

Security Architecture for Source Code Protection



By

Noor Yasin

2011-NUST-MS-CCS-33

Supervisor

Dr. Abdul Ghafoor

School of Electrical Engineering and Computer Science
National University of Sciences and Technology
Islamabad, Pakistan.

July 2015

Approval

It is certified that the contents and form of the thesis entitled “**Security Architecture for Source Code Protection**” submitted by **Noor Yasin** have been found satisfactory for the requirement of the degree.

Advisor: **Dr. Abdul Ghafoor**

Signature: _____

Date: _____

Committee Member 1: Dr. Sead Muftic

Signature: _____

Date: _____

Committee Member 2: Dr. Adnan Khalid Kiani

Signature: _____

Date: _____

Committee Member 3: Ms. Rahat Masood

Signature: _____

Date: _____

Abstract

Software industry is the biggest industry nowadays. When software is distributed it is distributed in the form of executable files. The source code isn't distributed because software could easily be modified or the working logic could be understood. If the source code of some software gets stolen the software gets vulnerable. So, software source code security is the biggest concern for the software industry. Lack of security techniques to protect these source code and binaries might result into great financial or data loss. Unfortunately, the source code is saved on a local drive in plain form. So anyone getting access to the source code (which is in plain form) can launch on its own computer using the IDE.

We have given architecture for both single user and teaming environment to protect the source code and binaries. The source code and their binaries would be automatically protected in this architecture at backend and the developer won't need to worry about their protection. Any single code written would be automatically protected and only authorized and authentic users would have access to this source code. If somehow these source code and binaries are stolen they would be of no use because they are protected.

We have adopted the security techniques like encryption, decryption, authentication and authorization to protect the source code and their binaries

Certificate of Originality

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at National University of Sciences & Technology (NUST) School of Electrical Engineering & Computer Science (SEECS) or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged.

Author Name: Noor Yasin

Signature: _____

Acknowledgement

I am very thankful to Almighty Allah for giving me the intellect and strength to carry out this project. I would like to thanks' my supervisor **Dr. Abdul Ghafoor** because this thesis research would not have been possible without his continuous support.

Thanks to my class fellow **Muhammad Shoaib** for proof reading of this dissertation for any grammatical mistakes.

Also thankful to my committee members, **Dr. Sead Muftic, Dr. Adnan Khalid Kiani,** and **Ms. Rahat Masood,** for their guidance and support.

And finally, thanks to my family, and numerous friends who endured this long process with me, always offering support and love.

Dedicated

To

My loving Family

&

Supervisor

Table of Contents

1 Introduction	1
1.1 Introduction and Motivation.....	2
1.2 Problem Statement	3
1.3 Research Contributions	3
1.4 Thesis Organization.....	4
2 Literature Review	5
2.1 Attacks on Source code and Binaries	6
2.2 Existing Protection Methods	7
2.2.1 Obfuscation	7
2.3 Reverse Engineering of Binaries	9
2.4 Related Papers	10
3 Research Methodology	12
3.1 Research	13
3.1.1 Research Method & Research Methodology	13
3.1.2 Types of Research Methods	13
3.2 Thesis Research Approach	14
3.2.1 Defining Research Area	15
3.2.2 Literature Survey	16
3.2.3 Problem Statement	16
3.2.4 Hypothesis	16

3.2.5 Design Formulation.....	17
3.3.6 Implementation and Verification of Protocol.....	17
3.3.7 Evaluation and Confirmation	17
4 System Architecture & Design	18
4.1 Security Protocols	19
4.1.1 Security protocol for Single User Environment.....	19
4.1.1.1 Architectural Design of the System	19
4.1.2 Security Protocol for Teaming Environment	24
4.1.2.1 Detailed Architectural Design	24
5 Implementation.....	29
5.1 Getting the Source code of Eclipse IDE.....	30
5.1.1 Import as Plug-in.....	30
5.1.2 Complete Source code available on Eclipse Website	34
5.1.3 Git Repository	35
5.2 Plug-in Spy	36
5.3 Development	37
5.3.1 Running eclipse Source code	37
5.3.2 Setting Startup Environment	38
5.3.3 Required Plug-in and classes.....	39
5.3.4 Authentication Module.....	40
5.3.5 Encryption and Decryption Functions.....	41
5.3.6 Final Outcome	41
6 Conclusion & Future Directions	44
6.1 Conclusion.....	45

6.2 Future Directions	46
A Appendix Source Code (Java)	47
A.1 Authentication.....	47
A.2 Encryption.....	53
A.3 Decryption.....	56
A.4 Registration.....	62

List of Figures

2.1 Risk Factors Diagram	6
2.2 Simple Hello World Program	7
2.3 Obfuscated Hello World Program	8
2.4 Decompiled .class file using de-compiler	10
3.1 Thesis Research Workflow	15
4.1 Authentication Diagram	20
4.2 Authorization Diagram	21
4.3 Architecture diagram for Single user	22
4.4 Architecture diagram for Teaming Environment	26
5.1 Eclipse Import Wizard	30
5.2 Eclipse Import Plug-in and Fragmentation Diagram.....	31
5.3 Eclipse Importing Existing Plug-ins.....	31
5.4 Eclipse Importing Existing Plug-ins 2.....	32
5.5 Eclipse Source code	33
5.6 Eclipse 3.7.2 Build	34
5.7 Git Repository.....	35
5.8 Plug-in Spy	37
5.9 Compiling Eclipse Source code	38

5.10 Eclipse Startup Environment	39
5.11 Required Plug-ins	40
5.12 Login Screen	41
5.13 Source file in plain form	42
5.14 Binary file in plain form	42
5.15 Source file in Encrypted form	43
5.16 Binary file in Encrypted form	43

Chapter 1

Introduction

“The more knowledge you have, the greater will be your fear of Allah.”

-Abu Bakar (R.A)

This chapter presents the introduction and the motivational part of our thesis work. In this chapter we will discuss why this area was chosen for thesis research. We will present a summary of our thesis in this chapter that will give a complete picture of our thesis. We will define the problem statement that will be addressed and also discuss about our research contribution made in this area. And the last part will tell about the thesis organization.

1.1 Introduction and Motivation

Software industry is the largest industry nowadays and it's expanding day by day. Lots of new software's are introduced daily in market. The software that are commercial or clients specific are always distributed in the form of executables. The source code is never handed over to user to protect it from modification and stopping others to understand its working. Some software's are open source in which the source code is freely available and modifications are encouraged. But majority of the software's are available without their source code. Any source code can easily be launched in a compatible IDE. If the source code gets stolen, it might result in great data or financial losses. On the other hand, a binary file can be easily reverse engineered to get source code using some de-compiler like JAD [1] to introduce malicious code injection. Therefore, it is the requirements of software industry to provide a comprehensive solution for protection of source code from hackers.

From literature survey, we found that no solution was proposed to protect these source files and their binaries. Furthermore, a lot of work has been done to preserve the piracy of software but all of them provide these through licensing mechanism which is considered a weak solution. We came across a solution that was proposed to protect important files. The solution described a methodology using AES and MD5 for encrypting files [2]. The outcome file was hashed and strongly encrypted through the software. These files were useless if they got stolen. We got inspired from this solution and thought we would protect our source code and binaries the same way. Our main motivation was that the user should have minimum involvement and these files automatically get encrypted and decrypted at backend.

So we presented a secure architecture for protecting these source code and binaries in both single user and teaming environment. In our architecture we have adopted some security techniques like encryption, decryption, authentication and authorization that will provide automatic protection to these files. We also implemented our solution to validate our design. An IDE (Integrated development

environment) was to be chosen whose source code was easily available. IDE is a software application designed to facilitate software developers for developing software. It is basically a single program in which all development is done. It consists of source code editor, compiler, debugger and a GUI to facilitate developers. We chose Eclipse IDE to validate our design. Eclipse is an open source IDE with an extensible plug-in architecture to customize the environment [3]. A plug-in was designed that could be easily integrated with existing eclipse architecture to facilitate users by provide protection. The plug-in will authenticate user and authorize his access to source code and binaries. These source file and binaries would be automatically encrypted and decrypted at backend with user's key.

Research has been done to explore the major security issues to source code files and their binaries and providing a secure architecture for their protection. Due to great importance of these source code files this area of research was chosen.

1.2 Problem Statement

To provide a secure architecture for the protection of source code and their binaries during the development/building phase. A plug-in will be developed which will be automatically integrated with the IDE to provide source code protection. The source code and binaries would be automatically encrypted and decrypted at backend.

1.3 Research Contributions

This research will contribute in protecting the source code and their binaries. From research point of view a research publication has been made in ICCSIT 2014. The publication described the overall architecture of the protection mechanism. In detail the threats to source code and their binaries were discussed.

Same paper was accepted by ICCSIT to be published in a Journal named LNSE Vol. 4, No. 2, May 2016, pp. 153-156.

From practical aspect, we have implemented the proposed solution. In which the source code and binaries would be automatically protected. The solution is in a form of plug-in that will be easily distributed. The plug-in will be easily integrated with existing IDE architecture.

1.4 Thesis Organization

This thesis is organized in a systematic way to clearly state the research approach adopted and the contributions made in the thesis. Chapter 2 will describes the detailed literature survey related to this research. Chapter 3 describes the research methodology adopted during the thesis to achieve this solution. Different research methodologies like deductive research and conceptual research methods were combined to achieve the required results.

In chapter 4, we will discuss the detail design and architecture of our system. Chapter 5 will include the implementation of the system that was discussed in chapter 4. In chapter 6, conclusion of our thesis is given with some proposals for future extension of the thesis. In the end, references are given.

Chapter 2

Literature Review

“The only truly secure system is one that is powered off, cast in a block of concrete and sealed in a lead-lined room with armed guards.”

– Gene Spafford

This chapter presents the related work that has been done for the protection of source code and binaries. We will look into the details of all possible attacks that could be launched on source code and their binaries. We will also discuss how binaries can be reversed engineered to get source code if they aren't protected. We will present a summary of related papers. And at last we will present an analysis on conventional ways of protecting source code and binaries.

2.1 Attacks on Source code and Binaries

In Software Industry software source code security is of great concern. Lack of protection technique to protect the source files and their binaries might result into great data or financial loss. If somehow the source code or their binaries get compromised the whole application/product is compromised. Risk factors increase when the source code or their binaries get compromised.

Risk factors that are involved when the source code or binary files are compromised are finding loopholes in an application, adding unwanted features in an application, Steal idea of the application and make similar product, it could be used to bypass license checks, malicious code can be added in the application, Source and binaries can give information about loopholes in an application, unwanted feature can be added to the product and redistributed, Working Logic of application can be understood etc.

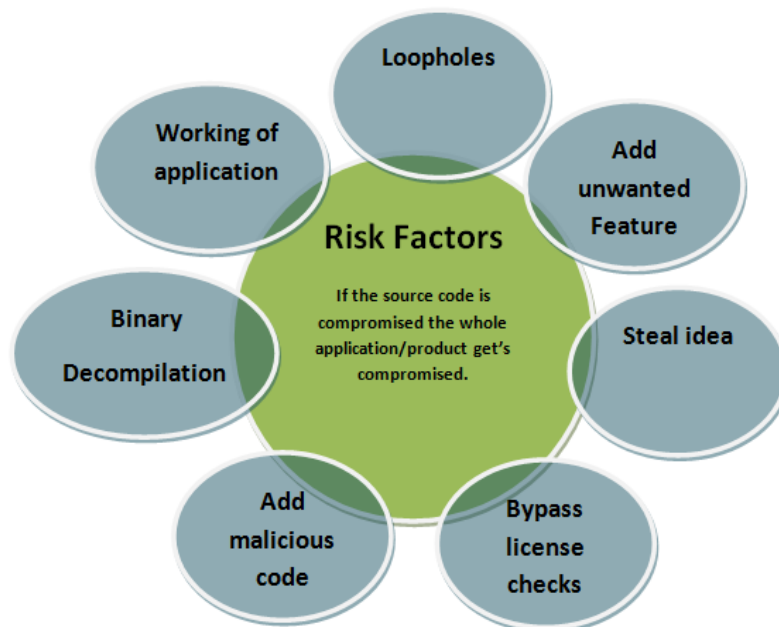


Figure 2.1: Risk Factors Diagram

2.2 Existing Protection Methods

2.2.1 Obfuscation

Obfuscation is a technique of transforming original source code that has original behavior but far more confusing, complex and harder to perceive or understand while preserving code semantics [4, 5]. This technique is used to prevent reverse engineering but providing same functionality. Obfuscating is replacing name and variables with meaningless ones that are harder to understand. Obfuscating removes all user defined comments. In Java obfuscation can be performed in two ways [6]:

- Source code obfuscation
- Byte Code Obfuscation

Source code obfuscation

Source code obfuscation involves in transforming the source code of an application to another source code that is similar in behavior but it's difficult to comprehend. In source code obfuscation you replace the names of identifiers with more ambiguous ones. A simple example of source code obfuscation is shown below.

```
public class HelloWorld {  
  
    public static void main(String args[]) {  
        System.out.println("Hello World!");  
    }  
}
```

Figure 2.2: Simple Hello World Program

When we obfuscate the above code using some tool the output would be something like this. The below function has the same output behavior but it's harder to understand and annoying.

```
public class HelloWorld {  
  
    public static void main(String args[]) {  
        double d1 = 0.0134654879927;  
        double d2 = 0.0234987519084;  
  
        for (int i1 = 0; i1 < 72; i1++) {  
            d1 = d2 + 0.00000001020102;  
        }  
  
        for (int i1 = 0; i1 < 59; i1++) {  
            d2 = d1 + 0.00000001120102;  
        }  
  
        //System.out.println(d1+d2);  
  
        if ((d1+d2) > 0.04699753441986 ) {  
            System.out.println("Hello World!");  
        }  
        else if ((d1+d2) < 0.04699753441186) {  
            System.out.println("Goodbye World!");  
        }  
        //This chain of alternatives could go on for a  
        //long time...  
    }  
}
```

Figure 2.3: Obfuscated Hello World Program

Obfuscation can make reverse-engineering, reading of code, writing of code difficult and time-consuming, but not necessarily impossible [7].

