of the users of the system
    b. Encryption that is needed to protect the system itself from unauthorized disclosure or tampering
3. Identify the separate needs and contexts for encryption:
    a. One-way (i.e., only the user or recipient needs to have the key). This can be achieved using public key cryptography, or other techniques in which the encrypting party (i.e., the software) does not need to have access to a private key.
    b. Two-way (i.e., the encryption can be automatically performed on behalf of a user, but the key must be available so that the plaintext can be automatically recoverable by that user). This requires storage of the private key in a format that is recoverable only by the user (or perhaps by the operating system) in a way that cannot be recovered by others.
4. Attack surface reduction:
   Use naming conventions and strong types to make it easier to spot when sensitive data is being used. When creating structures, objects, or other complex entities, separate the sensitive and non-sensitive data as much as possible.

Reference:

https://cwe.mitre.org/data/definitions/311.html

Vulnerability 10: SQL Tampering

Severity: Critical

a. **Business Impact**:

Scope	Impact	Description
Confidentiality	Read Application data	Since SQL databases generally hold sensitive data, loss of confidentiality is a frequent problem with SQL injection vulnerabilities.
Access Control	Bypass protection mechanism	If poor SQL commands are used to check user names and passwords, it may be possible to connect to a system as another user with no previous knowledge of the password. If authorization information is held in a SQL database, it may be possible to change this information through the successful exploitation of a SQL injection vulnerability.
Integrity	Modify application data	Just as it may be possible to read sensitive information, it is also possible to make changes or even delete this information with a SQL injection attack.

b. **Exploitability**:

This vulnerability exploits the injection of the third-party application EchoMirage, with which we can inject the process of running application and get the packet details containing sql query. With EchoMirage, we can access/modify/delete the data within the database without having knowledge of database credentials.

c. **Remediation**:

Possible use of encryption of data coming from database.
Avoid leakage of db queries in plaintext within application backend.

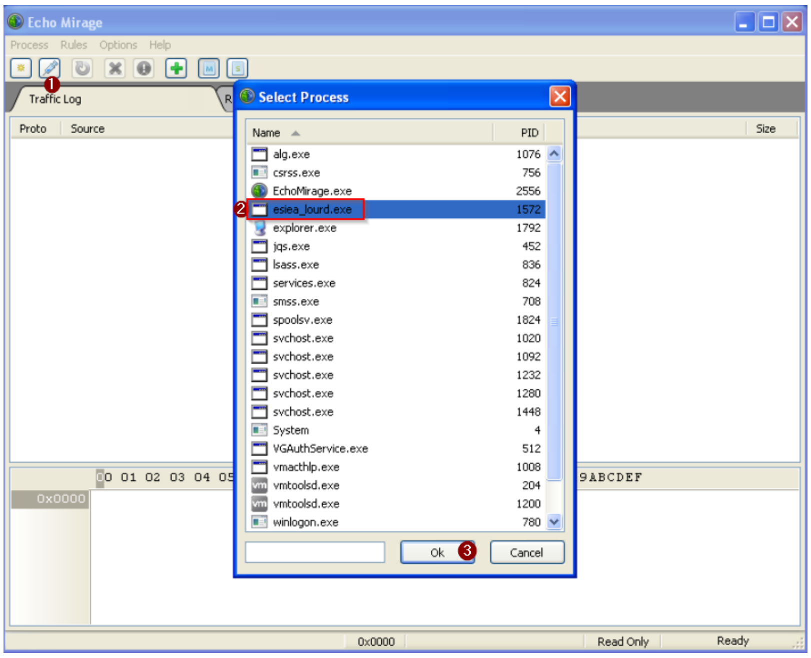
Description:

The software 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.

Without sufficient removal or quoting of SQL syntax in user-controllable inputs, the generated SQL query can cause those inputs to be interpreted as SQL instead of ordinary user data. This can be used to alter query logic to bypass security checks, or to insert additional statements that modify the back-end database, possibly including execution of system commands.

