A

Major Project

on

SECURITY THREATS TO MOBILE MULTIMEDIA APPLICATIONS

(Submitted in partial fulfillment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

CH.Phani(187R1A05D2)

Under the Guidance of

J.SRIVIDYA
(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi) Recognized Under Section 2(f) & 12(B) of the UGCAct.1956, Kandlakoya (V), Medchal Road, Hyderabad-501401.

2018-22

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled "**SECURITY THREATS TO MOBILE MULTIMEDIA APPLICATIONS**" is being submitted by **CH.phani(187R1A05D2),**in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by him/her under our guidance and supervision during the year 2021-22.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

J. Srividya
(Assistant Professor)
INTERNAL GUIDE

Dr. A. Raji Reddy
DIRECTOR

Dr. K. Srujan Raju
HOD

EXTERNAL EXAMINER

Submitted for viva voice Examination held on _____

ACKNOWLEDGMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project. We take this opportunity to express my profound gratitude and deep regard to my guide

J.SRIVIDYA, Assistant Professor for her exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by her shall carry us a long way in the journey of life on which we are about to embark. We also take this opportunity to express a deep sense of gratitude to Project Review Committee (PRC) **Dr. M. Varaprasad Rao, Mr. J. Narasimha Rao, Dr. T. S. MastanRao, Dr. Suwarna Gothane, Mr. A. Uday Kiran, Mr. A. Kiran Kumar, Mrs. G. Latha** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Srujan Raju,** Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy,** Director for being cooperative throughout the course of this project. We also express our sincere gratitude to Sri. **Ch. Gopal Reddy,** Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

CH PHANI(187R1A05D2)
D ANIKETH(187R1A05D9)
B ROHITNIVAS(187R1A05D4)

ABSTRACT

Today's mobile smartphones are very powerful, and many smartphone applications use wireless multimedia communications. Mobile phone security has become an important aspect of security issues in wireless multimedia communications. As the most popular mobile operating system, Android security has been extensively studied by researchers. However, few works have studied mobile phone multimedia security. In this article, we focus on security issues related to mobile phone cameras. Specifically, we discover several new attacks that are based on the use of phone cameras. We implement the attacks on real phones, and demonstrate the feasibility and effectiveness of the attacks. Furthermore, we propose a lightweight defense scheme that can effectively detect these attacks.

LIST OF FIGURES

LIST OF SCREENSHOTS

LIST OF CONTENT

1.INTRODUCTION

CHAPTER 1
INTRODUCTION

1.1 GENERAL:

Since 2007, the Android operating system (OS) has enjoyed an incredible rate of popularity. As of 2013, the Android OS holds 79.3 percent of global smartphone market shares. Meanwhile, a number of Android security and privacy vulnerabilities have been exposed in the past several years. Although the Android permission system gives users an opportunity to check the permission request of an application (app) before installation, few users have knowledge of what all these permission requests stand for; as a result, they fails to warn users of security risks. Meanwhile, an increasing number of apps specified to enhance security and protect user privacy have appeared in Android app markets. Most large anti-virus software companies have published their Android-version security apps, and tried to provide a shield for smartphones by detecting and blocking malicious apps. In addition, there are data protection apps that provide users the capability to encrypt, decrypt, sign, and verify signatures for private texts, emails, and files. However, mobile malware and privacy leakage remain a big threat to mobile phone security and privacy.

Generally, when talking about privacy protection, most smartphone users pay attention to the safety of SMS, emails, contact lists, calling histories, location information, and private files. They may be surprised that the phone camera could become a traitor; for example, attackers could stealthily take pictures and record videos by using the phone camera. Nowadays, various types of camera-based applications have appeared in Android app markets (photography, barcode readers, social networking, etc.). Spy camera apps have also become quite popular. As for Google Play, there are nearly 100 spy camera apps, which allow phone users to take pictures or record videos of other people without their permission. However, believe it or not, phone users themselves could also become victims.

Attackers can implement spy cameras in malicious apps such that the phone camera is launched automatically without the device owner's notice, and the captured photos and videos are sent out to these remote attackers. Even worse, according to a survey on Android malware analysis [1], camera permission ranks 12th of the most commonly requested permissions among

benign apps, while it is out of the top 20 in malware. The popularity of camera usage in benign apps and relatively less usage in malware lower users' alertness to camera-based multimedia application attacks.

1.2 OBJECTIVE:

Nowadays, people carry their phones everywhere; hence, their phones see lots of private information. If the phone camera is exploited by a malicious spy camera app, it may cause serious security and privacy problems. For example, the phone camera may record a user's daily activities and conversations, and then send these out via the Internet or multimedia messaging service (MMS). Secret photography is not only immoral but also illegal in some countries due to the invasion of privacy. Nevertheless, a phone camera could also provide some benefits if it is con-trolled well by the device owner. For example, when the owner wants to check if someone has used his/her phone without permission, the phone camera could be used to record the face of an unauthorized user. Besides, it can also help the owner find a lost phone.

In this article, we first conduct a survey on the threats and benefits of spy cameras. Then we present the basic attack model and two camera based attacks: the remote-controlled real-time monitoring attack and the passcode inference attack. We run these attacks along with popular antivirus software to test their stealthiness, and conduct experiments to evaluate both types of attacks. The results demonstrate the feasibility and effectiveness of these attacks. Finally, we propose a lightweight defense scheme.

1.3 EXISTING SYSTEM:

Several video-based attacks targeted at keystrokes have been proposed. The attacks can obtain user input on touch screen smart phones. Maggi *et al.* implement an automatic shoulder surfing attack against modern touch-enabled smart phones. The attacker deploys a video camera that can record the target screen while the victim is entering text. Then user input can be reconstructed solely based on the keystroke feedback displayed on the screen.

1.3.1 DRAWBACKS IN EXISTING SYSTEM:

- It works only when visual feedback such as magnified keys are available.

- Mobile malware and privacy leakage remain a big threat to mobile phone security and privacy

1.3.2 LITERATURE SURVEY:

TITLE : Dissecting Android Malware: Characterization and Evolution
AUTHOR : Y. Zhou and X. Jiang,
YEAR : 2012

DESCRIPTION

The popularity and adoption of smart phones has greatly stimulated the spread of mobile malware, especially on the popular platforms such as Android. In light of their rapid growth, there is a pressing need to develop effective solutions. However, our defense capability is largely constrained by the limited understanding of these emerging mobile malware and the lack of timely access to related samples. In this paper, we focus on the Android platform and aim to systematize or characterize existing Android malware. Particularly, with more than one year effort, we have managed to collect more than 1,200 malware samples that cover the majority of existing Android malware families, ranging from their debut in August 2010 to recent ones in October 2011. In addition, we systematically characterize them from various aspects, including their installation methods, activation mechanisms as well as the nature of carried malicious payloads. The characterization and a subsequent evolution-based study of representative families reveal that they are evolving rapidly to circumvent the detection from existing mobile anti-virus software. Based on the evaluation with four representative mobile security software, our experiments show that the best case detects 79.6% of them while the worst case detects only 20.2% in our dataset. These results clearly call for the need to better develop next-generation anti-mobile-malware solutions.

TITLE : A Fast Eavesdropping Attack against Touchscreens.

AUTHOR : F. Maggi, et

YEAR : 2011.

DESCRIPTION

The pervasiveness of mobile devices increases the risk of exposing sensitive information on the go. In this paper, we arise this concern by presenting an automatic attack against modern touchscreen keyboards. We demonstrate the attack against the Apple iPhone - 2010's most popular touchscreen device - although it can be adapted to other devices (e.g., Android) that employ similar key-magnifying keyboards. Our attack processes the stream of frames from a video camera (e.g., surveillance or portable camera) and recognizes keystrokes online, in a fraction of the time needed to perform the same task by direct observation or offline analysis of a recorded video, which can be unfeasible for large amount of data. Our attack detects, tracks, and rectifies the target touchscreen, thus following the device or camera's movements and eliminating possible perspective distortions and rotations In real-world settings, our attack can automatically recognize up to 97.07 percent of the keystrokes (91.03 on average), with 1.15 percent of errors (3.16 on average) at a speed ranging from 37 to 51 keystrokes per minute.

TITLE : ispy: Automatic Reconstruction of Typed Input from Compromising Reflections

AUTHOR : R. Raguram et al.,

YEAR : 2011.

DESCRIPTION

We investigate the implications of the ubiquity of personal mobile devices and reveal new techniques for compromising the privacy of users typing on virtual keyboards. Specifically, we show that so-called compromising reflections (in, for example, a victim's sunglasses) of a device's screen are sufficient to enable automated reconstruction, from video, of text typed on a virtual keyboard. Despite our deliberate use of low cost commodity video cameras, we are able to compensate for variables such as arbitrary camera and device positioning and motion through the application of advanced computer vision and machine learning techniques. Using footage captured in realistic environments (e.g., on a bus), we show that we are able to reconstruct fluent translations of recorded data in almost all of the test cases, correcting users' typing mistakes at the same time. We believe these results highlight the importance of adjusting privacy expectations in response to emerging technologies.

TITLE : Defending Web Services Against Denial of Service Attacks Using Client Puzzles

AUTHOR : Suriadi Suriadi, Douglas Stebila, Andrew Clark, and Hua Liu_

YEAR : 2010.

DESCRIPTION

The interoperable and loosely-coupled web services architecture, while beneficial, can be resource-intensive, and is thus susceptible to denial of service (DoS) attacks in which an attacker can use a relatively insignificant amount of resources to exhaust the computational resources of a web service. We investigate the effectiveness of defending web services from DoS attacks using client puzzles, a cryptographic countermeasure which provides a form of gradual authentication by requiring the client to solve some computationally difficult problems before access is granted. In particular, we describe a mechanism for integrating a hash-based puzzle into existing web services frameworks and analyze the effectiveness of the countermeasure using a variety of scenarios on a network testbed. Client puzzles are an effective defence against flooding attacks. They can also mitigate certain types of semantic-based attacks, although they may not be the optimal solution.

TITLE : A Client-Transparent Approach to Defend Against Denial of Service Attacks

AUTHOR : Mudhakar Srivatsay, Arun Iyengarz, Jian Yinz and Ling Liuy

YEAR : 2012

DESCRIPTION

Denial of Service (DoS) attacks attempt to consume a server's resources (network bandwidth, computing power, main memory, disk bandwidth etc) to near exhaustion so that there are no resources left to handle requests from legitimate clients.

In this paper, we propose a light-weight client transparent technique to defend against DoS attacks with two unique features: (i) Our technique can be implemented entirely using JavaScript support provided by a standard client-side browser like Mozilla FireFox or Microsoft Internet Explorer. Client transparency follows from the fact that: (i) no changes to client-side software are required, (ii) no client-side superuser privileges are required, and (iii) clients (human beings or automated clients) can browse a DoS protected website in the same manner that they browse other websites. (ii) Although we operate using the client-side browser (HTTP layer), our technique enables fast IP level packet _ltering at the server's _rewall and requires no changes to the application(s) hosted by the web server.

1.1 PROPOSED SYSTEM:

In this article, we first conduct a survey on the threats and benefits of spy cameras. Then we present the basic attack model and two camera based attacks: the remote-controlled real-time monitoring attack and the passcode inference attack. We run these attacks along with popular antivirus software to test their stealthiness, and conduct experiments to evaluate both types of attacks. The results demonstrate the feasibility and effectiveness of these attacks.

1.4.1 ADVANTAGES IN PROPOSED SYSTEM:

- The attacker needs considerable effort in translating central processing unit puzzle software to its functionally equivalent GPU version such that the translation cannot be done in real time.
- Moreover, we show how to implement puzzle in the generic server-browser model. To outsourcing any business onto a cloud.
- By using this Applications, we can easily be avoided by selecting the time to launch attack.
- The malicious camera app can periodically check the screen status and run the stealthy video recording only when the screen is off, which means that the user is not using the phone and the camera device is idle.

2.SYSTEM ANALYSIS

CHAPTER 2
SYSTEM ANALYSIS

2.1 GENERAL

The seriousness of the DoS/DDoS problem and their increased frequency has led to the advent of numerous defense mechanisms. In this paper, we are particularly interested in the countermeasures to DoS/DDoS attacks on server computation power. DoS and DDoS are effective if attackers spend much less resources than the victim server or are much more powerful than normal users.

There are five modules for the Software Puzzle.

2.2 METHODOLOGIES

Following modules involves

2.2.1 MODULES

- USER INTERFACE DESIGN
- GPU-INFLATED DOS ATTACK
- PUZZLE GENERATION
- CODE PROTECTION
- SECURITY ANALYSIS

- **User Interface Design:**

This is the first module of our project. The important role for the cloud user is to move login window to cloud user window. This module has created for the security purpose. In this login page we have to enter login user id and password. It will check username and password is match or not (valid user id and valid password). If we enter any invalid username or password we can't

enter into login window to user window it will shows error message. So we are preventing from unauthorized user entering into the login window to user window. It will provide a good security for our project. So server contain user id and password server also check the authentication of the user. It well improves the security and preventing from unauthorized user enters into the network. In our project we are using JSP for creating design. Here we validate the login user and server authentication.

- **GPU-Inflated Dos Attack:**

In order to elaborate software puzzle, we recap its rival GPU-inflated DoS attack in advance. When a client wants to obtain a service, she sends a request to the server. After receiving the client request, the server responds with a puzzle challenge *x*. If the client is genuine, she will find the puzzle solution *y* directly on the host CPU, and send the response *(x, y)* to the server. However, as shown in Fig. 1, by using the similar mechanism in accelerating calculation with GPU, a malicious user who controls the host will send the challenge *x* to GPU and exploit the GPU resource to accelerate the puzzle-solving process.

Since the virtual keyboard in a touch screen smartphone is much smaller than computer keyboards, the virtual keys are very close to each other. Based on measurement of a Galaxy Nexus 4 phone, even an offset of 5 mm could result in touching the wrong key. Hence, when typing, users tend to keep a short distance to the screen, which allows the phone (front) camera to have a clear view of a user's eye movements. A user's eyes move along with the keys being touched, which means that tracking the eye movement could possibly tell what the user is entering. Thus, it is of great importance to investigate whether an attacker could obtain a phone user's passcode by tracking the eye movements.