Byte Code Obfuscation

When source code is compiled it is transformed into intermediate code known as byte code. Byte code involves in transforming this byte code to another byte code that is same in functionality but makes reverse engineering particularly difficult. This byte code obfuscation is advance form of obfuscation which makes source code hard to decompile or recompile [8]. The main goal of byte code obfuscation is to produce

deviant class files, when these class files are decompiled it is difficult to recompile them.

2.3 Reverse Engineering of Binaries

Java is a platform independent programming language. When source code is compiled it is converted to an intermediate language known as class file. The class file contains a great amount of source code information. When class files are reverse engineered java files (source code files) can be obtained. When software is distributed to end user it is distributed in intermediate file format. This intermediate files can be reverse engineered using de-compilers or automated tools and making appropriate changes in the software. Developers can place some sort of protection in the software that can be removed by reverse engineering these files. Software's contains license checks that verify the presence of license files. However an attack could be launched by reverse engineering the intermediate files and removing those license checks. We will use a de-compiling tool to reverse engineer the intermediate files and support our claim.

There are different decompilers that offer the decompiling of class files (intermediate code) back to source code. JAD is a de-compiler available for Java. It decompiles the .class files back to .java files. Figure 2.4 shows that EntityCategory.class file is decompiled using de-compiler.

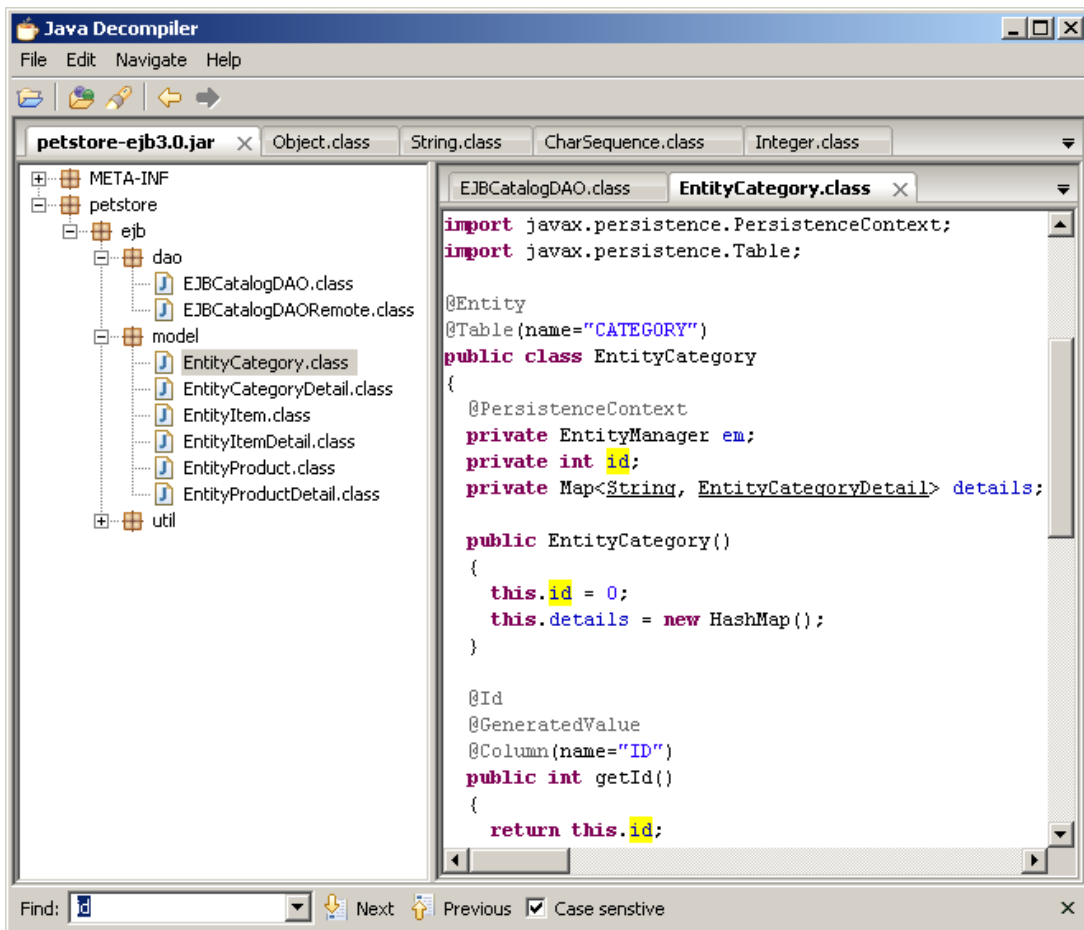


Figure 2.4: Decompiled .class files using de-compiler

2.4 Related Papers

The idea of protecting source code and its binaries was influenced from some of its related work

- Guy-Armand Yandji, Lui Lian Hao, Amir-Eddine Youssouf, Jules Ehoussou [9] presented a model for normal file encryption and decryption. The paper describes a methodology using AES and MD5 for encrypting files. The outcome file will as a result be hashed and strongly encrypted through the software.
- Xiufeng Zhang and Qiaoyan Wen [10] described the flexibility of Java language, which makes the protection become very difficult.

Using decompiler such as JAD we can easily extract the source code from the binary file. Therefore, any malicious users can use the anti-compiler tools to make reverse-engineering attacks. The paper presented an AOP-Based J2EE Source Code Protection technique in they gave solution to the problem that arises when encrypting J2EE applications.

- ByungRae Cha [11] presented a CRYPTEX model for protecting software source code. The model presented safe protection and access control of software source codes. The access control to the source code was achieved using digital certificate. The CRYPTEX consisted of software source codes and an algorithm to control access.
- A White Paper sponsored by CA Technologies [12] for Protecting API's against attack and hijack presented a secure API architecture. APIs are windows into applications and as with any window an API can easily be misused. APIs put applications under the hacker microscope and increase attack surface on client application. So a solution was presented using SecureSpan API proxy.
- SVN [13] and CVS [14] are used to control versions. A version control system keeps track of all work and all the changes in a set of files, and allows several developers to access them. Access to these files is controls using authentication and authorization if the files are not open source. Subversion can operate on network which will allow various people to modify and manage the same set of data.

Chapter 3

Research Methodology

“Research is to see what everybody else has seen, and to think what nobody else has thought.”

-Albert Szent-Gyorgyi

In this chapter, we will discuss the research methodology adopted during our thesis research. We will discuss different research approaches and also discuss in detail the approach that we have adopted in our thesis. Our research was divided into three main components. First, we had to find out all the possible attacks that could be launched on source code and binaries. Secondly, we presented an architecture that will protect the source code and binaries in both single user and teaming environment. And last phase will be, to implement our proof of concept. This chapter presents detail about selecting the right methodology for and effective research.

3.1 Research

The word research is composed of two syllables, “re” and “search” [15]. Re is a prefix meaning again or over again. Search is a verb meaning to make a thorough examine of. Together they make a noun meaning a systematic investigation in order to establish facts and reach new conclusions. It is basically gathering and searching information in order to answer particular question [16]. In research we come up with new ideas and try to execute those in order to help human nature. It is also about addressing new problems that haven’t been solved before. Research is communicating to larger audience the knowledge you have acquired. Research done systematically and proofed logically is considered good. Research could also be carried out on improving existing solutions to a problem.

3.1.1 Research Method & Research Methodology

Research method and Research methodology are two different terms that are often considered to be the same. Research methods are the methods, tools, techniques or processes that we use in our research [17]. Whereas research methodology is the approach adopted in answering research questions. Methodology is the principles that guide our research practices. In research methodology we study various steps that are adopted generally by the researchers in studying the research problem along with the logic.

3.1.2 Types of Research Methods

All the methods that are adopted during the any research are described below.

Applied research is a methodology used to solve a practical, specific problem of a group or an individual [18]. The study and research is used in education, medicine and business in order to find solutions that may solve everyday problem, scientific problems, develop technology or cure diseases.

Qualitative research is about finding people thinking. To understand their motivations and feelings it's about getting people to talk about their opinions [19]. It is about getting, analyzing and interpreting data by observing what people do or say. Qualitative research is more subjective and is performed by using different methods like interviews, meetings, group discussions.

Quantitative research aims to measure the quantity or amount and then compares it with past records to project future. In involves survey from a group of people by asking their opinions and views [20]. In quantitative research structured technique is applied like online questionnaires, telephonic or on-street interviews.

Fundamental research (Basic research) refers to the study and analysis that is meant to increase scientific knowledge [21]. Its main motivation is to expand man's knowledge not to invent or create something new.

Descriptive research describes specific behavior occurring in the environment [22]. It tells what exists and may help to uncover new facts and meaning like finding out the most common disease among children in a specific town which will tell what cure to do in order to live healthy life.

Conceptual research involves developing a theory to explain specific behavior or phenomena.

Empirical research does not involve theory rather it involves observation and experiments.

3.2 Thesis Research Approach

Our research is conducted in two main components. First we analyzed the behavior of IDE and find out how IDE saves source code and binaries. We needed to know all possible attacks that could be lunched on the source code and binaries. Storing the source code in plain form was not a good idea so we had to present a secure architecture that could protect these source code and binaries from getting

stolen or reverse engineered. We achieved our research by using more than one research methodologies. Deductive, Empirical, Analytical and Qualitative research methodologies have been combined, since we have diverse research objectives with multiple contributions.

3.2.1 Defining Research Area

As a first step, protection of source code and binaries were chosen as the area of research. Because the source files and their binaries have a lot of importance in software so this area was of our interest. A deductive approach was carried out for to find out different solutions proposed for the protection of source files and their binaries. Limitations of these solutions were observed and a problem statement was derived from that.

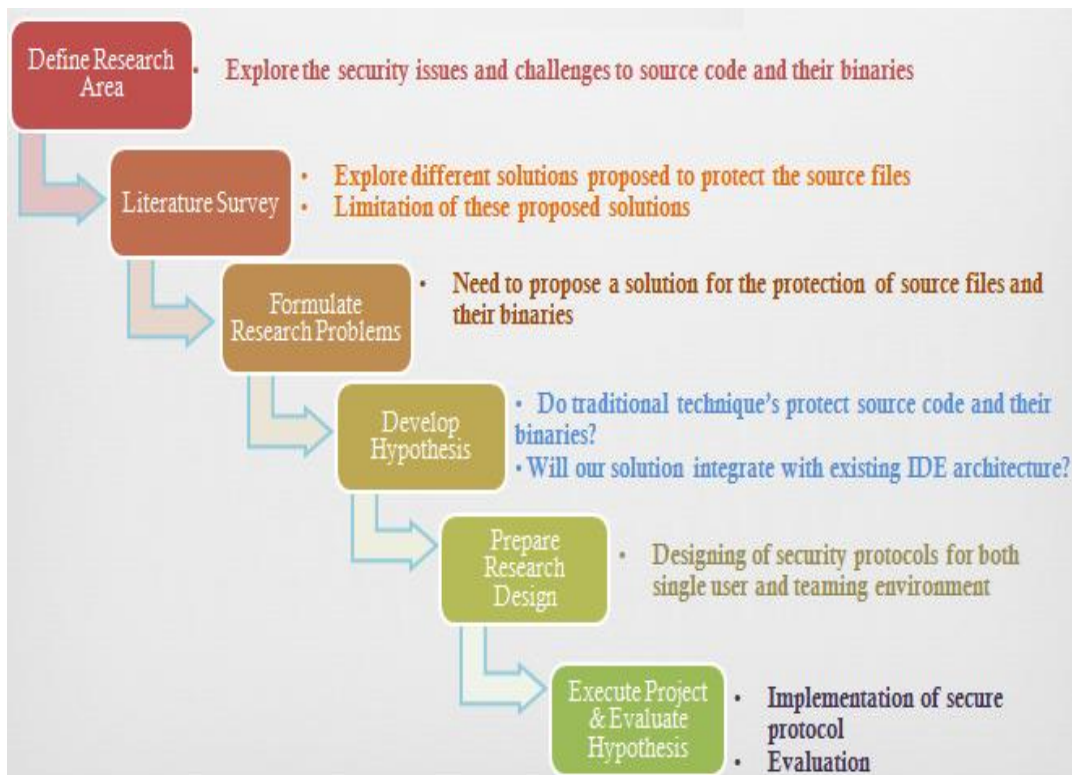


Figure 3.1: Thesis Research Workflow Diagram

3.2.2 Literature Survey

The most important step in any thesis is its literature survey. This step tells what you are planning to propose is valid or not. Existing solutions and their limitation are observed during literature survey. Literature survey also strengthens your research area. From literature survey problem statement is derived and validated.

In our literature survey we tried to find out different solution proposed for the protection of source code and their binaries. The importance of these source files and how binary files can be reversed engineered.

3.2.3 Problem Statement

After going through literature survey phase we derived our problem statement for the protection of source files and their binaries.

- To provide a secure architecture for the protection of source code and their binaries during the implementation phase in both single user and teaming environment.
- The source code and binaries are automatically encrypted and decrypted at backend. Only authorized user will have access to these files.

3.2.4 Hypothesis

The hypothesis of our research was,

- Does existing solution satisfy all the requirements by fully protecting the source code and their binaries and preventing them from all kind of attacks.
- Do these solutions protect binary files from getting reverse engineered?