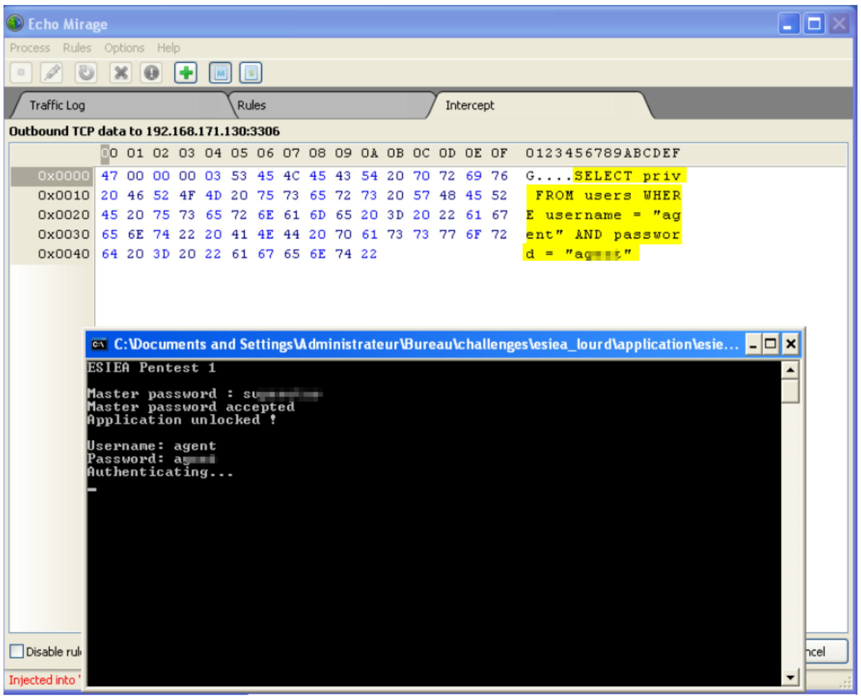
Exploitation:

a. Run application and then open Echo Mirage and inject the application process.



Name	PID
alg.exe	1076
csrss.exe	756
EchoMirage.exe	2556
esiea_lourd.exe	1572
explorer.exe	1792
jqs.exe	452
lsass.exe	836
services.exe	824
smss.exe	708
spoolsv.exe	1824
svchost.exe	1020
svchost.exe	1092
svchost.exe	1232
svchost.exe	1280
svchost.exe	1448
System	4
VGAuthService.exe	512
vmacthlp.exe	1008
vmtoolsd.exe	204
vmtoolsd.exe	1200
winlogon.exe	780