- **Puzzle Generation:**

In order to construct a puzzle, the server has to execute three modules: puzzle core generation, puzzle challenge generation, puzzle encrypting/obfuscating.

1) **Puzzle Core Generation***:* From the code block warehouse, the server first chooses *n* code blocks based on hash functions and a secret, e.g., the *j* th instruction block *bi j* , where *i j = H*1*(y,*

j), and $y = H2(key, sn)$, with one-way functions $H1(\cdot)$ and $H2(\cdot)$, *key* is the server's secret, and *sn* is a nonce or timestamp. All the chosen blocks are assembled into a puzzle core, denoted as $C(\cdot) = (bi1 ; bi2 ; \cdots ; bin)$. As an illustrative example, Table III in the appendix shows an example puzzle core *C* generated from AES operation blocks stored in warehouse **S**.

2) **Puzzle Challenge Generation***:* Given some auxiliary input messages such as IP addresses, and in-line constants, the server calculates a message *m* from public data such as their IP addresses, port numbers and cookies, and produces a challenge $x = C(y,m)$, smiliar to encrypting plaintext *m* with key *y* to produce ciphertext *x*. As the attacker does not know the puzzle core $C(\cdot)$ (or equivalently the puzzle function $P(\cdot)$) in advance, it can not exploit GPU to solve the puzzle $C0x$ in real time using the basic GPU-inflated DoS attack addressed in Subsection III-A. Nonetheless, if the puzzle is merely constructed as aboveit is possible for an attacker to generate the GPU kernel by mapping the CPU instructions in $C0x$ to the GPU instructions one by one, i.e., to automatically translate the CPU software puzzle $C0x$ into its functionally equivalent GPU version.

- **Code Protection:**

Intuitively, code obfuscation is able to thwart the above translation threat to some extent. Though there are no generic obfuscation techniques which can prevent a patient and advanced hacker from understanding a program results in show that obfuscation does increase the cost of reverse-engineering. Thus, although code obfuscation may be not satisfactory in long-term software defense against hacking, it is suitable for fortifying software puzzles which demand a protection period of several seconds only.

A puzzle consists of instructs, and each instruction has a form (opCode, [operands]), where opCode indicates which operation (*e.g.*, addition, shift, jump) is, whilethe operands, varying with opCode, are the parameters (*e.g.*, target address of jump instruction) to complete the operations. As a popular obfuscation technology, code encryption technology treats software code as data string and encrypts both operand and opCode.

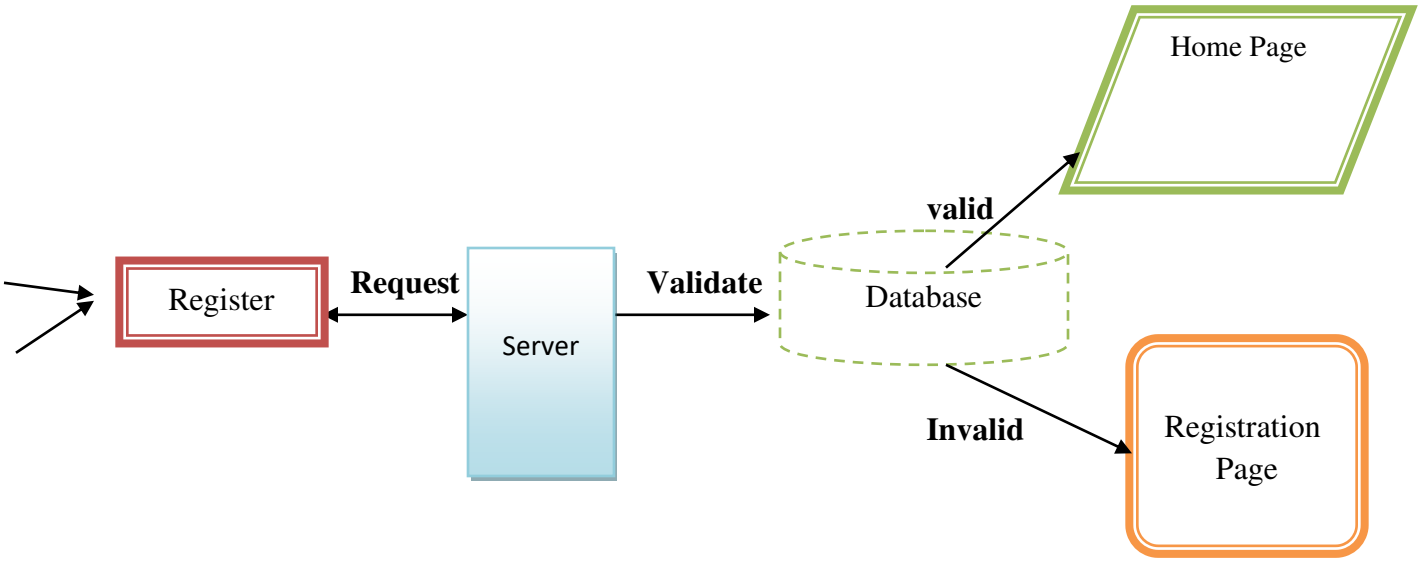
- **Security Analysis:**

In this module puzzle aims to prevent GPU from being used in the puzzle-solving process based on different instruction sets and real-time environments between GPU and CPU.

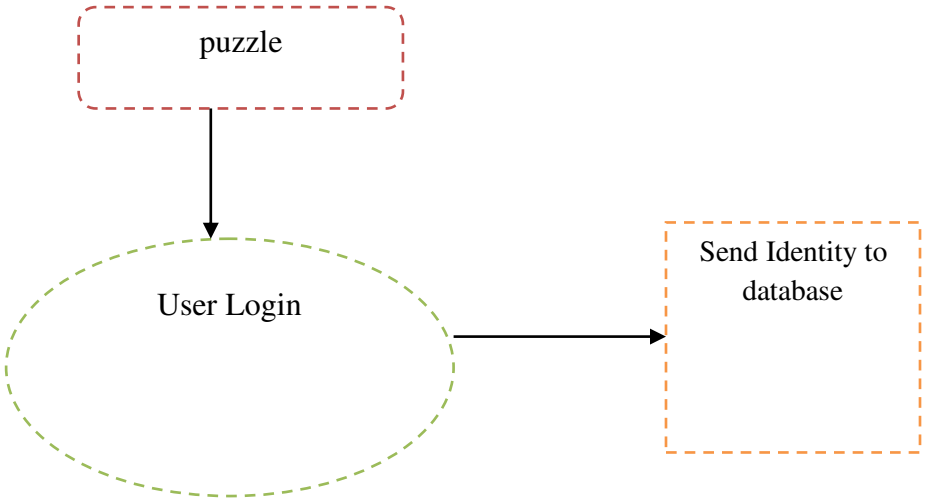
Conversely, an adversary may attempt to deface the puzzle scheme by simulating the host on GPU (Subsection V-A), cracking puzzle algorithm (Subsection V-B), re-producing GPU-version puzzle (Subsections V-C ∼ V-E), or abusing the access priority in puzzle-solving (Subsection V-F).

2.2.2 MODULE DIAGRAMS:

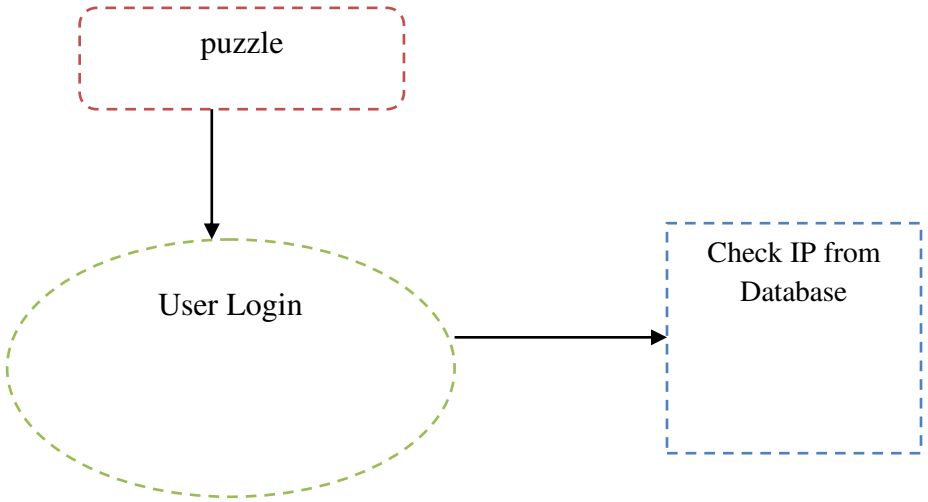
- **User Interface Design**



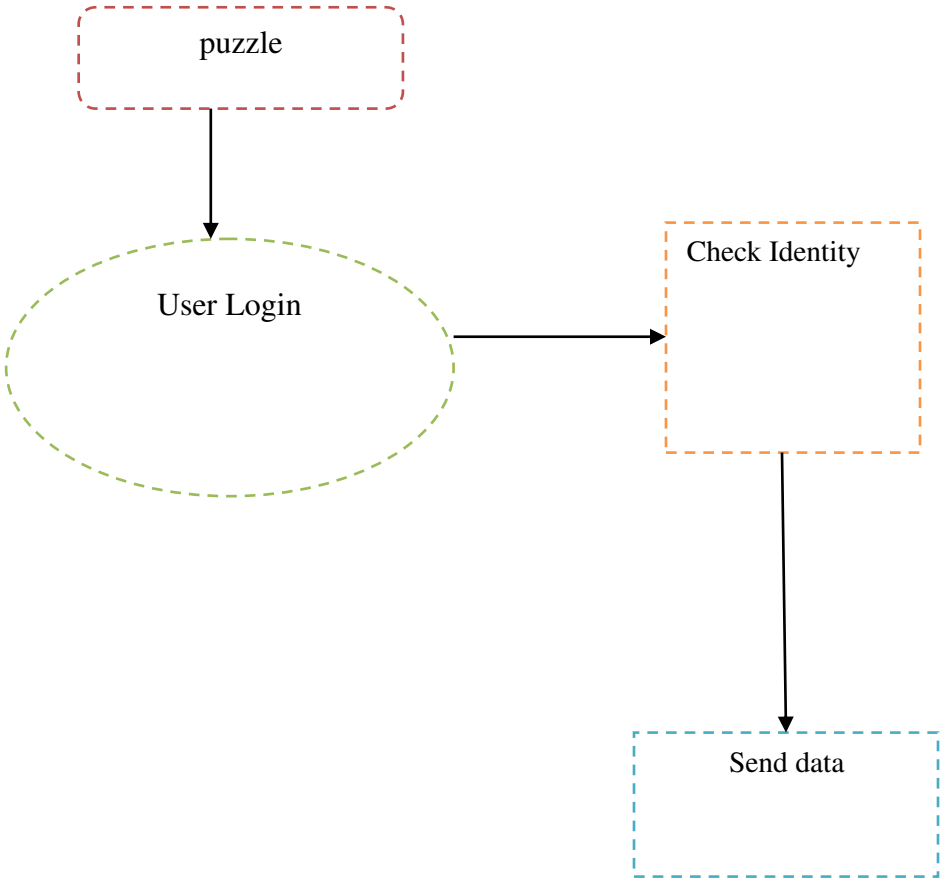
➢ **Gpu-Inflated Dos Attack**



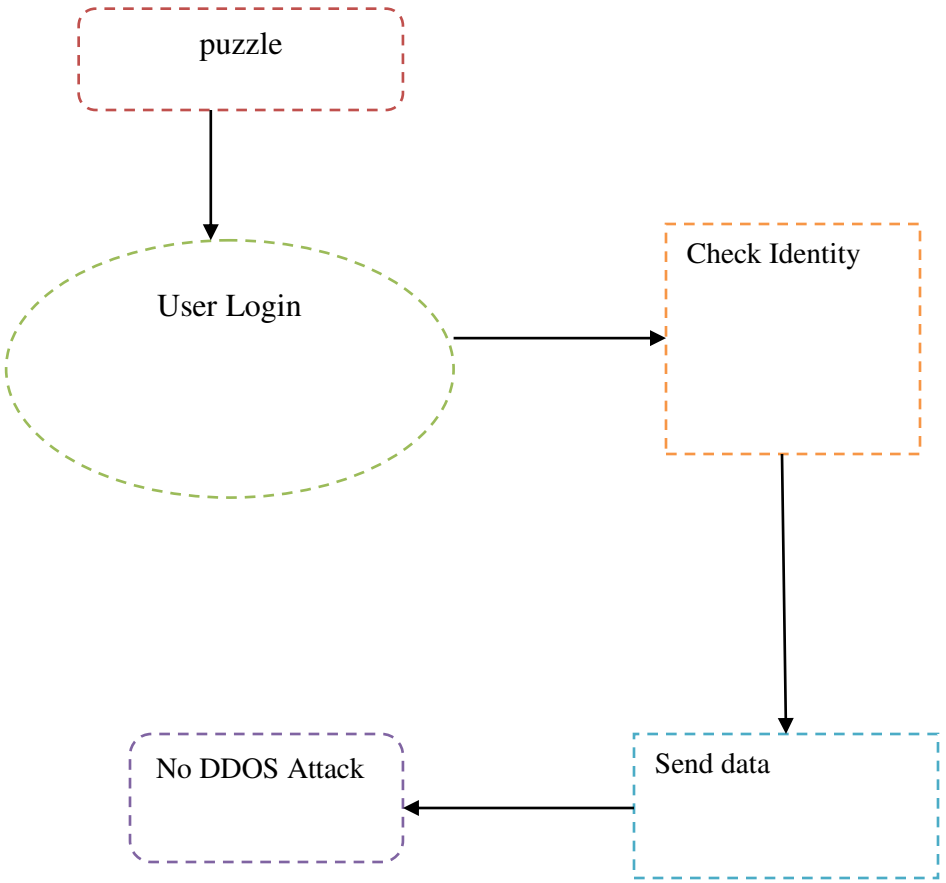
- **Puzzle Generation.**



- **Code Protection.**



- **Security Analysis**



2.3 SYSTEM TECHNIQUES:

Algorithm: Cracking Data Puzzle Algorithm

The practical strategy of the attacker is to accelerate the brute force process by exploiting the parallel computation capability of GPU cores. We classify client puzzles into two types. If a puzzle functions *P*, as all the existing client puzzle schemes, is fixed and disclosed in advance, the puzzle is called a data puzzle; otherwise, it is referred to as a software puzzle.