3.2.5 Design Formulation

The design of our system was performed in during this phase. In this phase all the architectural, sequence and system diagrams were designed during this phase. The architectural diagram tells the architecture of the system. How different components will interact with each other. Sequence diagram was designed to know about the flow of messages, events and actions between objects and components.

3.3.6 Implementation and Verification of Protocol

The last step of our deductive approach was implementation and verification of our research. First of all we had to get an open source code of an IDE that could be modified. The right was needed so that less effort could be made. The next step included in developing different modules that will perform encryption and decryption on source file and their binaries. The modules were then integrated with the existing IDE source code. And the last step was to verify the protocol and the outcome from IDE.

3.3.7 Evaluation and Confirmation

The encrypted files were stored on hard drive in normal fashion. The protected encrypted source files and their binaries were evaluated using reverse engineering and different tools. The protected files were safe from any kind of attack

Chapter 4

System Architecture & Design

"Scientists investigate that which already is, Engineers create that which has never been."

-Albert Einstein

This chapter presents the overall design of our system. In this chapter we will discuss in detail the security protocols. Two types of protocols will be discussed in this chapter. These protocols will provide security measures to source code and their binaries. The detail architectural design of these protocols and working will be discussed in detail.

4.1 Security Protocols

Software implementation is a phase during software development. The implementation of software is either done by a single user or multiple users in a teaming environment. In single user environment a single user will use an IDE and generate the source code and binaries on his system. All these files are stored on local machine in plain form. In a teaming environment multiple users implement software and the source code is placed on a central location in plain form. All the users access the same source code and new files are uploaded with modification on server. We will discuss in detail both the protocols and their design.

4.1.1 Security protocol for Single User Environment

In a single user environment a user using IDE on his system/computer would face all the challenges to the source code and their binaries that we have discussed. The Source code is stored on the local drive in plain form. Anyone getting access to that drive can misuse or steal the source code. As the source code is in plain form the thief could launch this source code on his computer using compatible IDE without any extra effort. Similarly, binaries could be reverse engineered using some de-compiler like JAD to get source file or could be used to inject malicious code. So, we have to protect the source code files and their reprehensive binaries.

4.1.1.1 Architectural Design of the System

The main components of our security protocol are editor console, authentication module, protection module, authorization module, cryptographic engine and identity and credential management system. All these components will make up a secure protocol needed to provide protection to the source code and their binaries. We will discuss in detail all these components.

Editor console is the GUI user interacts with. The editor console is used for user input. It is used to launch IDE, input user credentials, software coding, close IDE etc.

Authentication module is a module that authenticates user via credential. This module activates when IDE is launched and validate any user using username and password. When user is authenticated eclipse IDE is launched.

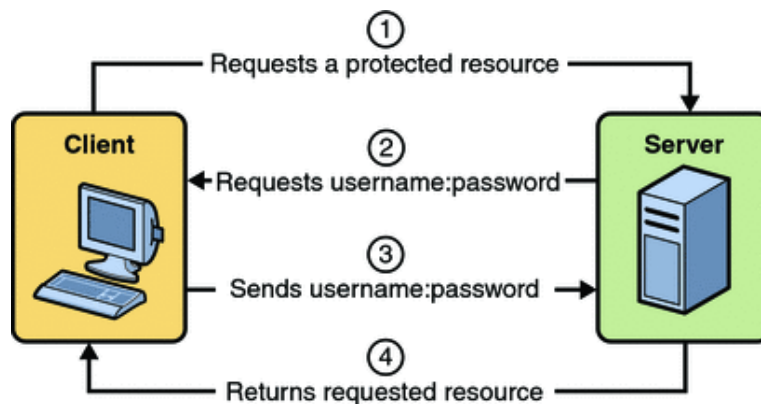


Figure 4.1: Authentication Diagram

The user authentication can be shown in figure 4.1. The user requests for some protected resource and is asked for the username and password. If the user enters correct username and password pair he would be authenticated and allowed to view the requested resource.

Protection module is the main module that controls source code protection and ownership of these files. Any existing protected files or opened files in IDE that need protection will be controlled by this module.

Authorization module controls the authorization process. Protected source files and their binaries will only be visible to authorized users. Authorization will be achieved through user password. Whenever user tries to open protected source files, these files would be decrypted using user's password. If these files were created by the same user then they would have been encrypted by the same user password. So,

decryption will be performed and these files will be visible to the user. Figure 2 shows the authorization process.

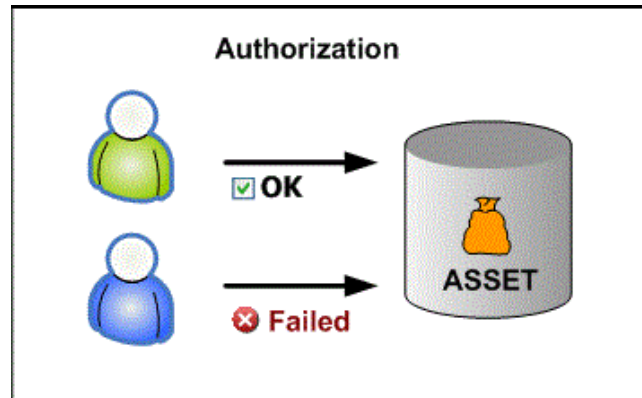


Figure 4.2: Authorization Diagram

Cryptographic engine performs the encryption and decryption of source files and their binaries. The protection module and the authorization module send source file and their binaries to cryptographic engine and asked to perform encryption or decryption on them.

Identity and Credential management system is used to store and retrieve the user credentials. The user credentials include username and password. The ICMS is accessed by the authentication module when authenticating user and accessed by the cryptographic engine to get user password and encrypt or decrypt source files and binaries with it.

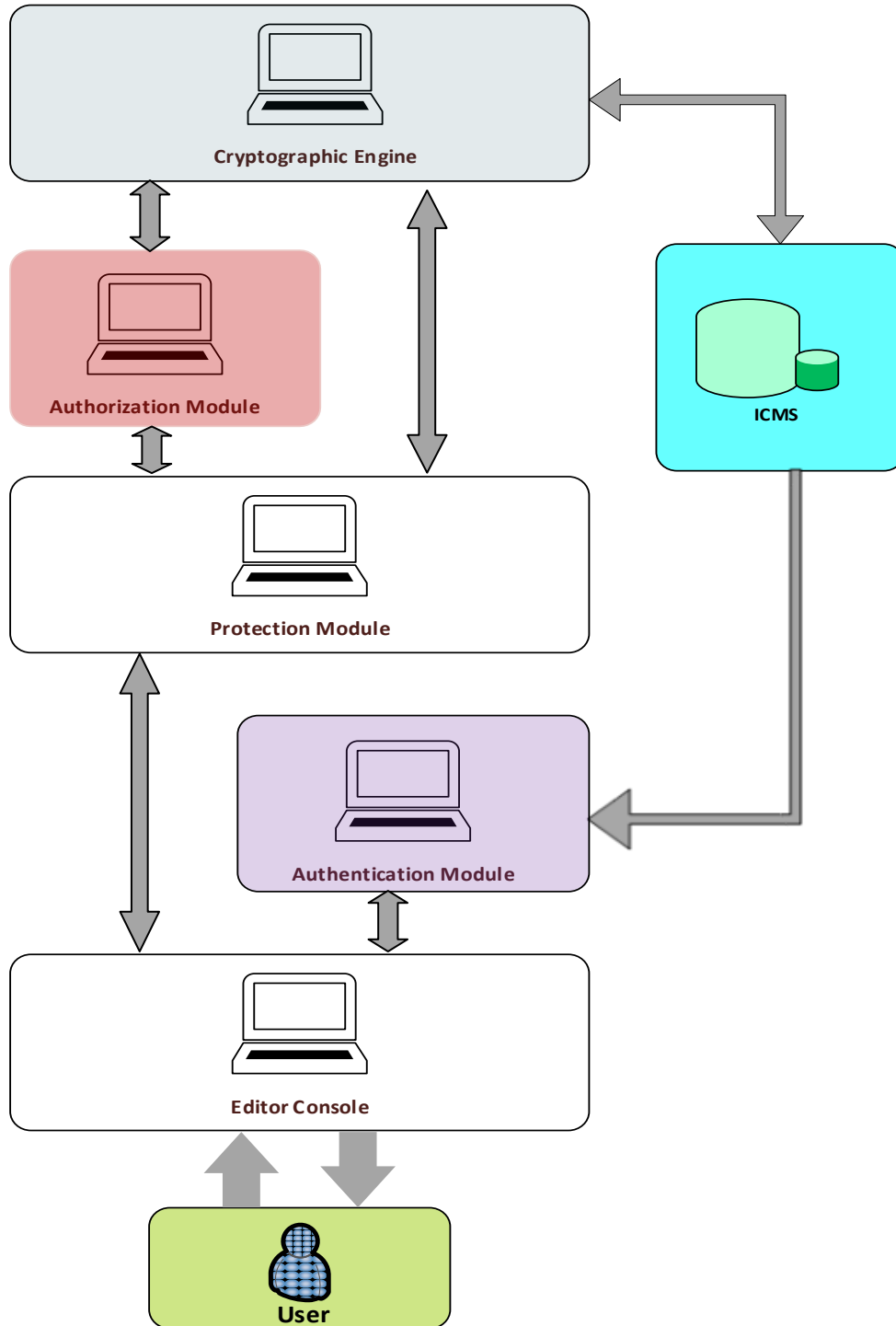


Figure 4.3: Architecture diagram for Single user

Architecture diagram of single user protection protocol can be seen in fig 4.3. First step would be to authenticate the user using the IDE. So when the user launches the IDE the authentication module activates and it opens a login screen asking for his credential (i.e. username and password). If the user is logging for the first time in IDE, he has to get registered and the hash of the password is taken and stored with username in ICMS. If his already registered the entered password has to be hashed and is compared with the hash that is already stored with a particular username. Now if the both hash match then the user would be login to the IDE. After a successful login the IDE would launch. If there is some kind of error the user would be asked to re-enter the credential, Skip and run IDE normally or exit.

Our main emphasis would be at start and exit of IDE. Because on these two states the source code should be protected. When the user tries to open an existing project in IDE, The protection module will ask the authorization module to check if he is authorized to view this project. The authorization module will send the protected source files to the cryptographic engine and will ask it to decrypt them. The cryptographic engine will decrypt these files with the user's password. If these files were encrypted with same user password then they will be decrypted without any error. After successful decryption the source files are available to user but if his wrong user he won't have access to it.

The authorization module will inform the protection main module that his the correct user. The protection main module will allow the user to view the file.

When the user tries to exit the IDE, first the source files would be saved automatically for any necessary changes then all the source files and binaries would be encrypted automatically with the existing user's key and then these encrypted source files and binaries would be stored on local drive.

In the given architecture if someone gets access to your source code or binaries he won't be able to open them in an IDE, because these files are in encrypted form rather than plain form.

4.1.2 Security Protocol for Teaming Environment

The teaming environment is quite an interesting environment in which developers working on same project would like to have the source code to be centralized. They could fetch the source code to their own machine. There should be some mechanism that would authenticate and authorized users on this source code. The mechanism that we would present would allow user's authentication, authorization, uploading and downloading of the source code.

4.1.2.1 Detailed Architectural Design

The main components of our security protocol are client application, authentication module, TGS, Data server, protection module, authorization module, cryptographic engine and credential management system CMS. All these components will make up a secure protocol needed to provide protection to the source code and their binaries. We will discuss in detail all these components.

Client application is an application running on client machine. It will interact with server to retrieve and upload source code from server.

Authentication module is a module that running at server end and used to authenticate user via credential. This module interacts with client application and validates user using username and password.

TGS ticket granting server grants ticket after user has been authenticated. The ticket is handed to client application which then takes the ticket and interacts with data server containing source code.

Data server contains the source code needed by the user. This server stores source code in encrypted form and provides only to authorized user. User will also upload his source code to this server. Version control of source code is also controlled by this system.

Protection module controls the authorization process. This module is revoked when user tries to get access to any source code from the given list.

Authorization module checks whether the user accessing the said source code is authorized for this action or not. This module will get the shared key from user and his name against the said project. If the said user is available in the authorized persons list, then this module will send the shared key and the source code to cryptographic engine for decryption.

Cryptographic engine controls the encryption and decryption process of this system. Authorization module will interact with this cryptographic engine to check security measures.

Credential management system CMS is a database storing the shared key value and the authorized persons list. Authorization module will interact with this system to retrieve and store credentials.