b. Try Login with any user in application and observe Echo Mirage



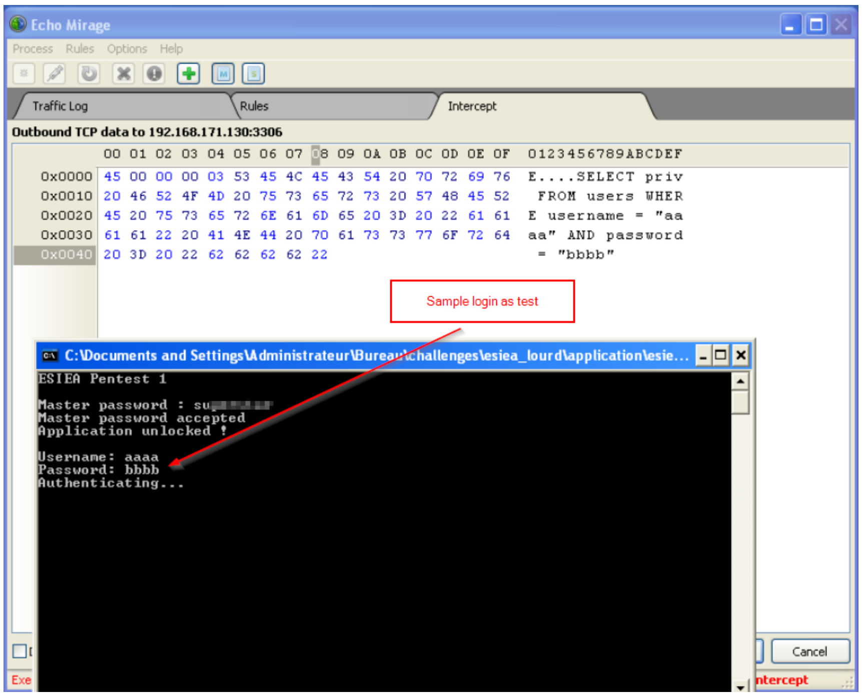
Outbound TCP data to 192.168.171.130:3306

```
       00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0x0000 47 00 00 00 03 53 45 4C 45 43 54 20 70 72 69 76  G....SELECT priv
0x0010 20 46 52 4F 4D 20 75 73 65 72 73 20 57 48 45 52   FROM users WHER
0x0020 45 20 75 73 65 72 6E 61 6D 65 20 3D 20 22 61 67  E username = "ag
0x0030 65 6E 74 22 20 41 4E 44 20 70 61 73 73 77 6F 72  ent" AND passwor
0x0040 64 20 3D 20 22 61 67 65 6E 74 22                 d = "ag[redacted]"
```

```
C:\Documents and Settings\Administrateur\Bureau\challenges\esiea_lourd\application\esie...
ESIEA Pentest 1

Master password : su[redacted]
Master password accepted
Application unlocked !

Username: agent
Password: a[redacted]
Authenticating...
_
```

We can clearly see SQL query which includes credentials that we entered.

c. Modification in query from Echo Mirage.



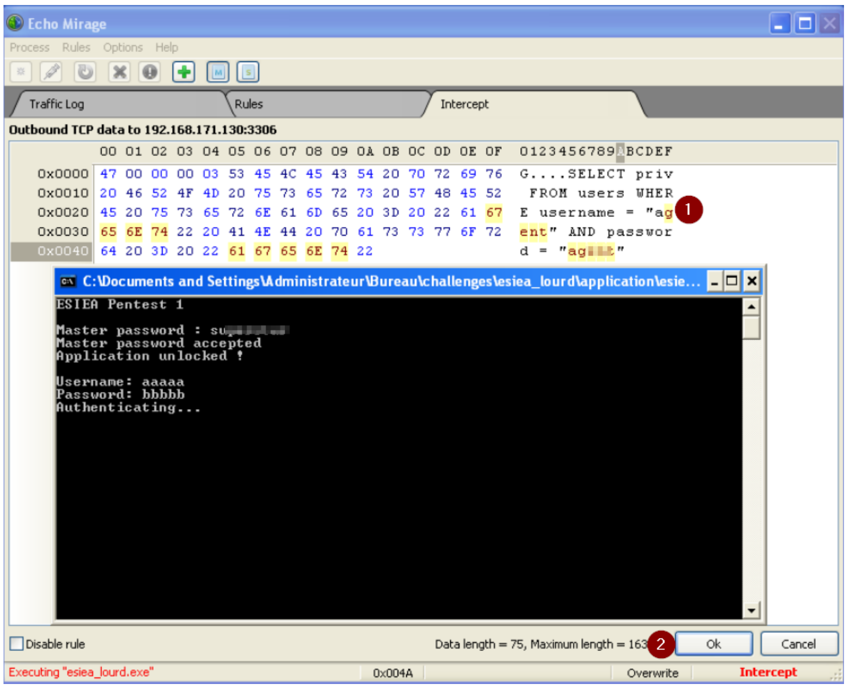
Outbound TCP data to 192.168.171.130:3306

```
        00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0x0000  45 00 00 00 03 53 45 4C 45 43 54 20 70 72 69 76  E....SELECT priv
0x0010  20 46 52 4F 4D 20 75 73 65 72 73 20 57 48 45 52   FROM users WHER
0x0020  45 20 75 73 65 72 6E 61 6D 65 20 3D 20 22 61 61  E username = "aa
0x0030  61 61 22 20 41 4E 44 20 70 61 73 73 77 6F 72 64  aa" AND password
0x0040  20 3D 20 22 62 62 62 62 22                        = "bbbb"
```