2.4 GENERAL

These are the requirements for doing the project. Without using these tools and software's we can't do the project. So we have two requirements to do the project. They are

1. Hardware Requirements.
2. Software Requirements.

2.5 HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It shows what the system does and not how it should be implemented.

PROCESSOR	:	PENTIUM IV 2.6 GHz, Intel Core 2 Duo.
RAM	:	512 MB DD RAM
MONITOR	:	15" COLOR
HARD DISK	:	40 GB

2.6 SOFTWARE REQUIREMENTS

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities,

performing tasks and tracking the team's and tracking the team's progress throughout the development activity.

Front End	:	JAVA SWING.
Back End	:	MY SQL 5.5
Operating System	:	Windows 07
IDE	:	Eclipse

3.ARCHICTECTURE

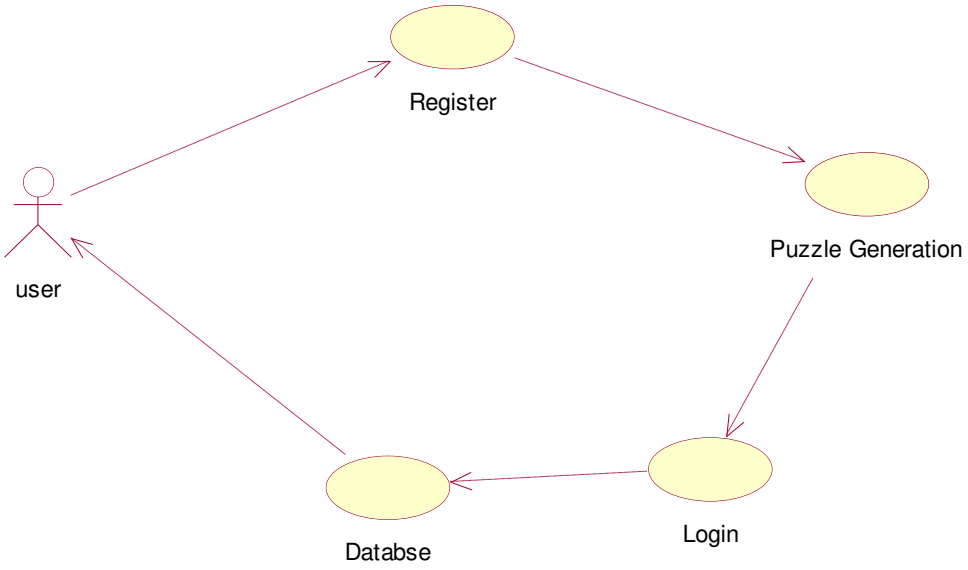
CHAPTER 3

ARCHITECTURE

3.1 GENERAL

Design Engineering deals with the various UML [Unified Modeling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into finished product.

3.1.2 Use Case Diagram:

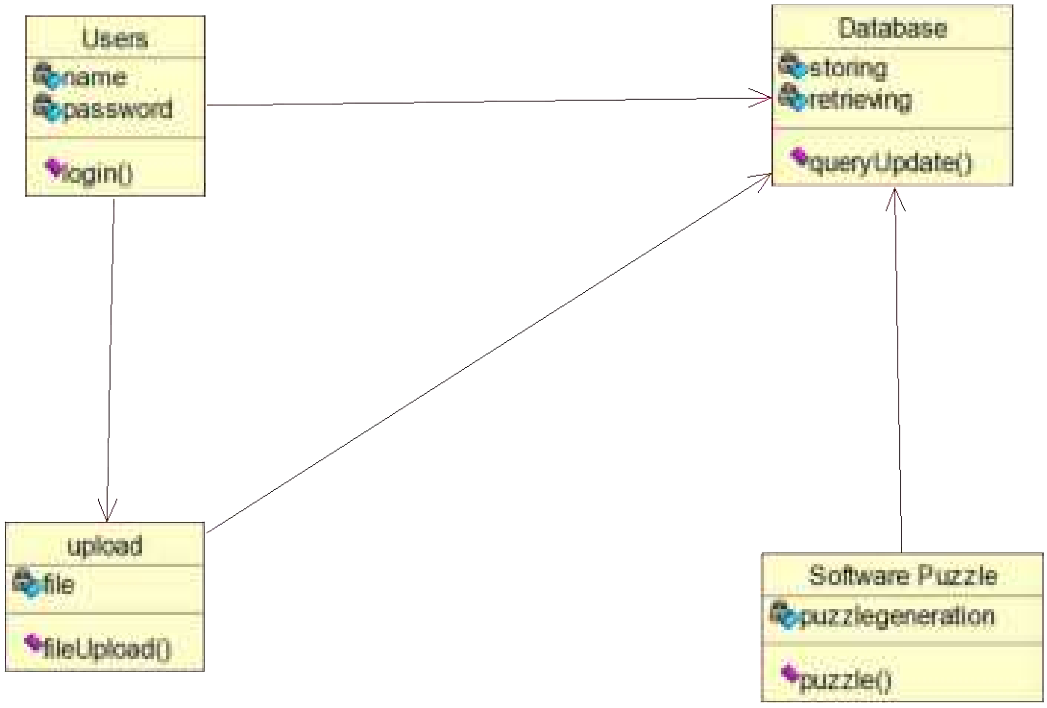


EXPLANATION:

The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. In our use case diagram first adopts software protection technologies to ensure challenge data confidentiality and code security for an appropriate time period**.** After receiving the puzzle sent from the server, a client

tries to solve the puzzle on the host CPU, and replies to the server, as the conventional client puzzle scheme does.

3.1.3 Class Diagram:

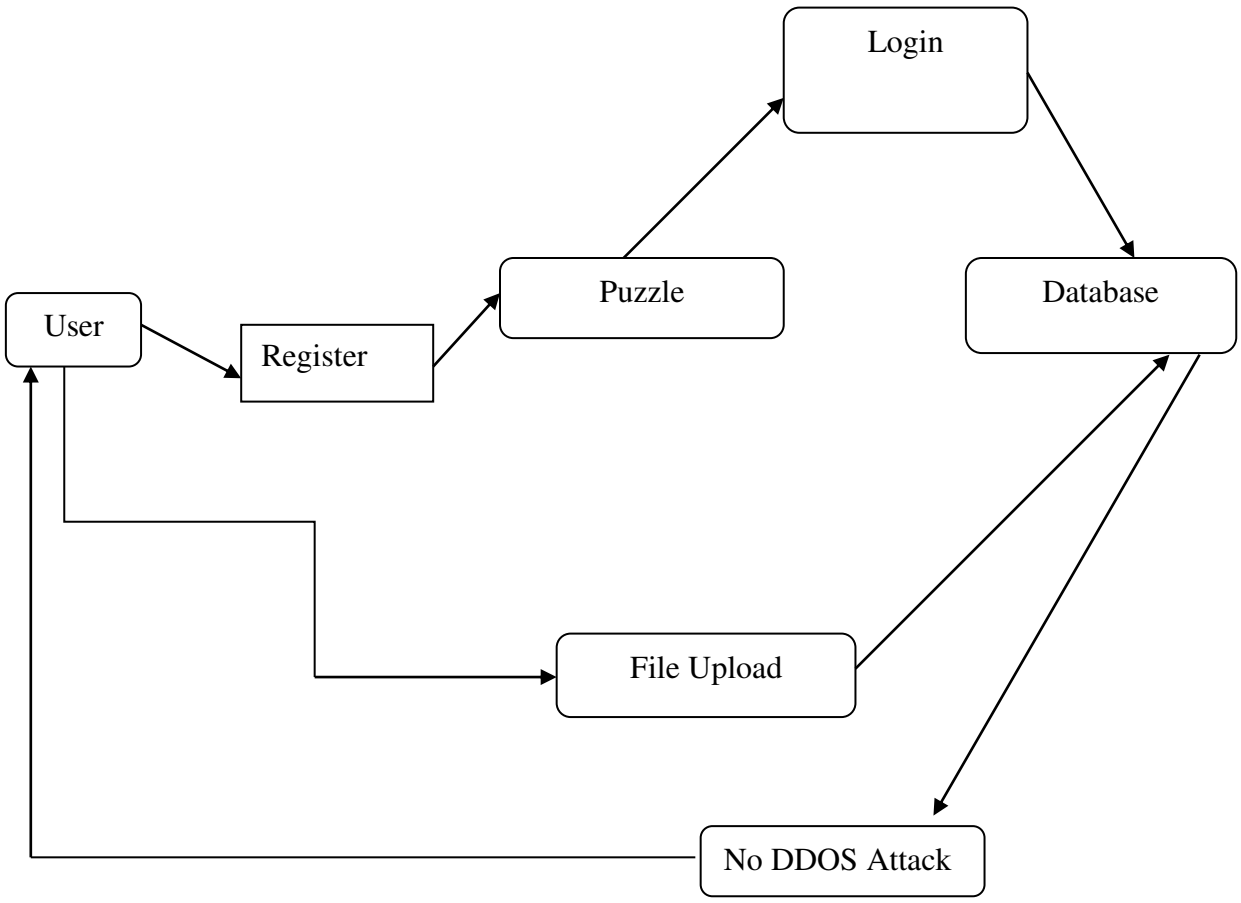


EXPLANATION:

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code.

In class diagram took the user, provider and consumer. It adopts software protection technologies to ensure challenge data confidentiality and code security for an appropriate time period**.** After receiving the puzzle sent from the server, a client tries to solve the software puzzle on the host CPU, and replies to the server, as the conventional client puzzle scheme does.

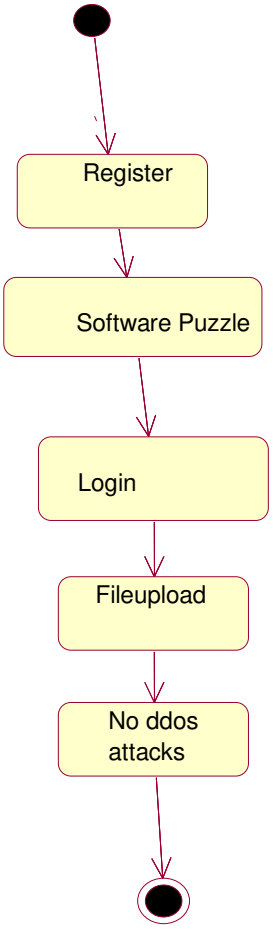
3.1.4 Object Diagram:



EXPLANATION:

Object diagram we are telling about the flow of objects how the process is running. In the above digram tells about the flow of objects between the classes.The main object of this diagram is to adopts software protection technologies to ensure challenge data confidentiality and code security for an appropriate time period**.** After receiving the puzzle sent from the server, a client tries to solve the puzzle on the host CPU, and replies to the server, as the conventional client puzzle scheme does.

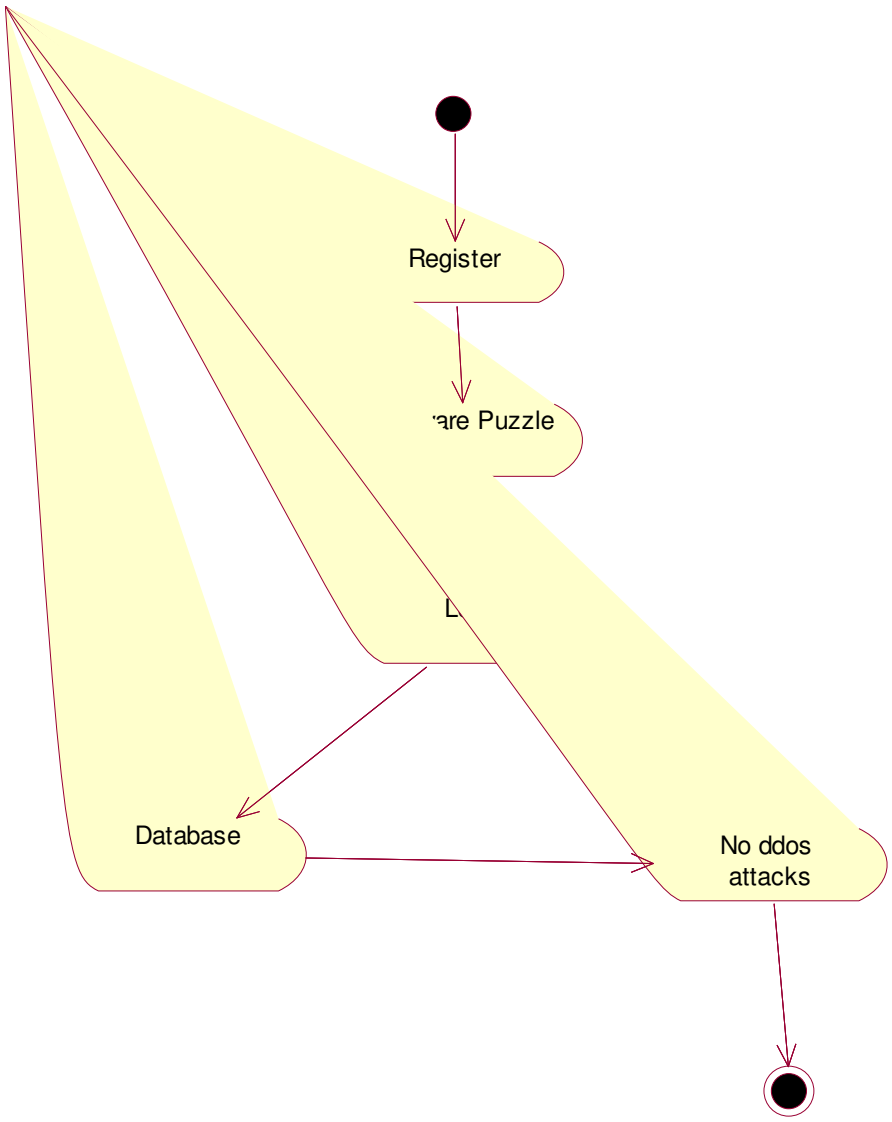
3.1.5 State Diagram:



EXPLANATION:

State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction. Many forms of state diagrams exist, which differ slightly and have different semantics. In our state diagram first to adopts software protection technologies to ensure challenge data confidentiality and code security for an appropriate time period**.** After receiving the puzzle sent from the server, a client tries to solve the puzzle on the host CPU, and replies to the server, as the conventional client puzzle scheme does.