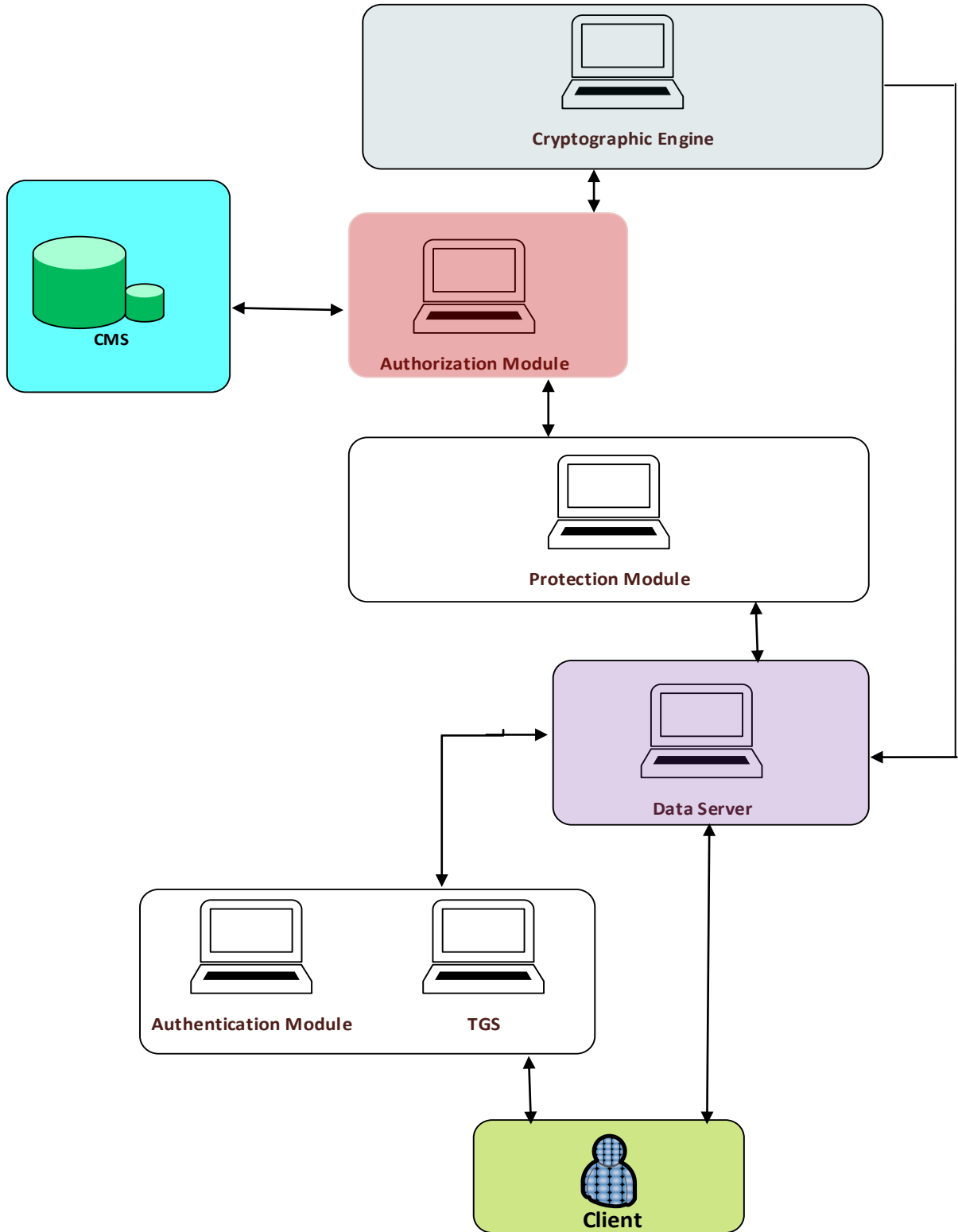


Figure 4.4: Architecture diagram for Teaming Environment

The teaming environment is an environment in which users would be working on a similar project and it has to be stored on some centralized location where everyone could have access to it. There would be a client module and server module to which the client would interact. User will add his credentials to the client module which would send it to the server module. As shown in figure 4.4 upon authentication of user the project list would be provided to the user.

Two servers would be needed to make this environment work (i.e. authentication/ TGS server and data server). First the client module will interact with the ATGS in order to achieve authentication. The client module will send username and password to the ATGS server after getting it from user.

The ATGS would have two modules running on it (i.e. ticket granting module & authentication module). The authentication module would verify that either the username or password provided are valid or not. The provided password would be taken hashed and compared with the one stored on the server, If they both match the user would be authenticated and then the ticket granting module would grant a ticket for the data server to the user. The user will take this ticket to the data server.

The ticket would be verified by the ticket verification module. If the ticket is valid then the user would be displayed with the list of all projects on the server. He would be able to upload a project or download one if his authorized to do that. First let's consider the uploading of project. The user will upload a project and it would be encrypted using a shared key. The owner of the project can also provide the list of users that are authorized to download this project. He has to share the shared key with them.

Now in the downloading part the user would be authenticated the same way and his ticket is also verified as discussed earlier. After seeing list of all projects the user would like to download one. The authorization module will check its authorization and if he is authorized then he would be asked to enter the shared key.

Now using this shared key the project would be decrypted and downloaded onto user's machine.

Chapter 5

Implementation

“Success is no accident. It is hard work, perseverance, learning, studying, sacrifice and most of all, love of what you are doing or learning to do”

-Pelé

In this chapter we will discuss the implementation details of our research. Eclipse IDE was chosen to add these security features. Because eclipse IDE is open source which means that its source code is available. So, we can get the source code of eclipse IDE and compile that code and make our necessary changes. We compiled eclipse source code within eclipse which means that eclipse was running its own source code. The process was slow because the source code was too much and it took time in compiling.

5.1 Getting the Source code of Eclipse IDE

Getting source code of Eclipse IDE was a task that took a great time. They are different ways you can get the eclipse source code discussed below.

5.1.1 Import as Plug-in

Source code of eclipse can be imported from its self. We can import all or a specific plug-in from eclipse target directory. When eclipse IDE is downloaded it contains a folder with all the plug-ins in it. Running eclipse IDE and selecting import from the file menu.

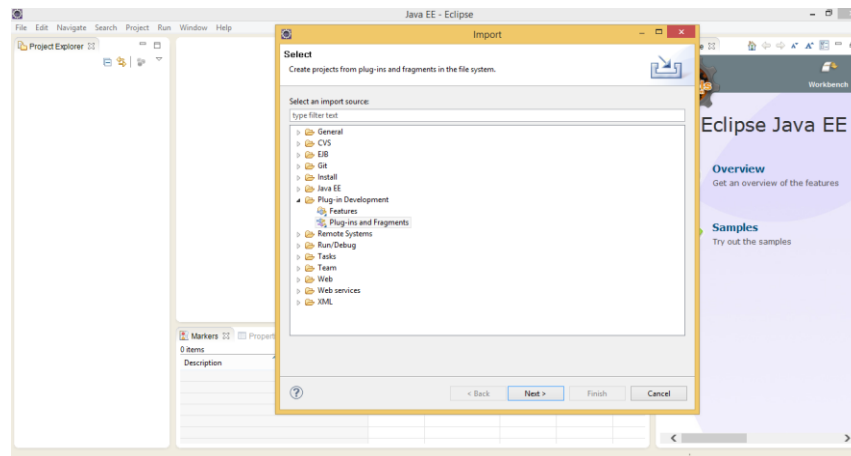


Figure 5.1: Eclipse Import Wizard

Selecting Plug-in and fragmentation from the import menu and click next.

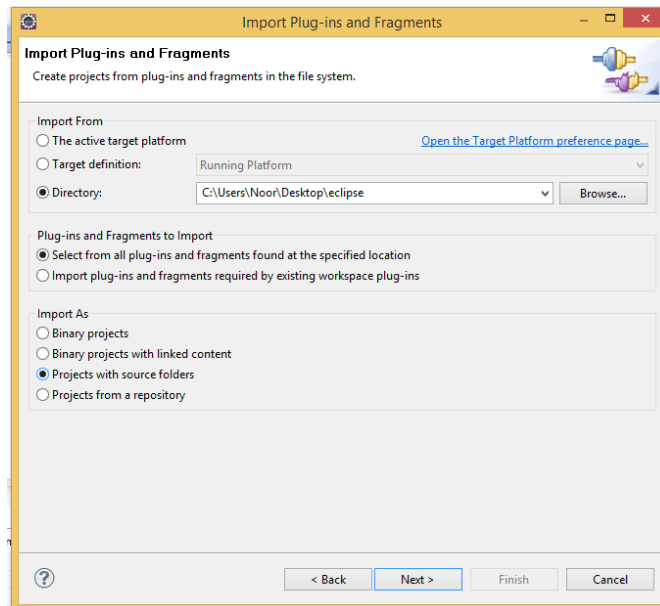


Figure 5.2: Eclipse Import Plug-in and Fragmentation Diagram

You have to select the directory from which the plug-ins will be imported. Giving the complete path of the directory and selecting projects with source folders and clicking next.

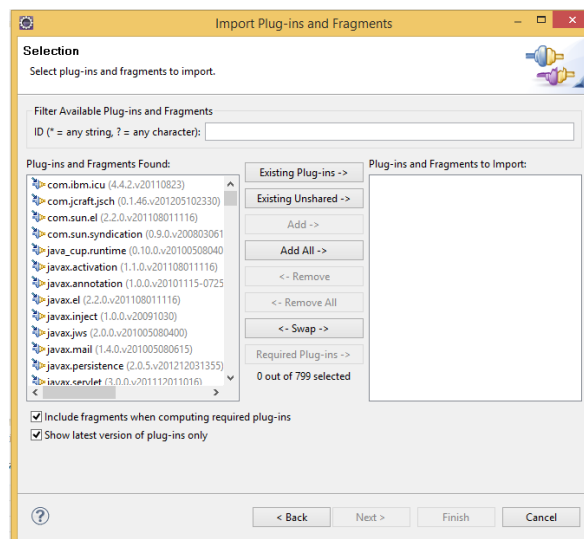


Figure 5.3: Eclipse Importing Existing Plug-ins

The selection menu will ask about the plug-ins you like to import in our case we imported all the plug-ins from the list.

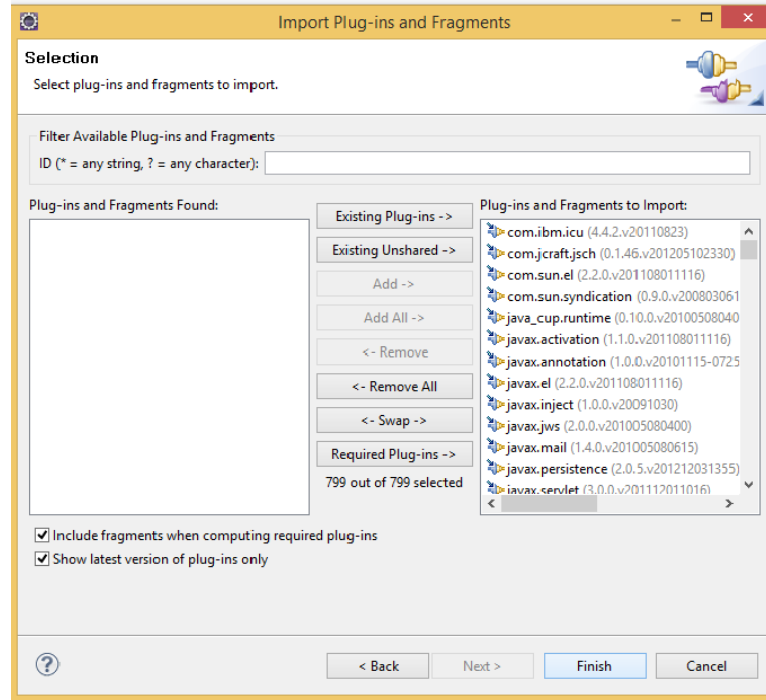


Figure 5.4: Eclipse Importing Existing Plug-ins(2)

Click finish and it will start importing the plug-ins from eclipse IDE into eclipse IDE. After importing all the plug-ins, eclipse IDE will provide all the complete list of the plug-ins in its project explorer.

The problem with this type of import is that they are in the form of class files not java files. You have to attach the source folders to it. You can run these files and eclipse would be launched but you can't view the source files. As we only required the source code of one or two plug-ins in which we had to make our changes. So we adopted two steps for this issue.

- Download required plug-in source code and attach it.
- Use JAD to reverse engineer the binaries of plug-ins and get source files and attach them.