C:\Documents and Settings\Administrateur\Bureau\challenges\esiea_lourd\application\esie...

```
ESIEA Pentest 1

Master password : su[redacted]
Master password accepted
Application unlocked !

Username: aaaa
Password: bbbb
Authenticating...
```

d. Change credentials to agent with Echo Mirage



Outbound TCP data to 192.168.171.130:3306

```
        00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0x0000  47 00 00 00 03 53 45 4C 45 43 54 20 70 72 69 76  G....SELECT priv
0x0010  20 46 52 4F 4D 20 75 73 65 72 73 20 57 48 45 52   FROM users WHER
0x0020  45 20 75 73 65 72 6E 61 6D 65 20 3D 20 22 61 67  E username = "ag
0x0030  65 6E 74 22 20 41 4E 44 20 70 61 73 73 77 6F 72  ent" AND passwor
0x0040  64 20 3D 20 22 61 67 65 6E 74 22                 d = "ag[redacted]"
```

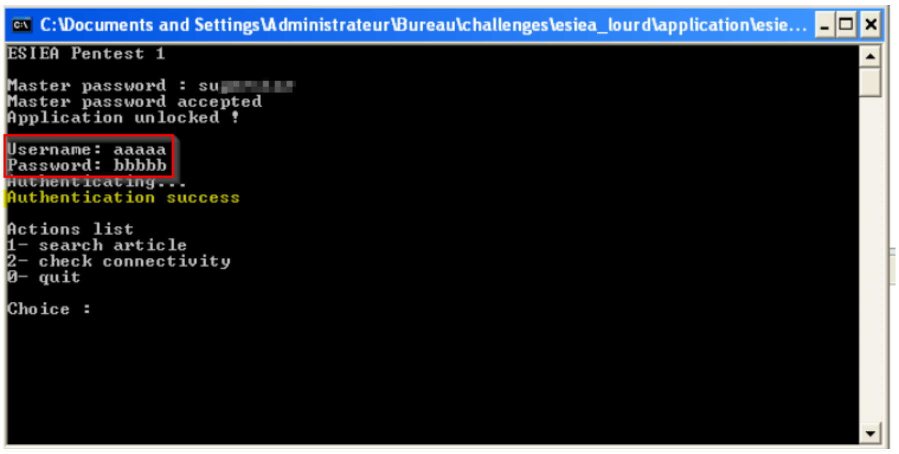
C:\Documents and Settings\Administrateur\Bureau\challenges\esiea_lourd\application\esie...

```
ESIEA Pentest 1

Master password : su[redacted]
Master password accepted
Application unlocked !

Username: aaaaa
Password: bbbbb
Authenticating...
```