3.1.6 Activity Diagram:

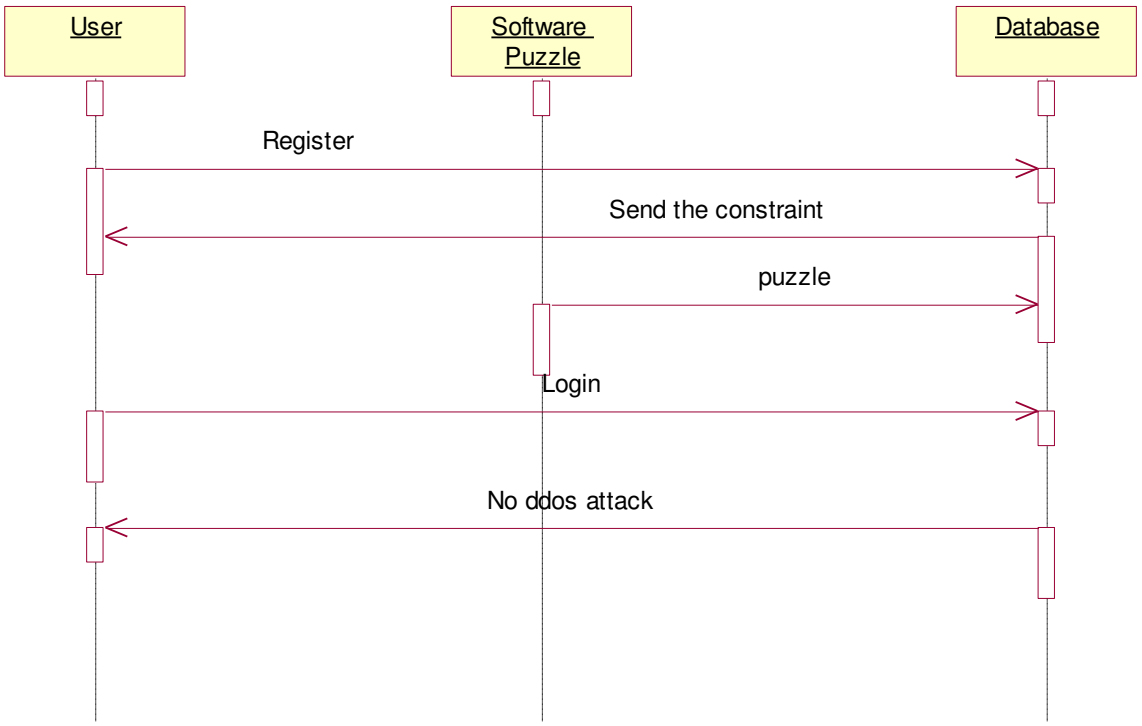


EXPLANATION:

In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control. In our activity diagram first to adopts software protection technologies to ensure challenge data confidentiality and code security for an appropriate time

period**.** After receiving the puzzle sent from the server, a client tries to solve the puzzle on the host CPU, and replies to the server, as the conventional client puzzle scheme does.

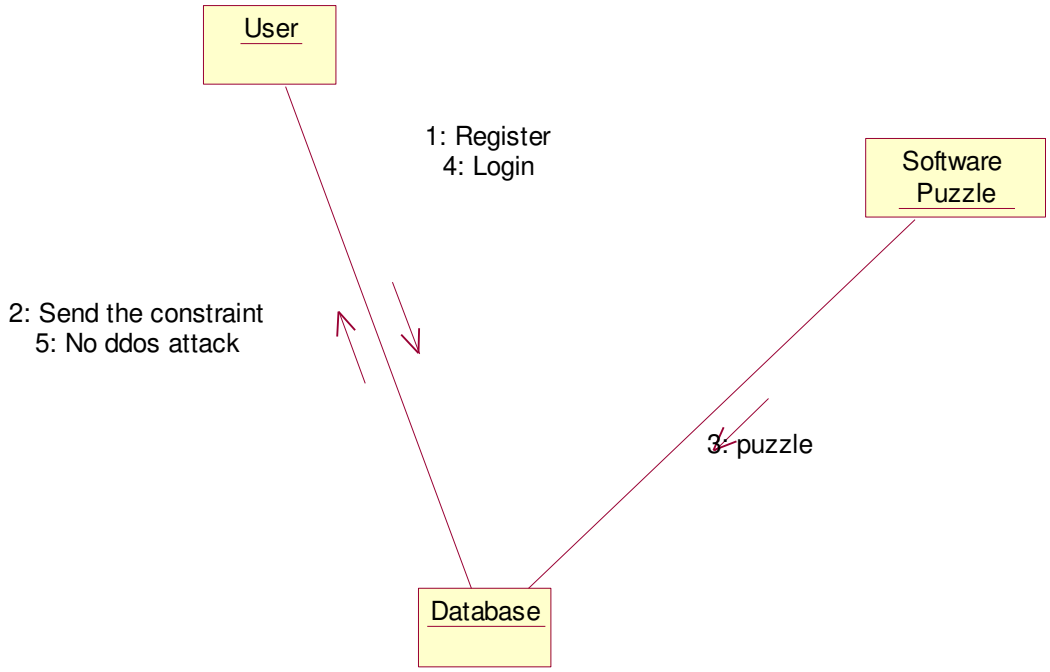
3.1.7 Sequence Diagram:



EXPLANATION:

In our sequence diagram specifying processes operate with one another and in order. In our sequence diagram first to adopts software protection technologies to ensure challenge data confidentiality and code security for an appropriate time period**.** After receiving the puzzle sent from the server, a client tries to solve the puzzle on the host CPU, and replies to the server, as the conventional client puzzle scheme does.

3.1.8 Collaboration Diagram:

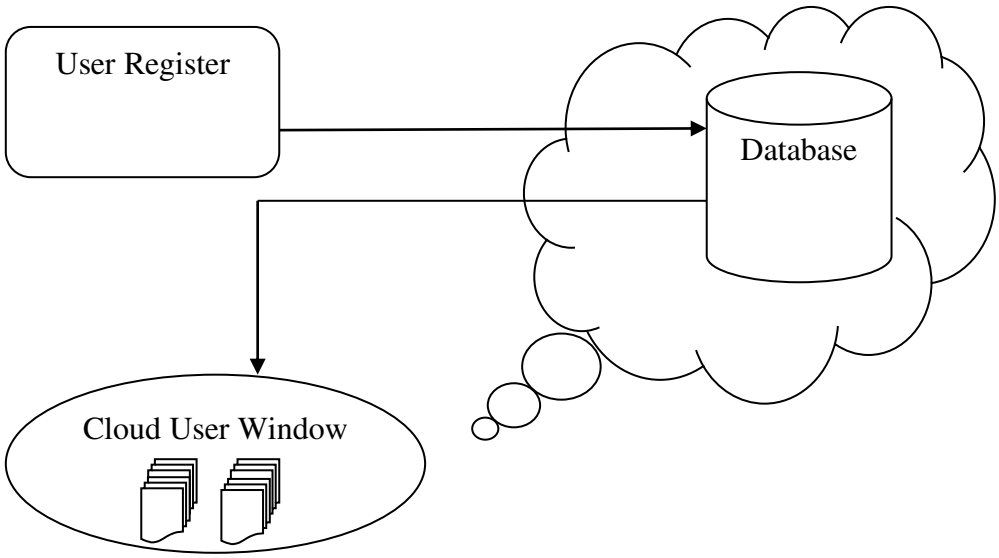


EXPLANATION:

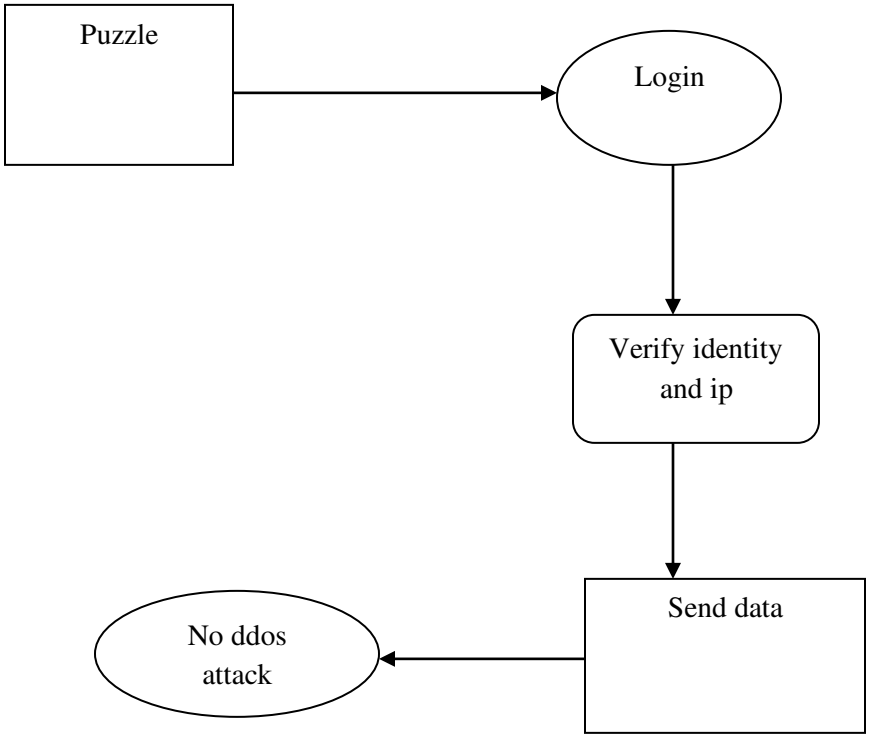
A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behavior of a system. In this diagram first to adopts software protection technologies to ensure challenge data confidentiality and code security for an appropriate time period**.** After receiving the puzzle sent from the server, a client tries to solve the puzzle on the host CPU, and replies to the server, as the conventional client puzzle scheme does.

3.1.9 Data Flow Diagram:

Level-0:



Level-1:

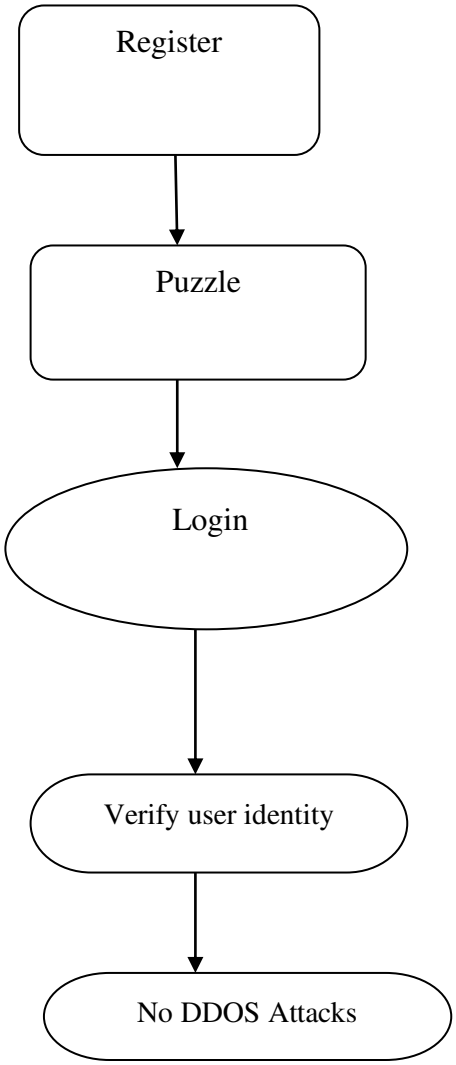


EXPLANATION:

It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel. In the DFDs the level zero process is based on the login validations. To adopt software protection technologies to ensure challenge data confidentiality and code security for an appropriate time period**.** After receiving the puzzle

sent from the server, a client tries to solve the puzzle on the host CPU, and replies to the server, as the conventional client puzzle scheme does.

3.1.10 E-R Diagram:

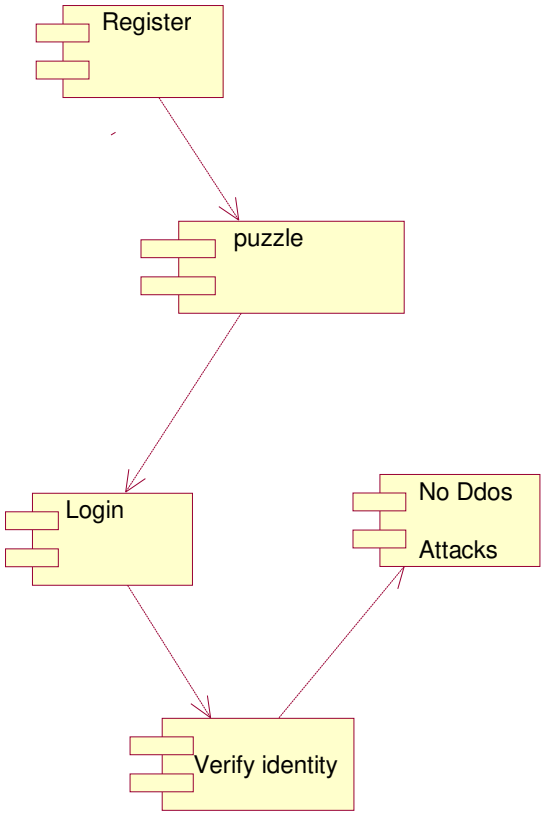


EXPLANATION:

Entity-Relationship Model (ERM) is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database. In our ER diagram to adopt software protection technologies to ensure challenge data confidentiality and code security for an appropriate time period**.** After receiving the puzzle sent from the server, a

client tries to solve the puzzle on the host CPU, and replies to the server, as the conventional client puzzle scheme does.