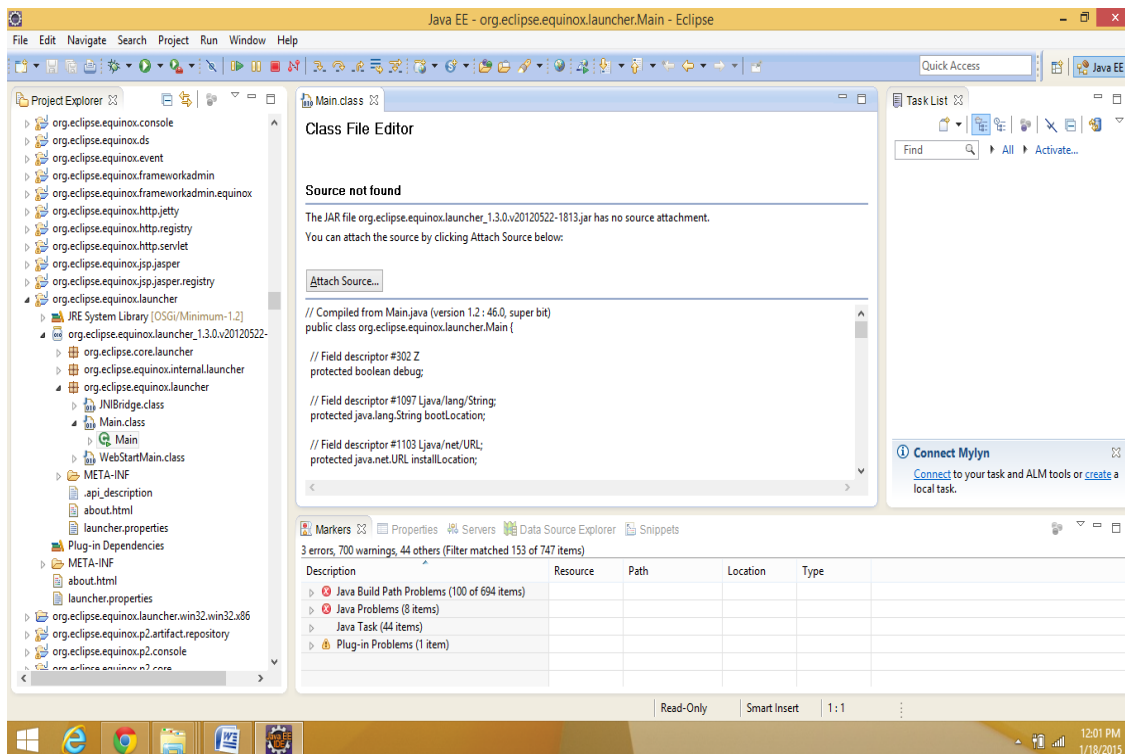
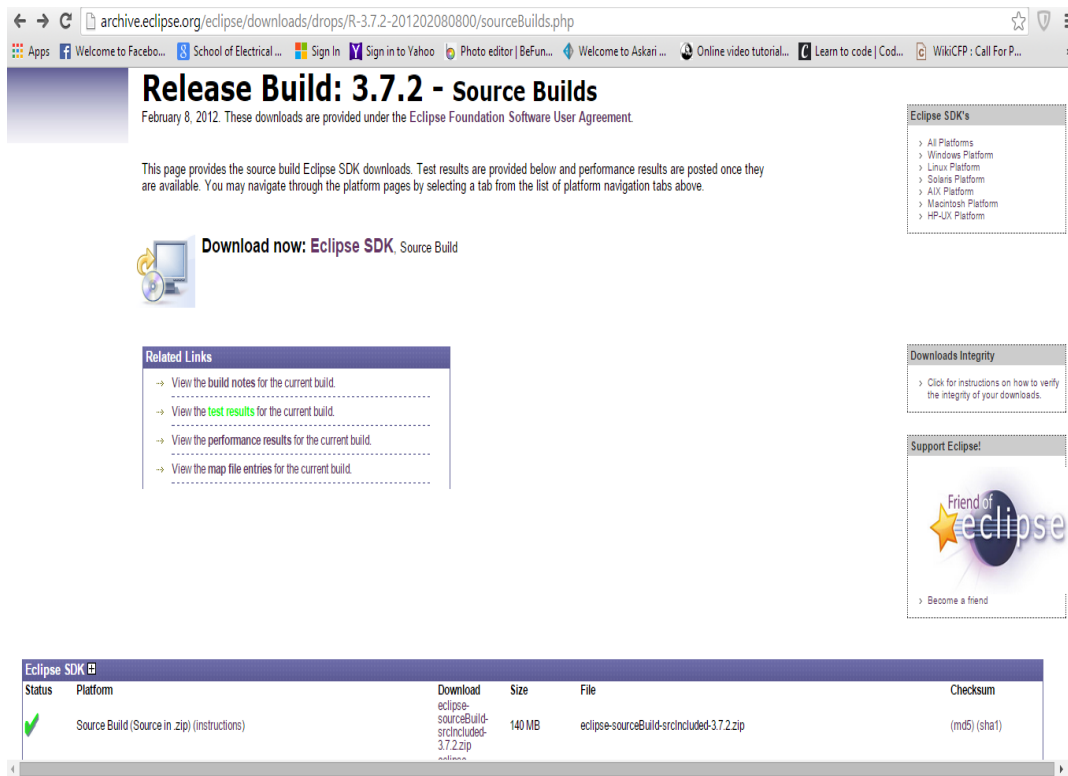


Figure 5.5: Eclipse Source code

Source code of the required plug-ins were found using above two techniques but when it was attached, it was not editable (means you can make any change in those source files). So this type of method could only be adopted if you want to make any new plug-in for eclipse and check the behavior of eclipse when it's integrated with it.

5.1.2 Complete Source code available on Eclipse Website

Eclipse complete source code could be downloaded from eclipse website.



The screenshot shows the Eclipse website page for the 3.7.2 source build. The page title is "Release Build: 3.7.2 - Source Builds" and it is dated February 8, 2012. The page provides information about the source build Eclipse SDK downloads and includes a "Download now: Eclipse SDK" button. There is a "Related Links" section with links to build notes, test results, performance results, and map file entries. A table at the bottom lists the source build download details.

Status	Platform	Download	Size	File	Checksum
✓	Source Build (Source in zip) (instructions)	eclipse-sourceBuild-srcIncludgd-3.7.2.zip	140 MB	eclipse-sourceBuild-srcIncludgd-3.7.2.zip	(md5) (sha1)

Figure 5.6: Eclipse 3.7.2 Build on Website

The folder contains all the plug-ins source code java files. We can import the plug-ins into eclipse and compile them. After importing and compiling these files there were too many errors or dependencies in it. It could have taken too much time to remove these errors and dependencies so we decided that its better to find an error free code.

5.1.3 Git Repository

Eclipse plug-ins source code was finally downloaded from grepcode repository. The grepcode repository contains the source code of all the versions of eclipse.

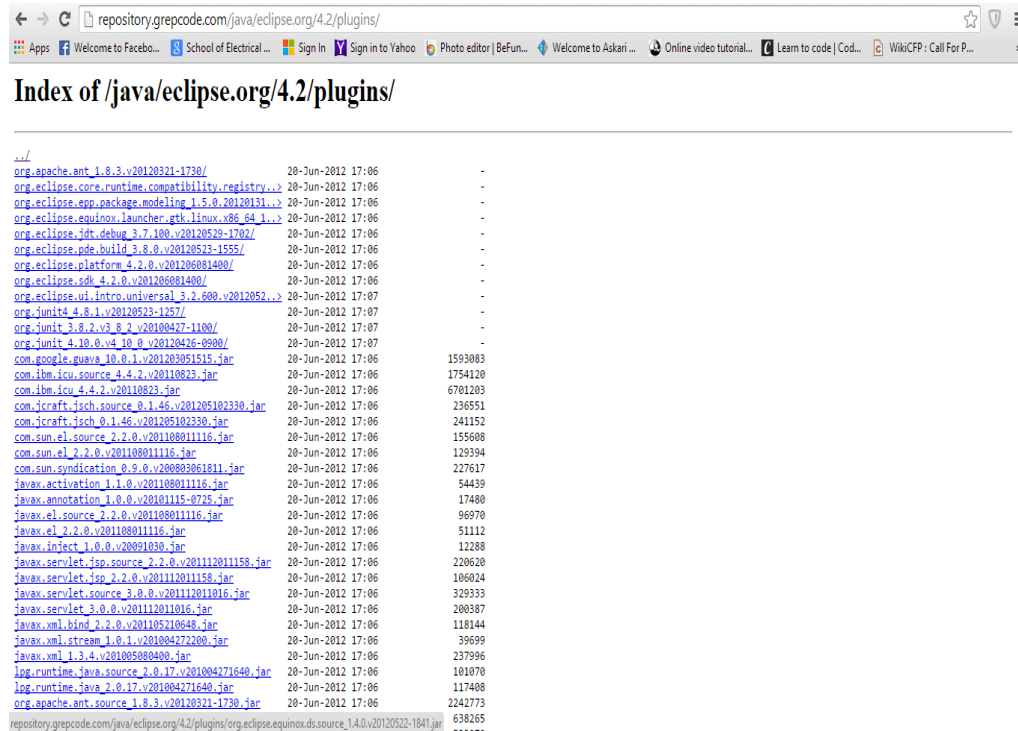


Figure 5.7: Git Repository

We were running eclipse juno so we downloaded the 4.2 version source code of eclipse IDE. It took a great time to get these files from repository because they were around 900 files that we had to download. After downloading all the files we imported these files into eclipse and compiled them. The result was great we had eclipse running within it and the source files were editable.

5.2 Plug-in Spy

Running UI modules information can be found with Plug-in spy. By pressing **Alt+Shift+F1** you can get information about the currently selected user interface component in your Eclipse IDE [23]. This way immediate access to the plug-in which is currently running can be found.



By pressing **Alt+Shift+F2** and select any menu then you could find out the class or plug-in contributing in this menu.

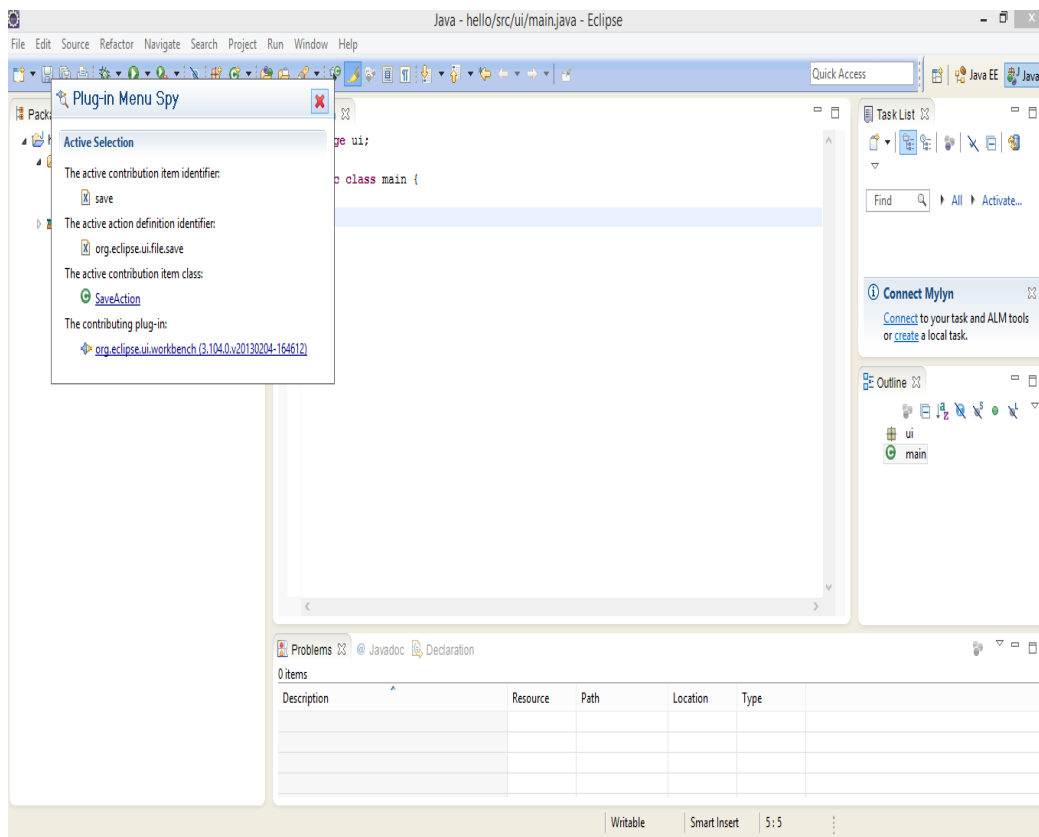


Figure 5.8: Plug-in Spy

In diagram 5.8 we can see the class contributing in save action and the plug-in containing this class.

5.3 Development

In this part of thesis we will discuss all the development done in our thesis research. We will describe how we protected our source code and binaries in IDE.

5.3.1 Running eclipse Source code

The first part was to compile and execute eclipse source code that we fetched from grepcode repository. We imported all the plug-ins in eclipse and executed them. The outcome was a running eclipse within eclipse.

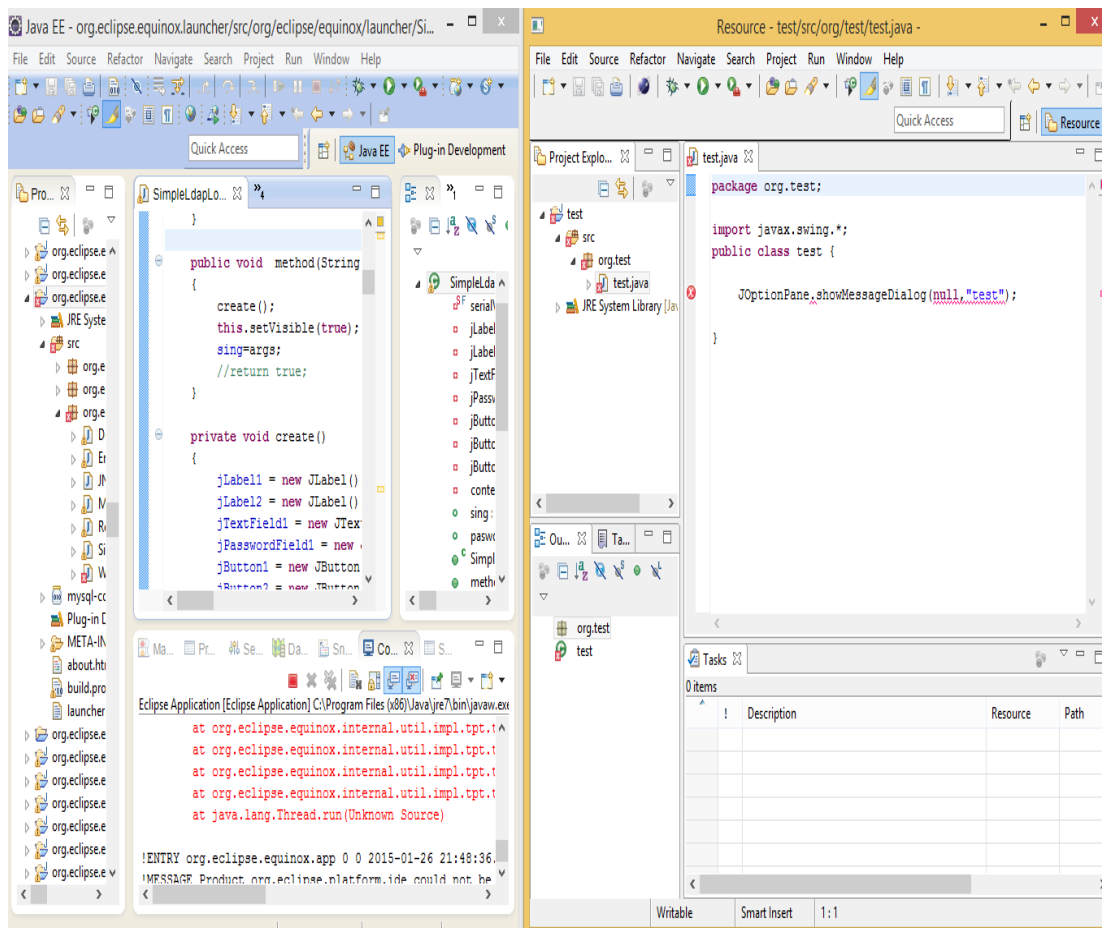


Figure 5.9: Compiling Eclipse Source code

5.3.2 Setting Startup Environment

Now we had to set the eclipse startup environment. This startup environment was the environment of virtual eclipse that was running in eclipse IDE. We had to set up this environment so that we can select what to run in virtual eclipse IDE. This environment could be set by running the run configurations. As we have specified the location folder of the workspace for our virtual eclipse IDE.