Data length = 75, Maximum length = 163

Executing "esiea_lourd.exe"   0x004A

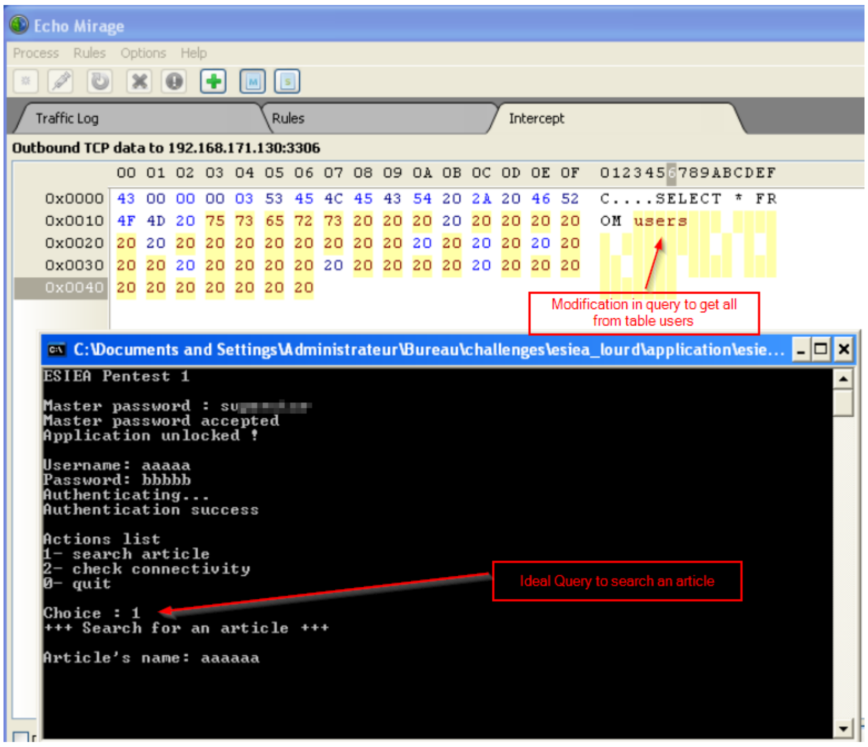
e. Click ok and see application



```
C:\Documents and Settings\Administrateur\Bureau\challenges\esiea_lourd\application\esie...
ESIEA Pentest 1

Master password : su[redacted]
Master password accepted
Application unlocked !

Username: aaaaa
Password: bbbbb
Authenticating...
Authentication success

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

Choice :
```

Here in application, it can see that even with false login credentials we are able to login because we modify the query with Echo Mirage.

f. Try to pass another query to fetch credentials from users table

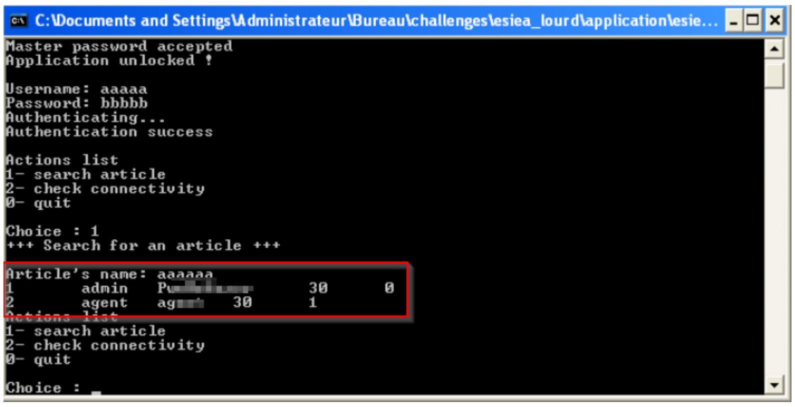


Echo Mirage
Process Rules Options Help
Traffic Log
Rules
Intercept
Modification in query to get all from table users
Ideal Query to search an article

```
Outbound TCP data to 192.168.171.130:3306
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F   0123456789ABCDEF
0x0000    43 00 00 00 03 53 45 4C 45 43 54 20 2A 20 46 52   C....SELECT * FR
0x0010    4F 4D 20 75 73 65 72 73 20 20 20 20 20 20 20 20   OM users
0x0020    20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0x0030    20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0x0040    20 20 20 20 20 20 20
```

```
C:\Documents and Settings\Administrateur\Bureau\challenges\esiea_lourd\application\esie...
ESIEA Pentest 1

Master password : su[redacted]
Master password accepted
Application unlocked !

Username: aaaaa
Password: bbbbb
Authenticating...
Authentication success

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

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

Article's name: aaaaaa
```

Click ok in Echo Mirage to execute the query and see the results in application console.



```
C:\Documents and Settings\Administrateur\Bureau\challenges\esiea_lourd\application\esie...
Master password accepted
Application unlocked !

Username: aaaaa
Password: bbbbb
Authenticating...
Authentication success

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

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

Article's name: aaaaaa
1       admin   Pu[redacted]        30      0
2       agent   ag[redacted]    30      1
Actions list
1- search article
2- check connectivity
0- quit

Choice : _
```

It returns the data from users table even through we login with agent credentials and now the attacker can get the credentials for admin as well.