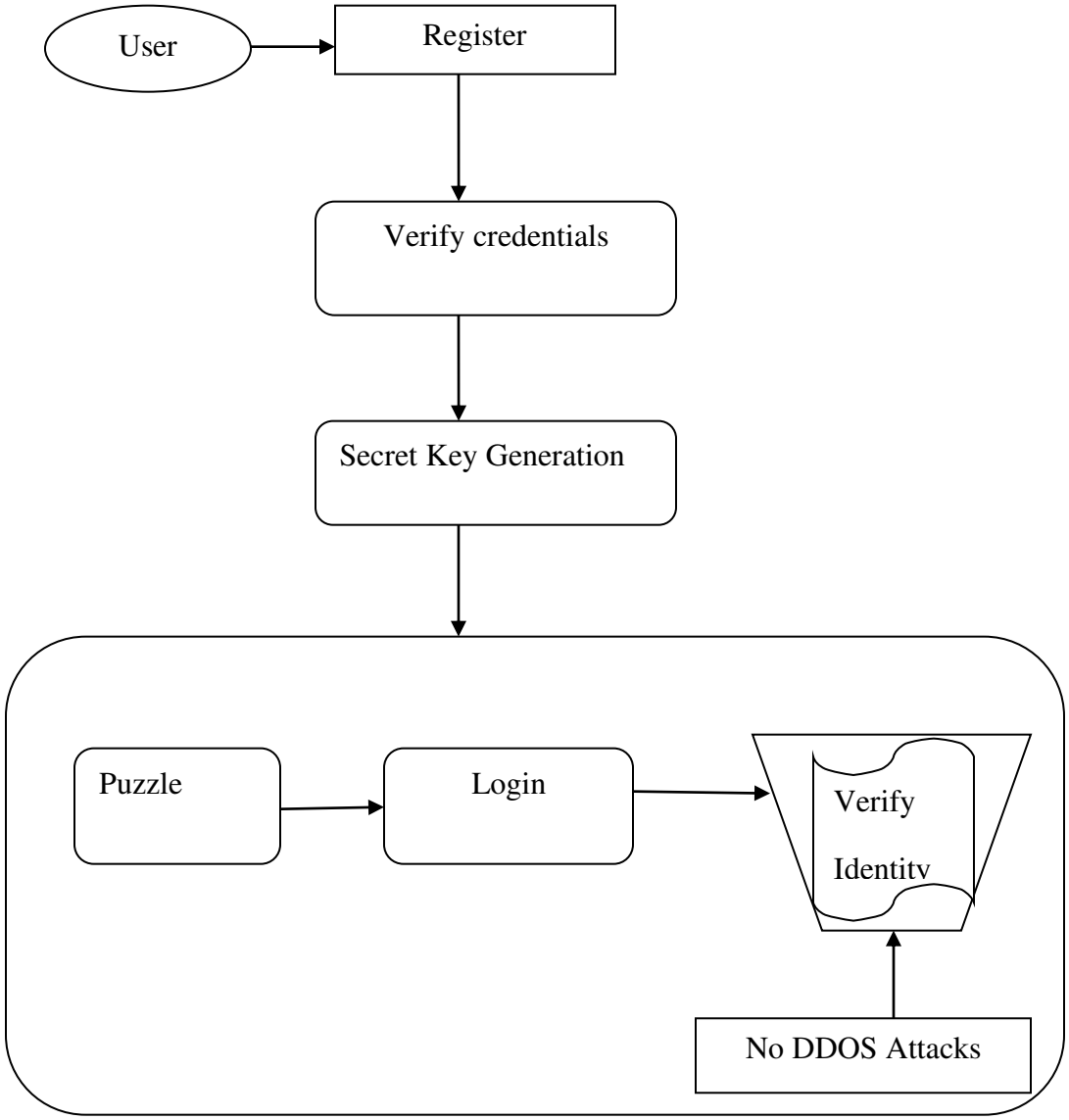
3.1.11 Component Diagram:



EXPLANATION:

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and they are used to illustrate the structure of arbitrarily complex systems. For this in our component diagram to adopt software protection technologies to ensure challenge data confidentiality and code security for an appropriate time period**.** After receiving the puzzle sent from the server, a client tries to solve the puzzle on the host CPU, and replies to the server, as the conventional client puzzle scheme does.

3.2 System Architecture:



EXPLANATION:

The systems architect establishes the basic structure of the system, this we know about that the practical strategy of the attacker is to accelerate the brute force process by exploiting the parallel computation capability of GPU cores. We classify client puzzles into two types. If a puzzle functions *P*, as all the existing client puzzle schemes, is fixed and disclosed in advance, the puzzle is called a data puzzle; otherwise, it is referred to as a puzzle. To ensure challenge

data confidentiality and code security for an appropriate time period**.** After receiving the puzzle sent from the server, a client tries to solve the software puzzle on the host CPU, and replies to the server, as the conventional client puzzle scheme does.

4.13 CONCLUSION:

In this paper, puzzle scheme is proposed for defeating GPU-inflated DoS attack. It adopts software protection technologies to ensure challenge data confidentiality and code security for an appropriate time period, *e.g.*, 1-2 seconds. Hence, it has different security requirement from the conventional cipher which demands long-term confidentiality only, and code protection which focuses on long-term robustness against reverse-engineering only. Since the puzzle may be built upon a data puzzle, it can be integrated with any existing server-side data puzzle scheme, and easily deployed as the present client puzzle schemes do.

4.DEVELOPMENT TOOLS

CHAPTER 4

DEVELOPMENT TOOLS

4.1 GENERAL

This chapter is about the software language and the tools used in the development of the project. The platform used here is JAVA.

4.2 FEATURES OF JAVA

4.2.1 THE JAVA FRAMEWORK

Java is a programming language originally developed by James Gosling at Sun Microsystemsand released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to bytecode that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is general-purpose, concurrent, class-based, and object-oriented, and is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere".

Java is considered by many as one of the most influential programming languages of the 20th century, and is widely used from application software to web applicationsThe java framework is a new platform independent that simplifies application development internet.Java technology's versatility, efficiency, platform portability, and security make it the ideal technology for network computing. From laptops to datacenters, game consoles to scientific supercomputers, cell phones to the Internet, Java is everywhere!

4.2.2 OBJECTIVES OF JAVA

To see places of Java in Action in our daily life, explore java.com.

Why Software Developers Choose Java

Java has been tested, refined, extended, and proven by a dedicated community. And numbering more than 6.5 million developers, it's the largest and most active on the planet. With its versatility, efficiency, and portability, Java has become invaluable to developers by enabling them to:

- Write software on one platform and run it on virtually any other platform
- Create programs to run within a Web browser and Web services
- Develop server-side applications for online forums, stores, polls, HTML forms processing, and more
- Combine applications or services using the Java language to create highly customized applications or services
- Write powerful and efficient applications for mobile phones, remote processors, low-cost consumer products, and practically any other device with a digital heartbeat

Some Ways Software Developers Learn Java

- Today, many colleges and universities offer courses in programming for the Java platform. In addition, developers can also enhance their Java programming skills by reading Sun's java.sun.com Web site, subscribing to Java technology-focused newsletters, using the Java Tutorial and the New to Java Programming Center, and signing up for Web, virtual, or instructor-led courses.

Object Oriented

To be an Object Oriented language, any language must follow at least the four characteristics.

1.Inheritance :It is the process of creating the new classes and using the behavior of the existing classes by extending them just to reuse the existing code and adding addition a features as needed.

2.Encapsulation: It is the mechanism of combining the information and providing the abstraction.

3.Polymorphism: As the name suggest one name multiple form, Polymorphism is the way of providing the different functionality by thefunctions having the same name based on the signatures of the methods.

4.Dynamic binding : Sometimes we don't have the knowledge of objects about their specific types while writing our code. It is the way of providing the maximum functionality to a program about the specific type at runtime.

4.2.3 COLLECTIONS:

The Java Collections API's provide Java developers with a set of classes and interfaces that makes it easier to handle collections of objects. In a sense Collection's works a bit like arrays, except their size can change dynamically, and they have more advanced behavior than arrays. In this project we are using Array List for collecting the user input and saving values.

4.2.4 THREAD:

In this project threading concept is very important. A thread is a sequential path of code execution within a program. And each thread has its own local variables, program counter and lifetime. Like creation of a single thread, we can also create more than one thread (multithreads) in a program using class Thread or implementing interface Runnable to make our project efficient and dynamic. In our project we are using request process with the help of multi threading concepts.

4.2.5 SWINGS:

Swing, which is an extension library to the AWT, includes new and improved components that enhance the look and functionality of GUIs. Swing can be used to build Standalone swing gui apps as well as Servlets and Applets. It employs a model/view design architecture. Swing is more portable and more flexible than AWT.

5.IMPLEMENTATION

CHAPTER 5
IMPLEMENTATION

5.1 GENERAL

SERVER.JAVA:

```
package com.server;

import java.awt.BorderLayout;

import java.awt.EventQueue;

import java.io.InputStream;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.io.OutputStream;

import java.net.InetAddress;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.ArrayList;

public class Server extends JFrame implements Runnable {

        private JPanel contentPane;

        OutputStream sou;

        public ObjectOutputStream sobou;

        public TreeMap t;
```

```
TreeMap ipport=new TreeMap();
TreeMap reg_po=new TreeMap();
public static ArrayList se=new ArrayList();
public ArrayList studentname=new ArrayList();
ArrayList al=new ArrayList<>();
public ArrayList regno=new ArrayList();
public ArrayList department=new ArrayList();
public ArrayList file=new ArrayList();
public static ArrayList table=new ArrayList();
public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
                public void run() {
                        try {
                                Server frame = new Server();
                                frame.setVisible(true);

                                Thread t=new Thread(frame);
                                t.start();
                                } catch (Exception e) {
                                e.printStackTrace();
                        }
```

```
                }
        });
    }
    @Override
    public void run() {
        // TODO Auto-generated method stub
        try{
            InetAddress ip=InetAddress.getLocalHost();
            String ipadd=ip.getHostAddress();
            ServerSocket serverSocket=null;
            serverSocket=new ServerSocket(101);
        while(true)
        {
        //System.out.println("Server Waiting");
System.out.println(" Main Server waiting for The Ip Address :  "+ipadd+" And the Protocol No is.....:101");
            Socket socket=serverSocket.accept();
            System.out.println("accepting request");
    ObjectInputStream objectInputStream=new ObjectInputStream(socket.getInputStream());
            ArrayList arrayList=(ArrayList)objectInputStream.readObject();
            String request=arrayList.get(0).toString();
```

```
System.out.println("register is\t"+request);
if(request.equals("register"))
{
        //Registration rg=(Registration)arrayList.get(1);
        String user=arrayList.get(1).toString().trim();
        String pass=arrayList.get(2).toString().trim();
        String cpass=arrayList.get(3).toString().trim();
        String email=arrayList.get(4).toString().trim();
        String sex=arrayList.get(5).toString().trim();
        String dob=arrayList.get(6).toString().trim();
        String con=arrayList.get(7).toString().trim();
        String add=arrayList.get(8).toString().trim();
        String ip1=arrayList.get(9).toString().trim();

        ArrayList al=new ArrayList<>();
        al.add(user);
        al.add(pass);
        al.add(cpass);
        al.add(email);
        al.add(sex);
        al.add(dob);
```

```
                    al.add(con);
                    al.add(add);
                    al.add(ip1);
        MethodImplementation implementation=MethodImplementation.getImplementation();
        int status=implementation.insert(al);
        ObjectOutputStream objectOutputStream=new
ObjectOutputStream(socket.getOutputStream());
                    objectOutputStream.writeObject(status);
                    arrayList.clear();
                        }
                    if(request.equals("login"))
                        {
        Registration rg=(Registration)arrayList.get(1);
                System.out.println("LOgin"+rg);
        MethodImplementation implementation=MethodImplementation.getImplementation();
                    boolean status=implementation.verifyCredentials(rg);
ObjectOutputStream objectOutputStream=new ObjectOutputStream(socket.getOutputStream());
                    objectOutputStream.writeObject(status);
                    arrayList.clear();
                    }
                    }
```