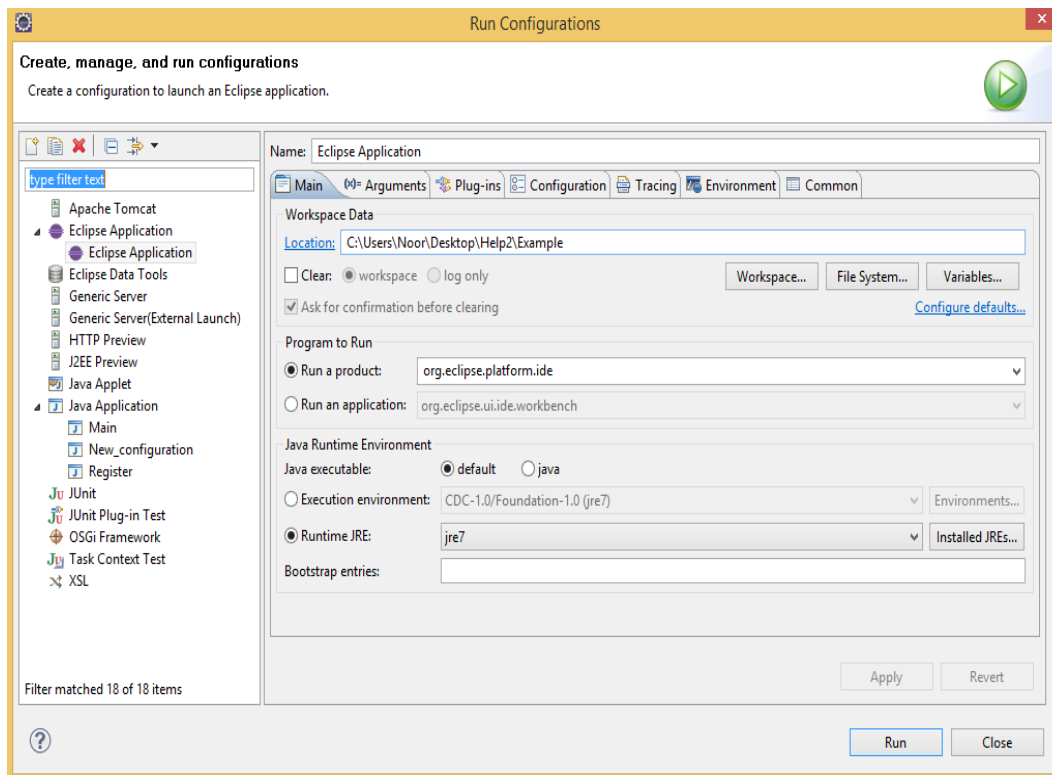


Figure 5.10: Eclipse Startup Environment

5.3.3 Required Plug-in and classes

Eclipse is a set of plug-ins. All the plug-in have different functionalities which when combined will result into eclipse IDE. A plug-in controls the start and closing of eclipse as well, we found out that `org.eclipse.equinox.launcher` was the said plug-in to control this feature and the main class in this plug-in was the class containing the main function.

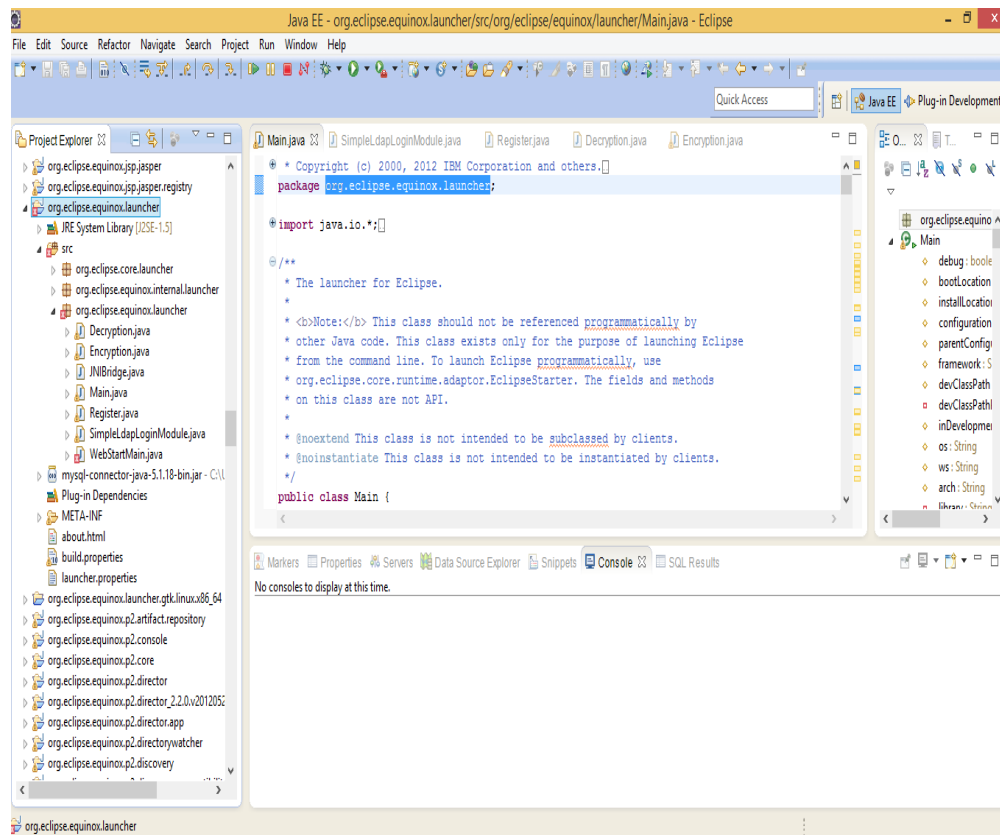


Figure 5.11: Required Plug-ins

We had to make the necessary changes in this class to make our architecture work. Authentication module had to be called at the start of main function and encryption module had to be called while exiting the main function.

5.3.4 Authentication Module

The authentication module controls the authentication part of an application. An authentication module is created to perform the said functionalities. The authentication module should accommodate existing as well as new user in the IDE. Mysql database was used to store the user ID's and passwords.

This Login module class instance was created in Main class and its function was called.

```
public static void main(String[] args) {
```

```
SimpleLdapLoginModule s=new SimpleLdapLoginModule();  
s.method(args);
```

Upon compiling the said code we had achieved the authentication part of our development. The outcome was a GUI launched at startup of eclipse asking user to enter username and password.

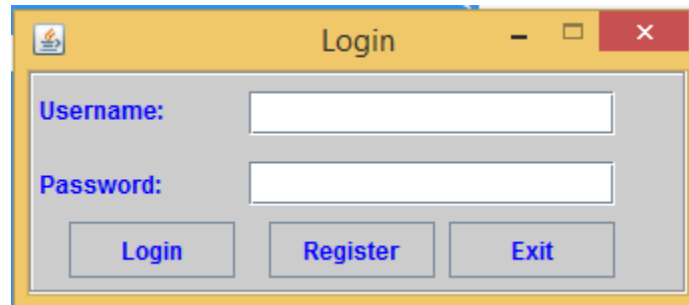


Figure 5.12: Login Screen

5.3.5 Encryption and Decryption Functions

The encryption and decryption was performed using AES. AES is a symmetric key block cipher algorithm. The encryption module was called on closing eclipse IDE in main class. The decryption module was same with functionalities in reverse.

5.3.6 Final Outcome

The final outcome of our thesis development was protected source code and binaries. We can see in figure 5.13 that the source file is in plain form and we can view its contents using notepad.

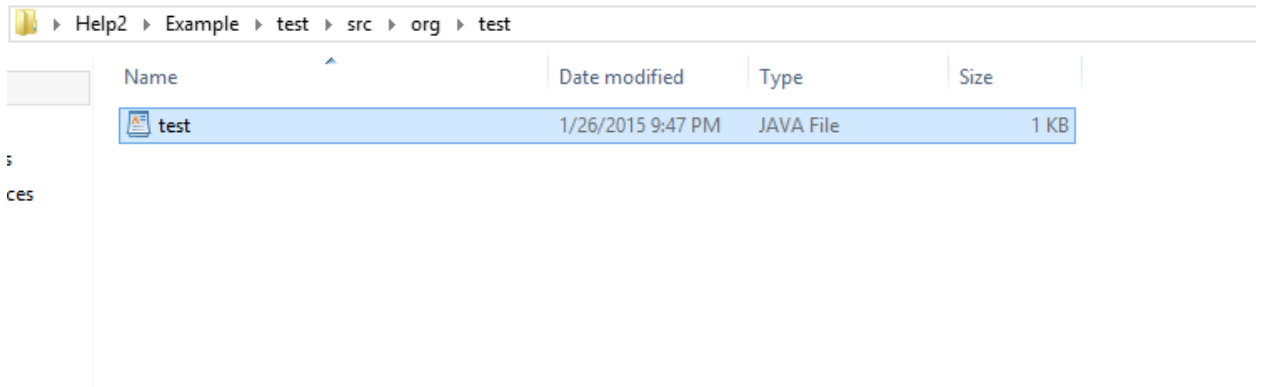


Figure 5.13: Source file in plain form

Figure 5.14 shows that the binary file is not protected and we can get the source code from it using any de-compiler like JAD.

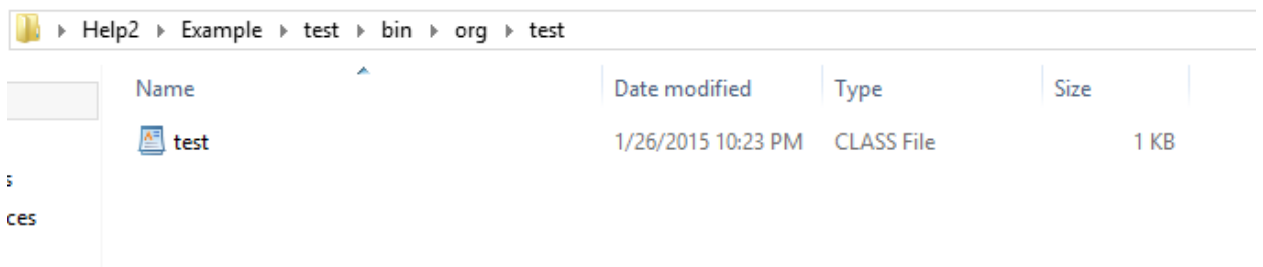


Figure 5.14: Binary file in plain form

Figure 5.15 shows that the source file is protected and cannot be viewed with any time of editor. The `java.enc` extension was used so that when we decrypt these source file and binaries we don't lose the original extensions. Just to differentiate between source files and binaries.

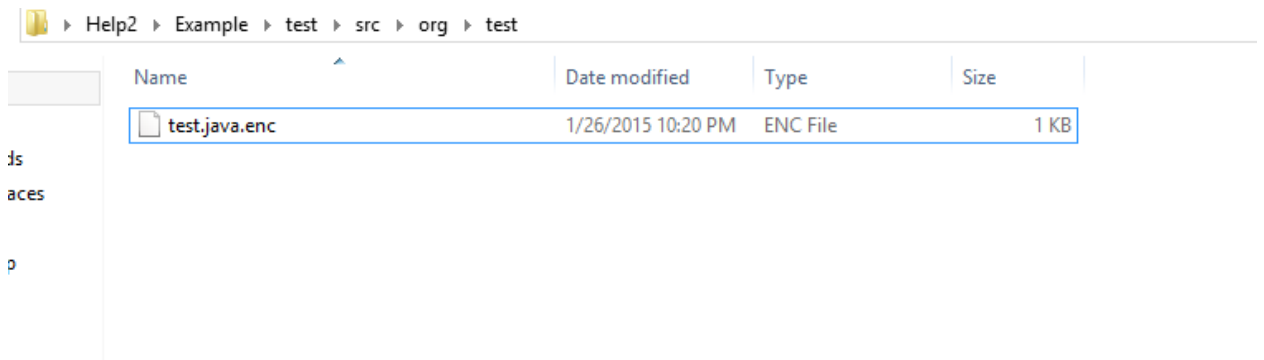


Figure 5.15: Source file in Encrypted form

Figure 5.16 shows that binary file is encrypted/ protected and we can't decompile such file using any de-compiler.

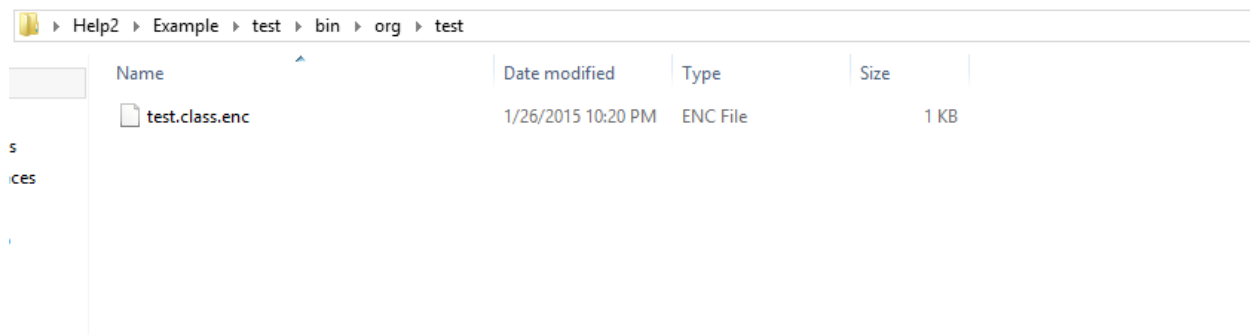


Figure 5.16: Binary file in Encrypted form

So our source files and binaries are protected automatically without the developer's involvement. He simply does not need to care about their protection.