<u>Remediation</u>:

1. Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. For example, consider using persistence layers such as Hibernate or Enterprise Java Beans, which can provide significant protection against SQL injection if used properly.
2. If available, use structured mechanisms that automatically enforce the separation between data and code.
   These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated.
3. While it is risky to use dynamically-generated query strings, code, or commands that mix control and data together, sometimes it may be unavoidable. Properly quote arguments and escape any special characters within those arguments. The most conservative approach is to escape or filter all characters that do not pass an extremely strict allowlist (such as everything that is not alphanumeric or white space). If some special characters are still needed, such as white space, wrap each argument in quotes after the escaping/filtering step. Be careful of argument injection (Refer more at CWE-88).
   Instead of building a new implementation, such features may be available in the database or programming language. For example, the Oracle DBMS_ASSERT package can check or enforce that parameters have certain properties that make them less vulnerable to SQL injection. For MySQL, the mysql_real_escape_string() API function is available in both C and PHP.
4. Consider Input validation strategy
   Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a list of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does.
5. Use an application firewall that can detect attacks against this weakness. It can be beneficial in cases in which the code cannot be fixed (because it is controlled by a third party), as an emergency prevention measure while more comprehensive software assurance measures are applied, or to provide defense in depth.
6. When using PHP, configure the application so that it does not use register_globals. During implementation, develop the application so that it does not rely on this feature, but be wary

of implementing a register_globals emulation that is subject to weaknesses such as CWE-95, CWE-621, and similar issues.

Reference:

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

Vulnerability 11: Improper Access Control

Severity: Critical

a. **Business Impact**:

Use of authorized functionality with other unauthorized users such as agent user

b. **Exploitability**:

Even if logged in with agent credentials, can access the options provided with admin login which are not visible in agent login session.

c. **Remediation**:

Very careful while managing the setting, management, and handling of privileges. Explicitly manage trust zones in the software.

Description:

The software does not restrict or incorrectly restricts access to a resource from an unauthorized actor.

Access control involves the use of several protection mechanisms such as:

1. Authentication (proving the identity of an actor)
2. Authorization (ensuring that a given actor can access a resource), and
3. Accountability (tracking of activities that were performed)

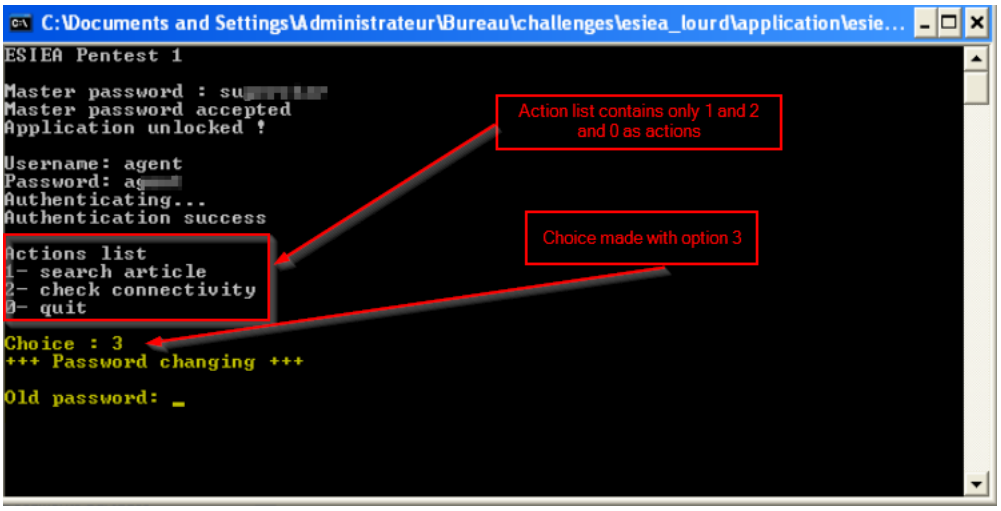
When any mechanism is not applied or otherwise fails, attackers can compromise the security of the software by gaining privileges, reading sensitive information, executing commands, evading detection, etc.