Picpuzzle.java

```
package com.design;

import java.awt.event.*;

import java.awt.*;

import javax.swing.*;

class picpuzzle2 extends JFrame implements ActionListener{

JButton b1,b2,b3,b4,b5,b6,b7,b8,b9,sample,starB;

Icon s1;

Icon star;

Icon ic0=new
ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\starB0.jpg");

Icon ic10=new
ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\starB10.jpg");

Icon ic20=new
ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\starB20.jpg");

Icon samicon1=new
ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\main.jpg");

Icon samicon2=new
ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\main2.jpg");

Icon samicon3=new
ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\main3.jpg");
```

```
Icon ic1=new ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\1.jpg");

Icon ic2=new ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\5.jpg");

Icon ic3=new ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\2.jpg");

Icon ic4=new ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\7.jpg");

Icon ic5=new ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\4.jpg");

Icon ic6=new ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\6.jpg");

Icon ic7=new ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\8.jpg");

Icon ic8=new ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\9.jpg");

Icon ic9=new ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\3.jpg");


Icon ic11=new
ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\12.jpg");

Icon ic12=new
ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\13.jpg");

Icon ic13=new
ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\16.jpg");

Icon ic14=new
ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\11.jpg");

Icon ic15=new
ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\14.jpg");
```

```
Icon ic16=new
ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\19.jpg");

Icon ic17=new
ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\17.jpg");

Icon ic18=new
ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\15.jpg");

Icon ic19=new
ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\18.jpg");

  Icon ic21=new
ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\24.jpg");

Icon ic22=new
ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\25.jpg");

Icon ic23=new
ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\21.jpg");

Icon ic24=new
ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\27.jpg");

Icon ic25=new
ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\23.jpg");

Icon ic26=new
ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\29.jpg");

Icon ic27=new
ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\28.jpg");
```

```
Icon ic28=new
ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\22.jpg");

Icon ic29=new
ImageIcon("E:\\prabaworkspace\\puzzleencrytion\\src\\pic\\26.jpg");

  picpuzzle2(){

  super("pic puzzle");

star=b9.getIcon();
getContentPane().add(b1);getContentPane().add(b2);getContentPane().add(b3);get
ContentPane().add(b4);getContentPane().add(b5);getContentPane().add(b6);getCo
ntentPane().add(b7);getContentPane().add(b8);

getContentPane().add(b9);getContentPane().add(sample);getContentPane().add(l1)
;getContentPane().add(l2);getContentPane().add(starB);getContentPane().add(l3);

b1.addActionListener(this); b2.addActionListener(this);

 b3.addActionListener(this); b4.addActionListener(this);

b5.addActionListener(this); b6.addActionListener(this);

 b7.addActionListener(this); b8.addActionListener(this);

 b9.addActionListener(this);

  sample.addActionListener(this);

getContentPane().setLayout(null);

setSize(600,500);

setVisible(true);
```

```
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
  public void actionPerformed(ActionEvent e){
if(e.getSource()==b1){
    Icon s1=b1.getIcon();
      if(b2.getIcon()==star){
        b2.setIcon(s1);
        b1.setIcon(star);
      } else if(b4.getIcon()==star){
                  b4.setIcon(s1);
                  b1.setIcon(star);
                 }
  }//end of if
  if(e.getSource()==b2){
    Icon s1=b2.getIcon();
      if(b1.getIcon()==star){
        b1.setIcon(s1);
        b2.setIcon(star);
      } else if(b5.getIcon()==star){
```

```
                b5.setIcon(s1);
                b2.setIcon(star);
               }
       else if(b3.getIcon()==star){
                b3.setIcon(s1);
                b2.setIcon(star);
               }
 }//end of if
 if(e.getSource()==b3){
   Icon s1=b3.getIcon();
     if(b2.getIcon()==star){
       b2.setIcon(s1);
       b3.setIcon(star);
     } else if(b6.getIcon()==star){
                b6.setIcon(s1);
                b3.setIcon(star);
               }
 }//end of if
 if(e.getSource()==b4){
```

```
  Icon s1=b4.getIcon();
    if(b1.getIcon()==star){
      b1.setIcon(s1);
      b4.setIcon(star);
    } else if(b5.getIcon()==star){
                b5.setIcon(s1);
                b4.setIcon(star);
               }
       else if(b7.getIcon()==star){
                b7.setIcon(s1);
                b4.setIcon(star);
               }
 }//end of if
 if(e.getSource()==b5){
   Icon s1=b5.getIcon();
    if(b2.getIcon()==star){
      b2.setIcon(s1);
      b5.setIcon(star);
    } else if(b4.getIcon()==star){
```

```
                b4.setIcon(s1);
                b5.setIcon(star);
               }
        else if(b6.getIcon()==star){
                b6.setIcon(s1);
                b5.setIcon(star);
               }
         else if(b8.getIcon()==star){
                b8.setIcon(s1);
                b5.setIcon(star);
               }
  }//end of if
 if(e.getSource()==b6){
   Icon s1=b6.getIcon();
     if(b3.getIcon()==star){
       b3.setIcon(s1);
       b6.setIcon(star);
     } else if(b5.getIcon()==star){
                b5.setIcon(s1);
```

```
                b6.setIcon(star);
                }
        else if(b9.getIcon()==star){
                b9.setIcon(s1);
                b6.setIcon(star);
                }
}//end of if
 if(e.getSource()==b7){
   Icon s1=b7.getIcon();
    if(b4.getIcon()==star){
      b4.setIcon(s1);
      b7.setIcon(star);
    } else if(b8.getIcon()==star){
                b8.setIcon(s1);
                b7.setIcon(star);
                }
 }

  if(e.getSource()==b8){
```

```
 Icon s1=b8.getIcon();
   if(b7.getIcon()==star){
     b7.setIcon(s1);
     b8.setIcon(star);
   } else if(b5.getIcon()==star){
                b5.setIcon(s1);
                b8.setIcon(star);
               }
      else if(b9.getIcon()==star){
                b9.setIcon(s1);
                b8.setIcon(star);
               }
 }
if(e.getSource()==b9){
 Icon s1=b9.getIcon();
   if(b8.getIcon()==star){
     b8.setIcon(s1);
     b9.setIcon(star);
   } else if(b6.getIcon()==star){
```

```
                b6.setIcon(s1);
                b9.setIcon(star);
               }
  }
if(e.getSource()==sample){
 s1=sample.getIcon();
 if(s1==samicon3){
sample.setIcon(samicon1);
b1.setIcon(ic1);
b2.setIcon(ic2);
b3.setIcon(ic3);
b4.setIcon(ic4);
b5.setIcon(ic5);
b6.setIcon(ic6);
b7.setIcon(ic7);
b8.setIcon(ic8);
b9.setIcon(ic9);
star=b9.getIcon();
System.out.println("completed");
```

```
System.out.println(star+"hello");

starB.setIcon(ic0);

System.out.println("completed");}//eof if

else if(s1==samicon1){

sample.setIcon(samicon2);

b1.setIcon(ic11);

b2.setIcon(ic12);

b3.setIcon(ic13);

b4.setIcon(ic14);

b5.setIcon(ic15);

b6.setIcon(ic16);

b7.setIcon(ic17);

b8.setIcon(ic18);

b9.setIcon(ic19);

star=b6.getIcon();

System.out.println("completed");


System.out.println(star+"hello");

starB.setIcon(ic10);
```

```
System.out.println("completed");
}//eof else
else{
sample.setIcon(samicon3);
b1.setIcon(ic21);
b2.setIcon(ic22);
b3.setIcon(ic23);
b4.setIcon(ic24);
b5.setIcon(ic25);
b6.setIcon(ic26);
b7.setIcon(ic27);
b8.setIcon(ic28);
b9.setIcon(ic29);
star=b6.getIcon();
System.out.println("completed");
System.out.println(star+"hello");
starB.setIcon(ic20);
System.out.println("completed");
Login l=new Login();
```

```
l.setVisible(true);
dispose();
}//eof else
 Login l=new Login();
 l.setVisible(true);
 dispose();
}
}
  public static void main(String args[]){
new picpuzzle2();
}//end of main
}//end of class
```

Home.java

```
package login;
import java.awt.BorderLayout;
import java.awt.EventQueue;
import javax.swing.JFrame;
import javax.swing.JPanel;
```

```
import javax.swing.border.EmptyBorder;

import javax.swing.JLabel;

import java.awt.Color;

import java.awt.Font;

import javax.swing.JOptionPane;

import javax.swing.JTextField;

import javax.swing.JPasswordField;

import javax.swing.JButton;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import java.io.InputStream;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.io.OutputStream;

import java.net.InetAddress;

import java.net.Socket;

import java.util.ArrayList;

import java.util.List;

import java.util.Random;
```

```
import java.util.TreeMap;
 import javax.swing.ImageIcon; import admin.Admin;
 public class Home extends JFrame {
        ArrayList li=new ArrayList();
        private JPanel contentPane;
        private JTextField regno;
        public String uname,pass;
public static ArrayList us=new ArrayList();
        public static void main(String[] args) {
                EventQueue.invokeLater(new Runnable() {
                        public void run() {
                                try {
                                        Home frame = new Home();
                                        frame.setVisible(true);
                                } catch (Exception e) {
                                        e.printStackTrace();
                                }
                        }
                });
```

```
        }
        public Home() {
        setForeground(new Color(0, 0, 255));
        setFont(new Font("Book Antiqua", Font.BOLD | Font.ITALIC, 10));
                setResizable(false);
                setTitle(" UNIVERSITY RESULT");
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                setBounds(100, 100, 588, 451);
                contentPane = new JPanel();
                contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
                setContentPane(contentPane);
                contentPane.setLayout(null);
                final JPanel panel = new JPanel();
                panel.setBounds(0, 0, 582, 423);
                contentPane.add(panel);
                panel.setLayout(null);
JLabel lblRegesterNo = new JLabel("ENTER YOUR REGESTER  NO:");
                lblRegesterNo.setFont(new Font("Tahoma", Font.PLAIN, 14));
                lblRegesterNo.setForeground(new Color(255, 69, 0));
```

```
lblRegesterNo.setBounds(95, 273, 219, 14);
panel.add(lblRegesterNo);
regno = new JTextField();
regno.setFont(new Font("Palatino Linotype", Font.BOLD, 13));
regno.setForeground(new Color(0, 0, 0));
regno.setColumns(10);
regno.setBounds(305, 265, 173, 32);
panel.add(regno);
JButton SUBMIT = new JButton("SUBMIT");
SUBMIT.setFont(new Font("Tahoma", Font.BOLD, 11));
SUBMIT.setIcon(new
ImageIcon("E:\\prabaworkspace\\Annuniversity\\src\\images\\clean.png"));
SUBMIT.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
        try
                {
                 String re=regno.getText();
                 if(re.equals(""))
                 {
```

```
                                        JOptionPane.showMessageDialog(panel,
"ENTER YOUR REGISTER NO");
                                        }
                                        else
                                        {
                                        String reg=re.toUpperCase();
                                                Random rand = new Random();
                                               int  val = rand.nextInt(50);
                                                System.out.println("port no \t"+val);
                                       System.out.println(reg);
                                       li.add("regno");
                                       li.add(reg);
                                       TreeMap resul=new TreeMap();
                                       TreeMap request=new TreeMap();
                        String ip=InetAddress.getLocalHost().getHostAddress();
                                resul.put(reg, val);
                                resul.put(String.valueOf(val), ip);
                        Socket clients= new Socket("192.168.0.5",177);
                        OutputStream ou=clients.getOutputStream();
```

```
                ObjectOutputStream obou=new ObjectOutputStream(ou);
                    obou.writeObject(li);
                ObjectOutputStream obo=new ObjectOutputStream(ou);
                    obo.writeObject(resul);
                    System.out.println("ip in home\t"+ip);
                    int port=clients.getPort();
                    InputStream in=clients.getInputStream();
                    ObjectInputStream obin=new ObjectInputStream(in);
                    String sname=obin.readObject().toString();
                    System.out.println("sname is\t"+sname);
                    if(sname.equalsIgnoreCase("false"))
                    {
JOptionPane.showMessageDialog(panel, "INVALID REGISTER NO");
                        regno.setText("");
                        dispose();
                        }
                    else
                    {
                    dispose();
```

```
                    Studentpage.port(val,sname,ip,port,reg);
                    Studentpage.main(null);
                    }
                     }
                    }
```

```
        JLabel lblWelcomeToUser = new JLabel("WELCOME TO ANNA
UNIVERSITY");
        lblWelcomeToUser.setBounds(175, 36, 249, 14);
        panel_1.add(lblWelcomeToUser);
        lblWelcomeToUser.setForeground(new Color(128, 0, 0));
        lblWelcomeToUser.setFont(new Font("SimSun-ExtB", Font.BOLD,
16));
        lblWelcomeToUser.setBackground(Color.GRAY);
        JLabel lblNewLabel_2 = new JLabel("");
        lblNewLabel_2.setIcon(new
ImageIcon("E:\\prabaworkspace\\Annuniversity\\img\\bimg\\hd\\abstract_backgrou
nd_3-wallpaper-1280x960.jpg"));
        lblNewLabel_2.setBounds(0, 116, 584, 307);
        panel.add(lblNewLabel_2);
```

```
        }
}
```

Admin.java:

```
package admin;
import java.awt.BorderLayout;
import java.awt.EventQueue;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import java.awt.Color;
import java.awt.Font;
import javax.swing.JOptionPane;
public class Admin extends JFrame {
        private JPanel contentPane;
        private JTextField name;
        private JPasswordField password;
public static String uname;
        public static void main(String[] args) {
```

```
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    Admin frame = new Admin();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }
    public Admin( ) {
        setTitle("ADMIN LOGIN");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 345, 374);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);
```

```
        final JPanel panel = new JPanel();
        panel.setBounds(0, 0, 329, 336);
        contentPane.add(panel);
        panel.setLayout(null);
    JLabel adminlogin = new JLabel("WELCOME TO ADMIN LOGIN");
        adminlogin.setFont(new Font("SimSun-ExtB", Font.BOLD, 16));
        adminlogin.setForeground(new Color(128, 0, 0));
        adminlogin.setBackground(Color.GRAY);
        adminlogin.setBounds(67, 49, 210, 14);
        panel.add(adminlogin);
        JLabel lblNewLabel = new JLabel("USER NAME");
        lblNewLabel.setFont(new Font("Tahoma", Font.PLAIN, 14));
        lblNewLabel.setForeground(Color.RED);
        lblNewLabel.setBounds(44, 126, 98, 14);
        panel.add(lblNewLabel);
        JLabel lblNewLabel_1 = new JLabel("PASSWORD");
        lblNewLabel_1.setFont(new Font("Tahoma", Font.PLAIN, 14));
        lblNewLabel_1.setForeground(Color.RED);
        lblNewLabel_1.setBounds(44, 196, 78, 14);
```

```
panel.add(lblNewLabel_1);
name = new JTextField();
name.setForeground(new Color(165, 42, 42));
name.setFont(new Font("Tahoma", Font.PLAIN, 13));
name.setBounds(165, 126, 112, 20);
panel.add(name);
name.setColumns(10);
JButton SUBMIT = new JButton("LOGIN");
SUBMIT.setIcon(new
ImageIcon("E:\\prabaworkspace\\Annuniversity\\src\\images\\login (2).png"));
SUBMIT.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
String uname=name.getText();
String pass=String.valueOf(password.getPassword());
if("y".equals(uname)&&"s".equals(pass))
{
JOptionPane.showMessageDialog(panel, "Welcome admin");
dispose();
Studentloginstatus frame = new Studentloginstatus();
```

```
                    frame.setVisible(true);
                }
                else
                    JOptionPane.showMessageDialog(panel,
"Username or Password wrong");
            }
        });
        SUBMIT.setMnemonic('I');
        SUBMIT.setBounds(47, 253, 123, 41);
        panel.add(SUBMIT);
        JButton EXIT = new JButton("EXIT");
        EXIT.setIcon(new
ImageIcon("E:\\prabaworkspace\\Annuniversity\\src\\images\\exit.png"));
        EXIT.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                dispose();
            }
        });
        EXIT.setMnemonic('X');
        EXIT.setBounds(202, 252, 98, 42);
```

```
                panel.add(EXIT);

                password = new JPasswordField();

                password.setForeground(new Color(165, 42, 42));

                password.setFont(new Font("Tahoma", Font.PLAIN, 13));

                password.setBounds(165, 193, 112, 20);

                panel.add(password);

                JLabel lblNewLabel_2 = new JLabel("");

                lblNewLabel_2.setIcon(new
ImageIcon("E:\\prabaworkspace\\Annuniversity\\src\\images\\abstract_background
_3-wallpaper-1280x960.jpg"));

                lblNewLabel_2.setBounds(0, 0, 329, 336);

                panel.add(lblNewLabel_2);

        }

}
```