Chapter 6

Conclusion & Future Directions

“Yesterday is gone. Tomorrow has not yet come. We have only today. Let us begin.”

— Mother Teresa

6.1 Conclusion

In the current era of software, software plays an important role in almost every aspect of human life. Software has a big role in modernizing the human lives i.e. imagine banking environment before banking software solutions, things were written and stored on paper. Just as software is very important nowadays, software source code is very important for software to survive. Source code is the code that is used to develop any software and it is further used to add any kind of modification in the software. The source code is kept safe and never distributed to the user (unless the software is open source). The software is distributed in the form of executables. During the developing phase of the software the source code is compiled using compatible compiler and they generate binary files. These binary files are not machine specific and can be run on any machine. These binary files are almost ignored during software development. These files have a great importance and can be reversed engineered to get the source code. These files are reversed engineered using some de-compiler like JAD.

As the source code and binaries have great importance they shouldn't be kept on hard disk in plain form. Anyone getting access to these files can misuse them. These files can be used to add unwanted feature in software, redistribute the software with some abnormal behavior, understand its working logic's, bypass license checks etc. So there was a need to protect these files.

We presented a solution that automatically encrypts and decrypts the source files and their binaries at backend. Only authorized users can access the source code. If somehow this source code or their binary files are stolen they are of no use because they are protected. The solution is presented for both the single user as well as teaming environment. The solution is an easy way to protect the source code and their binaries and any single code written would be protected automatically at backend. No source code or their binary will be stored in plain form.

6.2 Future Research

To date no effective solution is available to protect the source code or their binaries. Our solution is sufficient and enough to provide protection to the source code and their binaries. But due to the limitation of encryption algorithm, the protection mechanism becomes slow when it comes to encrypting and decrypting large amount of source code. So, some light weight algorithm could be introduced in place of AES which could speedup this process.

Appendix A

Source Code (JAVA)

1. Authentication Code

```
package org.eclipse.equinox.launcher;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.Statement;

public class SimpleLdapLoginModule extends JFrame
{
    private static final long serialVersionUID = 7526472295622776147L;
    private JLabel jLabel1;
    private JLabel jLabel2;
    private JTextField jTextField1;
    private JPasswordField jPasswordField1;
    private JButton jButton1;
    private JButton jButton2;
    private JButton jButton3;
    private JPanel contentPane;
    public String[] sing;
```

```
public String paswd;
public SimpleLdapLoginModule()
{
    super();
}
public void method(String[] args)
{
    create();
    this.setVisible(true);
    sing=args;
}
private void create()
{
    jLabel1 = new JLabel();
    jLabel2 = new JLabel();
    jTextField1 = new JTextField();
    jPasswordField1 = new JPasswordField();
    jButton1 = new JButton();
    jButton2 = new JButton();
    jButton3 = new JButton();
    contentPane = (JPanel)this.getContentPane();
    jLabel1.setHorizontalAlignment(SwingConstants.LEFT);
    jLabel1.setForeground(new Color(0, 0, 255));
    jLabel1.setText("Username:");
    jLabel2.setHorizontalAlignment(SwingConstants.LEFT);
    jLabel2.setForeground(new Color(0, 0, 255));
    jLabel2.setText("Password:");
    jTextField1.setForeground(new Color(0, 0, 255));
    jTextField1.setSelectedTextColor(new Color(0, 0, 255));
    jTextField1.setToolTipText("Enter your username");
    jTextField1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e)
    {
```

```
jTextField1_actionPerformed(e);
}
});

jPasswordField1.setForeground(new Color(0, 0, 255));
jPasswordField1.setToolTipText("Enter your password");
jPasswordField1.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jPasswordField1_actionPerformed(e);
}
});

jButton1.setBackground(new Color(204, 204, 204));
jButton1.setForeground(new Color(0, 0, 255));
jButton1.setText("Login");
jButton1.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jButton1_actionPerformed(e);
}
});

jButton2.setBackground(new Color(204, 204, 204));
jButton2.setForeground(new Color(0, 0, 255));
jButton2.setText("Exit");
jButton2.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jButton2_actionPerformed(e);
}
});

jButton3.setBackground(new Color(204, 204, 204));
jButton3.setForeground(new Color(0, 0, 255));
```

```
        jButton3.setText("Register");
        jButton3.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e)
            {
                jButton3_actionPerformed(e);
            }
        });
        contentPane.setLayout(null);
        contentPane.setBorder(BorderFactory.createEtchedBorder());
        contentPane.setBackground(new Color(204, 204, 204));
        addComponent(contentPane, jLabel1, 5,10,106,18);
        addComponent(contentPane, jLabel2, 5,47,97,18);
        addComponent(contentPane, jTextField1, 110,10,183,22);
        addComponent(contentPane, jPasswordField1, 110,45,183,22);
        addComponent(contentPane, jButton1, 20,75,83,28);
        addComponent(contentPane, jButton3, 120,75,83,28);
        addComponent(contentPane, jButton2, 210,75,83,28);
        this.setTitle("Login");
        this.setLocation(new Point(76, 182));
        this.setSize(new Dimension(335, 140));
        this.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        this.setResizable(false);
    }

    /** Add Component Without a Layout Manager (Absolute Positioning) */
    private void addComponent(Container container,Component c,int x,int y,int width,int height)
    {
        c.setBounds(x,y,width,height);
        container.add(c);
    }

    private void jTextField1_actionPerformed(ActionEvent e)
```

```

{
}
private void jPasswordField1_actionPerformed(ActionEvent e)
{
}
private void jButton3_actionPerformed(ActionEvent e)
{
    new Register();
}
private void jButton2_actionPerformed(ActionEvent e)
{
    System.exit(1);
}
public String Paswd_Fun()
{
    return paswd;
}
private void jButton1_actionPerformed(ActionEvent e)
{
    System.out.println("\njButton1_actionPerformed(ActionEvent e) called.");
    String user = new String(jTextField1.getText());
    //@SuppressWarnings("deprecation")
    paswd = new String(jPasswordField1.getText());
    if(user.equals("") || paswd.equals("")) // If password and username is empty > Do this >>>
    {
        jButton1.setEnabled(false);

        JLabel errorFields = new JLabel("<HTML><FONT COLOR = Blue>You must enter a username
and password to login.</FONT></HTML>");
        JOptionPane.showMessageDialog(null,errorFields);
        jTextField1.setText("");
        jPasswordField1.setText("");
        jButton1.setEnabled(true);
        this.setVisible(true);
    }
}

```

```

}
else if(user!=null && paswd!=null)
{
try{
String databaseUsername = "";
String databasePassword = "";
Class.forName("com.mysql.jdbc.Driver");
Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/test","root","1234");
Statement stmt = con.createStatement();
String SQL = "SELECT * FROM login WHERE username='" + user + "' && password='" + paswd+
"';";
ResultSet rs = stmt.executeQuery(SQL);
while (rs.next()) {
databaseUsername = rs.getString("username");
databasePassword = rs.getString("password");
}

if (user.equals(databaseUsername) && paswd.equals(databasePassword)) {
JOptionPane.showMessageDialog(null,"Successful Login");
Window w = SwingUtilities.getWindowAncestor(contentPane);
w.setVisible(false);
Decryption Dec=new Decryption();
Dec.Decrypt_Fun(paswd);
JOptionPane.showMessageDialog(null,"Project Decrypted");
Main m=new Main();
m.run(sing,paswd);
} else {
JOptionPane.showMessageDialog(null,"Incorrect Username or Password. Please retry");
}}
catch (Exception esp) {

esp.printStackTrace();
}}};

```

2. Encryption Code

```
package org.eclipse.equinox.launcher;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;
import java.util.Queue;
import java.io.IOException;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.util.zip.ZipEntry;
import java.util.zip.ZipOutputStream;
import java.io.File;
import javax.swing.*;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.security.SecureRandom;
import javax.crypto.Cipher;
import javax.crypto.CipherOutputStream;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

public class Encryption {
    public void Encrypt_Fun(String Password)throws Exception
    {
        String key=Password;
```



```

try{
List<File> allFiles = new ArrayList<File>();
Queue<File> dirs = new LinkedList<File>();
dirs.add(new File("C:\\Users\\Noor\\Desktop\\Help2\\Example"));
while (!dirs.isEmpty()) {
for (File f : dirs.poll().listFiles()) {
if (f.isDirectory()) {
dirs.add(f);

} else if (f.isFile()) {
String filename = f.getName();
if (filename.endsWith(".java") || filename.endsWith(".class"))
allFiles.add(f);
}
}
}
for (File file : allFiles) {
String tempFileName1=file.getCanonicalPath()+".enc";
copy(Cipher.ENCRYPT_MODE, file.getCanonicalPath(),tempFileName1, key);
BufferedInputStream input1 = new BufferedInputStream(new FileInputStream(tempFileName1));
BufferedOutputStream      output1      =      new      BufferedOutputStream(new
FileOutputStream(file.getCanonicalPath()));
byte[] buf = new byte[1024];
int len;
while ((len = input1.read(buf)) > 0) {
output1.write(buf, 0, len);
}
input1.close();
output1.close();
File myfile = new File(file.getCanonicalPath());
myfile.delete();
}
}

```

```

catch(IOException e)
{
}
}

private static final int IV_LENGTH=16; //key length

public static byte[] encrypt(String plainText, String password) throws Exception {
    ByteArrayInputStream bis = new ByteArrayInputStream(plainText.getBytes("UTF8"));
    ByteArrayOutputStream bos = new ByteArrayOutputStream();
    encrypt(bis, bos, password);
    return bos.toByteArray();
}

public static void encrypt(InputStream in, OutputStream out, String password) throws Exception{
    SecureRandom r = new SecureRandom();
    byte[] iv = new byte[IV_LENGTH];
    r.nextBytes(iv);
    out.write(iv); //write IV as a prefix
    out.flush();

    Cipher cipher = Cipher.getInstance("AES/CFB8/NoPadding");
    //Cipher.getInstance("DES/ECB/PKCS5Padding";"AES/CBC/PKCS5Padding"
    SecretKeySpec keySpec = new SecretKeySpec(password.getBytes(), "AES");
    IvParameterSpec ivSpec = new IvParameterSpec(iv);
    cipher.init(Cipher.ENCRYPT_MODE, keySpec, ivSpec);
    out = new CipherOutputStream(out, cipher);
    byte[] buf = new byte[1024];
    int numRead = 0;
    while ((numRead = in.read(buf)) >= 0) {
        out.write(buf, 0, numRead);
    }
    out.close();
}

public static void copy(int mode, String inputFile, String outputFile, String password) throws
Exception {
    BufferedInputStream is = new BufferedInputStream(new FileInputStream(inputFile));

```

```
BufferedOutputStream os = new BufferedOutputStream(new FileOutputStream(outputFile));
if(mode==Cipher.ENCRYPT_MODE){
encrypt(is, os, password);
}
is.close();
os.close();
}
}
```

3. Decryption Code

```
package org.eclipse.equinox.launcher;

import java.io.IOException;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.util.ArrayList;

import java.util.LinkedList;

import java.util.List;

import java.util.Queue;

import java.util.zip.ZipEntry;

import java.util.zip.ZipOutputStream;

import java.io.File;

import javax.swing.*;

import java.io.BufferedInputStream;

import java.io.BufferedOutputStream;

import java.io.BufferedReader;

import java.io.ByteArrayInputStream;

import java.io.ByteArrayOutputStream;
```

```
import java.io.InputStream;

import java.io.InputStreamReader;

import java.io.OutputStream;

import java.security.SecureRandom;

import javax.crypto.Cipher;

import javax.crypto.CipherInputStream;

import javax.crypto.CipherOutputStream;

import javax.crypto.spec.IvParameterSpec;

import javax.crypto.spec.SecretKeySpec;

public class Decryption {

    public void Decrypt_Fun(String Password)throws Exception

    {

        String key=Password;

        try{

            List<File> allFiles = new ArrayList<File>();

            Queue<File> dirs = new LinkedList<File>();

            dirs.add(new File("C:\\Users\\Noor\\Desktop\\Help2\\Example"));

            while (!dirs.isEmpty()) {

                for (File f : dirs.poll().listFiles()) {

                    if (f.isDirectory()) {

                        dirs.add(f);

                    } else if (f.isFile()) {

                        String filename = f.getName();

                        if (filename.endsWith(".enc"))
```

```
allFiles.add(f);  
  
}  
  
}  
  
}  
  
for (File file : allFiles) {  
  
String tempFileName1=file.getCanonicalPath()+".dec";  
  
copy(Cipher.DECRYPT_MODE, file.getCanonicalPath(), tempFileName1, key);  
  
BufferedInputStream input = new BufferedInputStream(new  
FileInputStream(tempFileName1));  
  
BufferedOutputStream output = new BufferedOutputStream(new  
FileOutputStream(file.getCanonicalPath()));  
  
byte[] buf = new byte[1024];  
  
int len;  
  
while ((len = input.read(buf)) > 0) {  
  
output.write(buf, 0, len);  
  
}  
  
input.close();  
  
output.close();  
  
String foo = tempFileName1;  
  
foo = foo.substring(0, foo.lastIndexOf('.'));  
  
foo = foo.substring(0, foo.lastIndexOf('.'));  
  
File oldFileName = new File(tempFileName1);  
  
File newFileName = new File(foo);  
  
oldFileName.renameTo(newFileName);  
  
File myfile = new File(file.getCanonicalPath());
```

```
myfile.delete();  
  
}  
  
}  
  
catch(Exception e){  
e.printStackTrace();  
  
}  
  
}  
  
private static final int IV_LENGTH=16;//key length  
  
static String stripExtension (String str) {  
if (str == null) return null;  
int pos = str.lastIndexOf(".");  
if (pos == -1) return str;  
return str.substring(0, pos);  
}  
  
public static byte[] decrypt(String cipherText, String password) throws Exception {  
byte[] cipherTextBytes = cipherText.getBytes();  
ByteArrayInputStream bis = new ByteArrayInputStream(cipherTextBytes);  
ByteArrayOutputStream bos = new ByteArrayOutputStream();  
decrypt(bis, bos, password);  
return bos.toByteArray();  
}  
  
public static void decrypt(InputStream in, OutputStream out, String password) throws  
Exception{  
byte[] iv = new byte[IV_LENGTH];
```

```

in.read(iv);

Cipher cipher = Cipher.getInstance("AES/CFB8/NoPadding");
// "DES/ECB/PKCS5Padding"; "AES/CBC/PKCS5Padding"

SecretKeySpec keySpec = new SecretKeySpec(password.getBytes(), "AES");

IvParameterSpec ivSpec = new IvParameterSpec(iv);

cipher.init(Cipher.DECRYPT_MODE, keySpec, ivSpec);

in = new CipherInputStream(in, cipher);

byte[] buf = new byte[1024];

int numRead = 0;

while ((numRead = in.read(buf)) >= 0) {

    out.write(buf, 0, numRead);

}

out.close();

}

public static void copy(int mode, String inputFile, String outputFile, String password) throws
Exception {

    BufferedInputStream is = new BufferedInputStream(new FileInputStream(inputFile));

    BufferedOutputStream os = new BufferedOutputStream(new FileOutputStream(outputFile));

    if(mode==Cipher.DECRYPT_MODE){

        decrypt(is, os, password);

    }

    is.close();

    os.close();

}

static public void zipFolder(String srcFolder, String destZipFile) throws Exception {