There are two distinct behaviours that can introduce access control weaknesses:

1. Specification: incorrect privileges, permissions, ownership, etc. are explicitly specified for either the user or the resource (for example, setting a password file to be world-writable, or giving administrator capabilities to a guest user). This action could be performed by the program or the administrator.
2. Enforcement: the mechanism contains errors that prevent it from properly enforcing the specified access control requirements (e.g., allowing the user to specify their own privileges, or allowing a syntactically-incorrect ACL to produce insecure settings). This problem occurs within the program itself, in that it does not actually enforce the intended security policy that the administrator specifies.

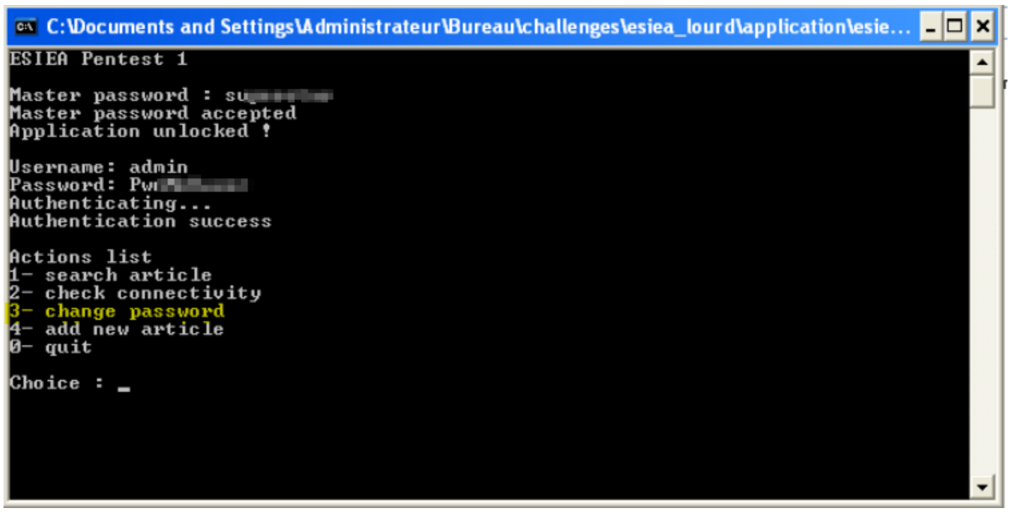
Exploitation:

a. Login with agent credentials and choose option 3



```
C:\Documents and Settings\Administrateur\Bureau\challenges\esiea_lourd\application\esie...
ESIEA Pentest 1

Master password : su[redacted]
Master password accepted
Application unlocked !

Username: agent
Password: ag[redacted]
Authenticating...
Authentication success

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

Choice : 3
+++ Password changing +++

Old password: _
```

Action list contains only 1 and 2 and 0 as actions
Choice made with option 3



```
C:\Documents and Settings\Administrateur\Bureau\challenges\esiea_lourd\application\esie...
ESIEA Pentest 1

Master password : su[redacted]
Master password accepted
Application unlocked !

Username: admin
Password: Pw[redacted]
Authenticating...
Authentication success

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

Choice : _
```

Even the option 3 to change password is privileged to admin and not for user agent, it is accessible.

Remediation:

1. Very careful while managing the setting, management, and handling of privileges. Explicitly manage trust zones in the software.
2. Compartmentalize the system to have "safe" areas where trust boundaries can be unambiguously drawn. Do not allow sensitive data to go outside of the trust boundary and always be careful when interfacing with a compartment outside of the safe area.
   Ensure that appropriate compartmentalization is built into the system design, and the compartmentalization allows for and reinforces privilege separation functionality. Architects and designers should rely on the principle of least privilege to decide the appropriate time to use privileges and the time to drop privileges.

Reference:

https://cwe.mitre.org/data/definitions/284.html