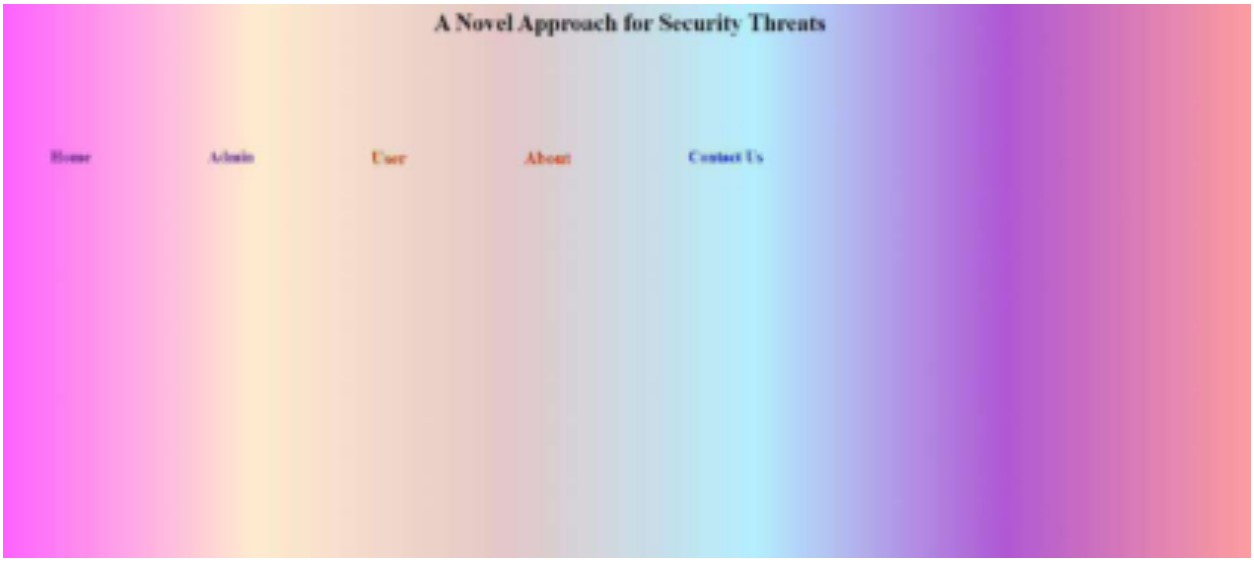
6.SNAPSHOTS

CHAPTER 6

SNAPSHOTS

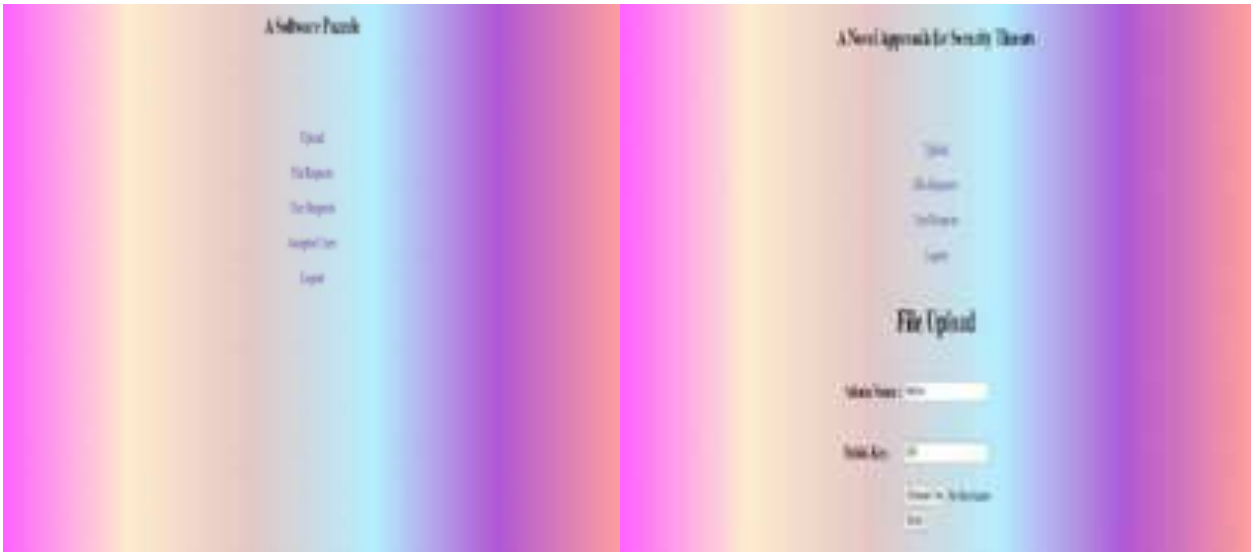
6.1 General

Home Page



Admin login

6.2 Various Snapshots



User Registration

7.TESTING

CHAPTER 7

SOFTWARE TESTING

7.1 GENERAL

The seriousness of the DoS/DDoS problem and their increased frequency has led to the advent of numerous defense mechanisms. In this paper, we are particularly interested in the countermeasures to DoS/DDoS attacks on server computation power. DoS and DDoS are effective if attackers spend much less resources than the victim server or are much more powerful than normal users.

7.2 DEVELOPING METHODOLOGIES

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used.

The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

7.3Types of Tests

7.3.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

7.3.2 Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

7.3.3 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

7.3.4 Performance Test

The Performance test ensures that the output be produced within the time limits,and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

7.3.5 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

7.3.6 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Acceptance testing for Data Synchronization:

- The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
- The Route add operation is done only when there is a Route request in need
- The Status of Nodes information is done automatically in the Cache Updation process

7.3.7 Build the test plan

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

8.RESULT

CHAPTER 8
RESULT

8.1 APPLICATION

A Network application is any application running on one host and provides a communication to another application running on a different host, the application may use an existing application layer protocols such as: HTTP(e.g. the Browser and web server), and may be the application does not use any existing protocols and depends on the socket programming to communicate to another application. So the web application is a type of the network applications. We can distinguish between **network** and **stand-alone** applications. For example, if you use Microsoft Word to write a letter and save it on your PC, both the program and the data are stored on your computer. Since your computer does not have to be connected to a network, this is an example of a stand-alone application.

8.2 FUTURE ENHANCEMENTS

GPU-inflation attack, its idea can be extended to thwart DoS attackers which exploit other inflation resources such as Cloud Computing. For example, suppose the server inserts some anti-debugging codes for detecting Cloud platform into software puzzle, when the puzzle is running.

9.CONCLUSION

CHAPTER 9
CONCLUSION

In this article, we study camera-related vulnerabilities in Android phones for mobile multimedia applications. We discuss the roles a spy camera can play to attack or benefit phone users. We discover several advanced spy camera attacks, including the remote-controlled real-time monitoring attack and two types of passcode inference attacks. Meanwhile, we propose an effective defense scheme to secure a smartphone from all these spy camera attacks. In the future, we will investigate the feasibility of performing spy camera attacks on other mobile operating systems.

10.BIBILIOGRAPHY

CHAPTER 10

BIBILIOGRAPHY

1.Y. Zhou and X. Jiang, "Dissecting Android Malware: Characterization and Evolution", IEEE Symp. Security and Privacy 2012, pp. 95-109, 2012.

2.R. Schlegel, "Soundcomber: A Stealthy and Context Aware Sound Trojanfor Smartphones", NDSS, pp. 17-33, 2011.

3.N. Xu, "Stealthy Video Capturer: A New Video-Based Spyware in 3g Smartphones", Proc. 2nd ACM Conf. Wireless Network Security, pp. 69-78, 2009.

4.F. Maggi, "A Fast Eavesdropping Attack against Touchscreens", 7th Int'l. Conf. Info. Assurance and Security, pp. 320-25, 2011.

5.R. Raguram, "ispy: Automatic Reconstruction of Typed Input from Compromising Reflections", Proc. 18th ACM Conf. Computer and Commun. Security, pp. 527-36, 2011.

6."Android-eye", 2012.

7."Nanohttpd".

8.A. P. Felt and D. Wagner, "Phishing on Mobile Devices", Proc. WEB 2.0 Security and Privacy, 2011.

9.D. Li, D. Winfield and D. Parkhurst, "Starburst: A Hybrid Algorithm for Video-Based Eye Tracking Combining Feature-Based and Model-Based Approaches", IEEE Computer Soc. Conf. Computer Vision and Pattern Recognition Workshops, pp. 79, 2005.

10.P. Aldrian, "Fast Eyetracking", 2009.

Vol 13, Issue 06, June/2022

ISSN NO:0377-9254

Security Threats To Mobile Multimedia Applications

1J SRIVIDYA, 2PHANI CH, 3ANIKETH D, 4ROHITNIVAS B

1Assistant Professor, 234B.Tech Student

DEPT OF CSE

CMR TECHNICAL CAMPUS, Hyderabad

ABSTRACT: Today's mobile smartphones are very powerful, and many smartphone applications use wireless multimedia communications. Mobile phone security has become an important aspect of security issues in wireless multimedia communications. As the most popular mobile operating system, Android security has been extensively studied by researchers. However, few works have studied mobile phone multimedia security. In this article, we focus on security issues related to mobile phone cameras. Specifically, we discover several new attacks that are based on the use of phone cameras. We implement the attacks on real phones, and demonstrate the feasibility and effectiveness of the attacks. Furthermore, we propose a lightweight defense scheme that can effectively detect these attacks.

I.INTRODUCTION

Since 2007, the Android operating system (OS) has enjoyed an incredible rate of popularity. As of 2013, the Android OS holds 79.3 percent of global smartphone market shares. Meanwhile, a number of Android security and privacy vulnerabilities have been exposed in the past several years. Although the Android permission system gives users an opportunity to check the permission request of an application (app) before installation, few users have knowledge of what all these permission requests stand for; as a result, they fails to warn users of security risks. Meanwhile, an increasing number of apps specified to enhance security and protect user privacy have appeared in Android app markets. Most large anti-virus software companies have published their Android-version security apps, and tried to provide a shield for smartphones by detecting and blocking malicious apps. In addition, there are data protection apps that provide users the capability to encrypt, decrypt, sign, and verify signatures for private texts, emails, and files. However, mobile malware and privacy leakage remain a big threat to mobile phone security and privacy.

Generally, when talking about privacy protection, most smartphone users pay attention to the safety of SMS, emails, contact lists, calling histories, location information, and private files. They may be surprised that the phone camera could become a traitor; for example, attackers could stealthily take pictures and record videos by using the phone camera. Nowadays, various types of camera-based applications have appeared in Android app markets (photography, barcode readers, social networking, etc.). Spy camera apps have also become quite popular. As for Google Play, there are nearly 100 spy camera apps, which allow phone users to take pictures or record videos of other people without their permission. However, believe it or not, phone users themselves could also become victims.

Attackers can implement spy cameras in malicious apps such that the phone camera is launched automatically without the device owner’s notice, and the captured photos and videos are sent out to these remote attackers. Even worse, according to a survey on Android malware analysis [1], camera permission ranks 12th of the most commonly requested permissions among benign apps, while it is out of the top 20 in malware. The popularity of camera usage in benign apps and relatively less usage in

malware lower users' alertness to camera-based multimedia application attacks.

1.1 OBJECTIVE:

Nowadays, people carry their phones everywhere; hence, their phones see lots of private information. If the phone camera is exploited by a malicious spy camera app, it may cause serious security and privacy problems. For example, the phone camera may record a user's daily activities and conversations, and then send these out via the Internet or multimedia messaging service (MMS). Secret photography is not only immoral but also illegal in some countries due to the invasion of privacy. Nevertheless, a phone camera could also provide some benefits if it is con-trolled well by the device owner. For example, when the owner wants to check if someone has used his/her phone without permission, the phone camera could be used to record the face of an unauthorized user. Besides, it can also help the owner find a lost phone.

In this article, we first conduct a survey on the threats and benefits of spy cameras. Then we present the basic attack model and two camera based attacks: the remote-controlled real-time monitoring attack and the passcode inference attack. We run these attacks along with popular antivirus software to test their stealthiness, and conduct experiments to evaluate both types of attacks. The results demonstrate the feasibility and effectiveness of these attacks. Finally, we propose a lightweight defense scheme.