```

```
ZipOutputStream zip = null;

FileOutputStream fileWriter = null;

fileWriter = new FileOutputStream(destZipFile);

zip = new ZipOutputStream(fileWriter);

addFolderToZip("", srcFolder, zip);

zip.flush();

zip.close();

}

static private void addFileToZip(String path, String srcFile, ZipOutputStream zip)
throws Exception {

File folder = new File(srcFile);

if (folder.isDirectory()) {

addFolderToZip(path, srcFile, zip);

} else {

byte[] buf = new byte[1024];

int len;

FileInputStream in = new FileInputStream(srcFile);

zip.putNextEntry(new ZipEntry(path + "/" + folder.getName()));

while ((len = in.read(buf)) > 0) {

zip.write(buf, 0, len);

}

}

}

static private void addFolderToZip(String path, String srcFolder, ZipOutputStream zip)
throws Exception {
```



```
File folder = new File(srcFolder);  
for (String fileName : folder.list()) {  
    if (path.equals("")) {  
        addFileToZip(folder.getName(), srcFolder + "/" + fileName, zip);  
    } else {  
        addFileToZip(path + "/" + folder.getName(), srcFolder + "/" + fileName, zip);  
    }  
}}}
```

4. Registration Code

```
package org.eclipse.equinox.launcher;  
  
import java.awt.*;  
  
import java.awt.event.*;  
  
import javax.swing.*;  
  
import java.io.File;  
  
import java.io.IOException;  
  
import javax.imageio.ImageIO;  
  
import java.sql.Connection;  
  
import java.sql.DriverManager;  
  
import java.sql.ResultSet;  
  
import java.sql.PreparedStatement;  
  
import java.sql.SQLException;  
  
import java.sql.Statement;  
  
public class Register extends JFrame  
  
{
```

```
private static final long serialVersionUID = 7526472295622776147L;

private JLabel jLabel1;

private JLabel jLabel2;

private JLabel jLabel3;

private JTextField jTextField1;

private JPasswordField jPasswordField1;

private JPasswordField jconfirmPasswordField1;

private JButton jButton1;

private JButton jButton2;

private JPanel contentPane;

public Register()

{

super();

create();

this.setVisible(true);

}

private void create()

{

jLabel1 = new JLabel();

jLabel2 = new JLabel();

jLabel3 = new JLabel();

jTextField1 = new JTextField();

jPasswordField1 = new JPasswordField();

jconfirmPasswordField1 = new JPasswordField();

jButton1 = new JButton();
```

```
jButton2 = new JButton();

contentPane = (JPanel)this.getContentPane();

jLabel1.setHorizontalAlignment(SwingConstants.LEFT);

jLabel1.setForeground(new Color(0, 0, 255));

jLabel1.setText("Username:");

jLabel2.setHorizontalAlignment(SwingConstants.LEFT);

jLabel2.setForeground(new Color(0, 0, 255));

jLabel2.setText("Password:");

jLabel3.setHorizontalAlignment(SwingConstants.LEFT);

jLabel3.setForeground(new Color(0, 0, 255));

jLabel3.setText("Confirm Password:");

jTextField1.setForeground(new Color(0, 0, 255));

jTextField1.setSelectedTextColor(new Color(0, 0, 255));

jTextField1.setToolTipText("Enter your username");

jTextField1.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e)

    {

        jTextField1_actionPerformed(e);

    }

});

jPasswordField1.setForeground(new Color(0, 0, 255));

jPasswordField1.setToolTipText("Enter your password");

jPasswordField1.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e)

    {
```

```
jPasswordField1_actionPerformed(e);
}
});

jconfirmPasswordField1.setForeground(new Color(0, 0, 255));
jconfirmPasswordField1.setToolTipText("Enter your password again");
jconfirmPasswordField1.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jPasswordField1_actionPerformed(e);
}
});

jButton1.setBackground(new Color(204, 204, 204));
jButton1.setForeground(new Color(0, 0, 255));
jButton1.setText("OK");
jButton1.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jButton1_actionPerformed(e);
}
});

jButton2.setBackground(new Color(204, 204, 204));
jButton2.setForeground(new Color(0, 0, 255));
jButton2.setText("Exit");
```

```
        jButton2.addActionListener(new ActionListener() {  
            public void actionPerformed(ActionEvent e)  
            {  
                jButton2_actionPerformed(e);  
            }  
        });  
  
        contentPane.setLayout(null);  
  
        contentPane.setBorder(BorderFactory.createEtchedBorder());  
  
        contentPane.setBackground(new Color(204, 204, 204));  
  
        addComponent(contentPane, jLabel1, 10,10,106,18);  
        addComponent(contentPane, jLabel2, 10,47,97,18);  
        addComponent(contentPane, jLabel3, 10,84,110,18);  
        addComponent(contentPane, jTextField1, 125,10,183,22);  
        addComponent(contentPane, jPasswordField1, 125,45,183,22);  
        addComponent(contentPane, jconfirmPasswordField1, 125,80,183,22);  
        addComponent(contentPane, jButton1, 125,120,83,28);  
        addComponent(contentPane, jButton2, 225,120,83,28);  
  
        this.setTitle("Registration");  
  
        this.setLocation(new Point(76, 182));  
  
        this.setSize(new Dimension(350,190));  
  
        this.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);  
  
        this.setResizable(false);  
    }  
  
    public static Connection getConnection() throws Exception {  
        String driver = "org.gjt.mm.mysql.Driver";
```

```
String url = "jdbc:mysql://localhost:3306/test";

String username = "root";

String password = "1234";

Connection conn = null;

Class.forName(driver);

try {

    conn = DriverManager.getConnection(url, username, password);

} catch (SQLException e) {

    System.out.println("ERROR: Unable to Connect to Database.");

}

return conn;

}

private void addComponent(Container container,Component c,int x,int y,int width,int height)

{

    c.setBounds(x,y,width,height);

    container.add(c);

}

private void jTextField1_actionPerformed(ActionEvent e)

{

}

private void jPasswordField1_actionPerformed(ActionEvent e)

{

}
```

```
private void jconfirmPasswordField1_actionPerformed(ActionEvent e)
{
}

private void jButton3_actionPerformed(ActionEvent e)
{
}

private void jButton2_actionPerformed(ActionEvent e)
{
System.exit(1);
}

private void jButton1_actionPerformed(ActionEvent e){

System.out.println("\njButton1_actionPerformed(ActionEvent e) called.");

String Value1 = new String(jTextField1.getText());

//@SuppressWarnings("deprecation")

String Value2 = new String(jPasswordField1.getText());

String Value3 = new String(jconfirmPasswordField1.getText());

if (Value2.equals(Value3))
{
try{

Class.forName("com.mysql.jdbc.Driver");

Connection con =

DriverManager.getConnection("jdbc:mysql://localhost:3306/test","root","1234");
```

```
PreparedStatement stmt = con.prepareStatement("INSERT INTO login(username,password)
VALUES (?,?)");

stmt.setString(1, Value1);

stmt.setString(2, Value2);

//stmt.setString(3, "Lee");

stmt.executeUpdate();

Window w = SwingUtilities.getWindowAncestor(contentPane);

w.setVisible(false);

}

catch (Exception esp) {

esp.printStackTrace();

}

}

else

{

JOptionPane.showMessageDialog(null,"Error: Password Does not Match. Please retry");

}};
```


Bibliography

- [1] T. Proesbsting, S. Watterson, and Krakatoa. “Decompilation in java (does bytecode reveal source?)” In Proceedings of the 3rd USENIX Conference on Object-Oriented Technologies and Systems, pp. 185–197, Portland, Oregon, 1997.
- [2] Guy-Armand Yandji, Lui Lian Hao, Amir-Eddine Youssouf, Jules Ehoussou “Research on a normal file encryption and decryption” The International Conference on Computer and Management (CAMAN),Wuhan,China,19-21 May 2011.
- [3] “Eclipse (Software)” http://en.wikipedia.org/wiki/Eclipse_%28software%29 [visited: Jan 2015].
- [4] Jan M. Memon, Shams-ul-Arfeen, Asghar Mughal, Faisal Memon “Preventing Reverse Engineering Threat in Java Using Byte Code Obfuscation Techniques” 2nd International Conference on Emerging Technologies (IEEE—ICET 2006), Peshawar, Pakistan 13-14 November 2006.
- [5] S.K. Udupa, S.K. Debray, and M. Madou, Deobfuscation: “Reverse Engineering Obfuscated Code” Proceedings of the 12th Working Conference on Reverse Engineering (WCRE’05), 2005, pp. 45-54.
- [6] M. Madou, B. Anckaert, B.D. Bus, K.D. Bosschere, J.Cappaert, and B. Preneel, “On the Effectiveness of Source Code Transformations for Binary Obfuscation” Proceedings of the International Conference on Software Engineering Research and Practice (SERP06), CSREA Press, 2006, pp. 527-533.
- [7] “Can We Obfuscate Programs?” http://www.math.ias.edu/~boaz/Papers/obf_informal.html [visited: Jan 2015].
- [8] Jien-Tsai Chan *, Wu Yang “Advanced obfuscation techniques for Java bytecode” The Journal of Systems and Software 71, (2004), pp. 1–10.
- [9] Guy-Armand Yandji, Lui Lian Hao, Amir-Eddine Youssouf, Jules Ehoussou “Research on a normal file encryption and decryption” The International Conference on Computer and Management (CAMAN),Wuhan,China,19-21 May 2011.

- [10] X. Zhang, Q. Wen, “AOP-Based J2EE Source Code Protection” International Conference on Computational Intelligence and Security Workshops, Harbin, Heilongjiang, China, 2007, pp. 581-584.
- [11] ByungRae Cha “CRYPTEX Model for Software Source Code” International Conference on Information Security and Assurance, Busan, Korea , 24-26 April 2008.
- [12] Layer 7 Technologies “Protecting Your APIs Against Attack & Hijack” http://docs.media.bitpipe.com/io_11x/io_113161/item_826348/8%20%20Protecting%20Your%20APIs%20Against%20Attack%20and%20Hijack.pdf [visited: May 2014]
- [13] “Subversion” <http://svnbook.red-bean.com/en/1.6/svn.intro.whatis.html> [visited: Jan 2015]
- [14] “CVS” http://en.wikipedia.org/wiki/Concurrent_Versions_System [visited: Jan 2015]
- [15] “RESEARCH METHODOLOGY” http://www.mech.hku.hk/bse/bbse3002/Research_Methodology.pdf [visited: Feb 2015]
- [16] “What is Research?” <http://www.smccd.edu/accounts/csmlibrary/tutorials/what.html> [visited: Feb 2015]
- [17] “Method or methodology, what’s the difference?” <http://whanauoraresearch.co.nz/news/method-or-methodology-whats-the-difference/> [visited: Feb 2015]
- [18] “Applied Research” <http://examples.yourdictionary.com/examples-of-applied-research.html> [visited: Feb 2015]
- [19] “What is qualitative research” <http://www.qsrinternational.com/what-is-qualitative-research.aspx>[visited: Feb 2015]
- [20] “What is quantitative research?” <http://www.marketingdonut.co.uk/marketing/market-research/what-is-quantitative-research-> [visited: Feb 2015]
- [21] “Basic research” http://en.wikipedia.org/wiki/Basic_research - [visited: Feb 2015]

- [22] “DESCRIPTIVE RESEARCH” http://www.researchproposalsforhealthprofessionals.com/descriptive_research1.htm [visited: March 2015]
- [23] Eclipse Source Code Tutorial” <http://www.vogella.com/tutorials/EclipseCodeAccess/article.html> [visited: March 2015]