II. EXISTING SYSTEM:

Several video-based attacks targeted at keystrokes have been proposed. The attacks can obtain user input on touch screen smart phones. Maggi *et al.* implement an automatic shoulder surfing attack against modern touch-enabled smart phones. The attacker deploys a video camera that can record the target screen while the victim is entering text. Then user input can be reconstructed solely based on the keystroke feedback displayed on the screen.

DRAWBACKS IN EXISTING SYSTEM:

- It works only when visual feedback such as magnified keys are available.
- Mobile malware and privacy leakage remain a big threat to mobile phone security and privacy

III.LITERATURE SURVEY:

TITLE : Dissecting Android Malware: Characterization and Evolution
AUTHOR : Y. Zhou and X. Jiang,
YEAR : 2012

DESCRIPTION

The popularity and adoption of smart phones has greatly stimulated the spread of mobile malware, especially on the popular platforms such as Android. In light of their rapid growth, there is a pressing need to develop effective solutions. However, our defense capability is largely constrained by the limited understanding of these emerging mobile malware and the lack of timely access to related samples. In this paper, we focus on the Android platform and aim to systematize or characterize existing Android malware. Particularly, with more than one year effort, we have managed to collect more than 1,200 malware samples that cover the majority of existing Android malware families, ranging from their debut in August 2010 to recent ones in October 2011. In addition, we systematically characterize them from various aspects, including their installation methods, activation mechanisms as well as the nature of carried malicious payloads. The characterization and a subsequent evolution-based study of representative families reveal that they are evolving rapidly to circumvent the detection from existing mobile anti-virus software. Based on the

evaluation with four representative mobile security software, our experiments show that the best case detects 79.6% of them while the worst case detects only 20.2% in our dataset. These results clearly call for the need to better develop next-generation anti-mobile-malware solutions.

TITLE : A Fast Eavesdropping Attack against Touchscreens.

AUTHOR : F. Maggi, et

YEAR : 2011.

DESCRIPTION

The pervasiveness of mobile devices increases the risk of exposing sensitive information on the go. In this paper, we arise this concern by presenting an automatic attack against modern touchscreen keyboards. We demonstrate the attack against the Apple iPhone - 2010's most popular touchscreen device - although it can be adapted to other devices (e.g., Android) that employ similar key-magnifying keyboards. Our attack processes the stream of frames from a video camera (e.g., surveillance or portable camera) and recognizes keystrokes online, in a fraction of the time needed to perform the same task by direct observation or offline analysis of a recorded video, which can be unfeasible for large amount of data. Our attack detects, tracks, and rectifies the target touchscreen, thus following the device or camera's movements and eliminating possible perspective distortions and rotations In real-world settings, our attack can automatically recognize up to 97.07 percent of the keystrokes (91.03 on average), with 1.15 percent of errors (3.16 on average) at a speed ranging from 37 to 51 keystrokes per minute.

IV.PROPOSED SYSTEM:

In this article, we first conduct a survey on the threats and benefits of spy cameras. Then we present the basic attack model and two camera based attacks: the remote-controlled real-time monitoring attack and the passcode inference attack. We run these attacks along with popular antivirus software to test their stealthiness, and conduct experiments to evaluate both types of attacks. The results demonstrate the feasibility and effectiveness of these attacks.

ADVANTAGES IN PROPOSED SYSTEM:

- The attacker needs considerable effort in translating central processing unit puzzle software to its functionally equivalent GPU version such that the translation cannot be done in real time.
- Moreover, we show how to implement puzzle in the generic server-browser model. To outsourcing any business onto a cloud.
- By using this Applications, we can easily be avoided by selecting the time to launch attack.
- The malicious camera app can periodically check the screen status and run the stealthy video recording only when the screen is off, which means that the user is not using the phone and the camera device is idle.

V.IMPLEMANTATION

The seriousness of the DoS/DDoS problem and their increased frequency has led to the advent of numerous defense mechanisms. In this paper, we are particularly interested in the countermeasures to DoS/DDoS attacks on server computation power. DoS and DDoS are effective if attackers spend much less resources than the victim server or are much more powerful than normal users.

There are five modules for the Software Puzzle.

5.1MODULES

- USER INTERFACE DESIGN
- GPU-INFLATED DOS ATTACK
- PUZZLE GENERATION
- CODE PROTECTION

- SECURITY ANALYSIS

- **User Interface Design:**

This is the first module of our project. The important role for the cloud user is to move login window to cloud user window. This module has created for the security purpose. In this login page we have to enter login user id and password. It will check username and password is match or not (valid user id and valid password). If we enter any invalid username or password we can't enter into login window to user window it will shows error message. So we are preventing from unauthorized user entering into the login window to user window. It will provide a good security for our project. So server contain user id and password server also check the authentication of the user. It well improves the security and preventing from unauthorized user enters into the network. In our project we are using JSP for creating design. Here we validate the login user and server authentication.

- **GPU-Inflated Dos Attack:**

In order to elaborate software puzzle, we recap its rival GPU-inflated DoS attack in advance. When a client wants to obtain a service, she sends a request to the server. After receiving the client request, the server responds with a puzzle challenge x. If the client is genuine, she will find the puzzle solution y directly on the host CPU, and send the response (x, y) to the server. However, as shown in Fig. 1, by using the similar mechanism in accelerating calculation with GPU, a malicious user who controls the host will send the challenge x to GPU and exploit the GPU resource to accelerate the puzzle-solving process.

Since the virtual keyboard in a touch screen smartphone is much smaller than computer keyboards, the virtual keys are very close to each other. Based on measurement of a Galaxy Nexus 4 phone, even an offset of 5 mm could result in touching the wrong key. Hence, when typing, users tend to keep a short distance to the screen, which allows the phone (front) camera to have a clear view of a user's eye movements. A user's eyes move along with the keys being touched, which means that tracking the eye movement could possibly tell what the user is entering. Thus, it is of great importance to investigate whether an attacker could obtain a phone user's passcode by tracking the eye movements.

- **Puzzle Generation:**

In order to construct a puzzle, the server has to execute three modules: puzzle core generation, puzzle challenge generation, puzzle encrypting/obfuscating.

1) **Puzzle Core Generation***:* From the code block warehouse, the server first chooses n code blocks based on hash functions and a secret, e.g., the j th instruction block $bi\,j$, where $i\,j = H1(y, j)$, and $y = H2(key, sn)$, with one-way functions $H1(\cdot)$ and $H2(\cdot)$, key is the server's secret, and sn is a nonce or timestamp. All the chosen blocks are assembled into a puzzle core, denoted as $C(\cdot) = (bi1 ; bi2 ; \cdots ; bin)$. As an illustrative example, Table III in the appendix shows an example puzzle core C generated from AES operation blocks stored in warehouse **S**.

2) **Puzzle Challenge Generation***:* Given some auxiliary input messages such as IP addresses, and in-line constants, the server calculates a message m from public data such as their IP addresses, port numbers and cookies, and produces a challenge $x = C(y,m)$, smiliar to encrypting plaintext m with key y to produce ciphertext x. As the attacker does not know the puzzle core $C(\cdot)$ (or equivalently the puzzle function $P(\cdot)$) in advance, it can not exploit GPU to solve the puzzle $C0x$ in real time using the basic GPU-inflated DoS attack addressed in Subsection III-A. Nonetheless, if the puzzle is merely constructed as aboveit is possible for an attacker to generate the GPU kernel by mapping the CPU instructions in $C0x$ to the GPU instructions one by one, i.e., to automatically

translate the CPU software puzzle *C0x* into its functionally equivalent GPU version.

- **Code Protection:**

  Intuitively, code obfuscation is able to thwart the above translation threat to some extent. Though there are no generic obfuscation techniques which can prevent a patient and advanced hacker from understanding a program results in show that obfuscation does increase the cost of reverse-engineering. Thus, although code obfuscation may be not satisfactory in long-term software defense against hacking, it is suitable for fortifying software puzzles which demand a protection period of several seconds only.
  A puzzle consists of instructs, and each instruction has a form (opCode, [operands]), where opCode indicates which operation (*e.g.*, addition, shift, jump) is, while

  the operands, varying with opCode, are the parameters (*e.g.*, target address of jump instruction) to complete the operations. As a popular obfuscation technology, code encryption technology treats software code as data string and encrypts both operand and opCode.

- **Security Analysis:**

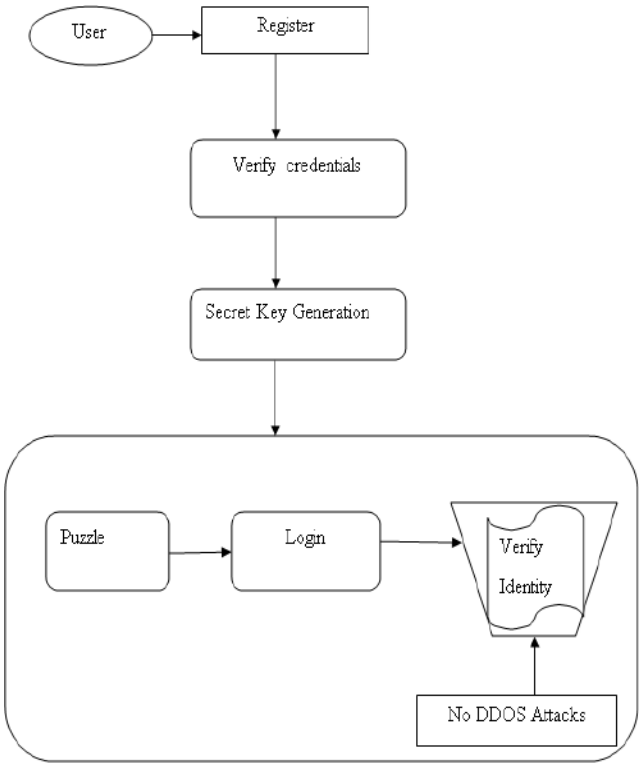
In this module puzzle aims to prevent GPU from being used in the puzzle-solving process based on different instruction sets and real-time environments between GPU and CPU. Conversely, an adversary may attempt to deface the puzzle scheme by simulating the host on GPU (Subsection V-A), cracking puzzle algorithm (Subsection V-B), re-producing GPU-version puzzle (Subsections V-C ~ V-E), or abusing the access priority in puzzle-solving (Subsection V-F).

5.2 SYSTEM TECHNIQUES:

Algorithm: Cracking Data Puzzle Algorithm

The practical strategy of the attacker is to accelerate the brute force process by exploiting the parallel computation capability of GPU cores. We classify client puzzles into two types. If a puzzle functions *P*, as all the existing client puzzle schemes, is fixed and disclosed in advance, the puzzle is called a data puzzle; otherwise, it is referred to as a software puzzle.

VI.SYSTEM ARCHITECTURE



EXPLANATION:

The systems architect establishes the basic structure of the system, this we know about that the practical strategy of the attacker is to accelerate the brute force process by exploiting the parallel computation capability of GPU cores. We classify client puzzles into two types. If a puzzle functions *P*, as all the existing client puzzle schemes, is fixed and disclosed in advance, the puzzle is called a data puzzle; otherwise, it is referred to as a puzzle. To ensure

challenge data confidentiality and code security for an appropriate time period**.** After receiving the puzzle sent from the server, a client tries to solve the software puzzle on the host CPU, and replies to the server, as the conventional client puzzle scheme does.

VII.APPLICATION

A Network application is any application running on one host and provides a communication to another application running on a different host, the application may use an existing application layer protocols such as: HTTP(e.g. the Browser and web server), and may be the application does not use any existing protocols and depends on the socket programming to communicate to another application. So the web application is a type of the network applications.

We can distinguish between **network** and **stand-alone** applications. For example, if you use Microsoft Word to write a letter and save it on your PC, both the program and the data are stored on your computer. Since your computer does not have to be connected to a network, this is an example of a stand-alone application.

VIII.FUTURE ENHANCEMENTS

GPU-inflation attack, its idea can be extended to thwart DoS attackers which exploit other inflation resources such as Cloud Computing. For example, suppose the server inserts some anti-debugging codes for detecting Cloud platform into software puzzle, when the puzzle is running.

IX.CONCLUSION

In this article, we study camera-related vulnerabilities in Android phones for mobile multimedia applications. We discuss the roles a spy camera can play to attack or benefit phone users. We discover several advanced spy camera attacks, including the remote-controlled real-time monitoring attack and two types of passcode inference attacks. Meanwhile, we propose an effective defense scheme to secure a smartphone from all these spy camera attacks. In the future, we will investigate the feasibility of performing spy camera attacks on other mobile operating systems.

REFERENCES

1.Y. Zhou and X. Jiang, "Dissecting Android Malware: Characterization and Evolution", IEEE Symp. Security and Privacy 2012, pp. 95-109, 2012.

2.R. Schlegel, "Soundcomber: A Stealthy and Context Aware Sound Trojanfor Smartphones", NDSS, pp. 17-33, 2011.

3.N. Xu, "Stealthy Video Capturer: A New Video-Based Spyware in 3g Smartphones", Proc. 2nd ACM Conf. Wireless Network Security, pp. 69-78, 2009.

4.F. Maggi, "A Fast Eavesdropping Attack against Touchscreens", 7th Int'l. Conf. Info. Assurance and Security, pp. 320-25, 2011.

5.R. Raguram, "ispy: Automatic Reconstruction of Typed Input from Compromising Reflections", Proc. 18th ACM Conf. Computer and Commun. Security, pp. 527-36, 2011.

6."Android-eye", 2012.

7."Nanohttpd".

8.A. P. Felt and D. Wagner, "Phishing on Mobile Devices", Proc. WEB 2.0 Security and Privacy, 2011.

9.D. Li, D. Winfield and D. Parkhurst, "Starburst: A Hybrid Algorithm for Video-Based Eye Tracking Combining Feature-Based and Model-Based Approaches", IEEE Computer Soc. Conf. Computer Vision and Pattern Recognition Workshops, pp. 79, 2005.

10.P. Aldrian, "Fast Eyetracking", 2009.

JES
Journal of Engineering Sciences

Journal of Engineering Sciences

ISSN:0377-9254, Web:www.jespublication.com

CERTIFICATE OF PUBLICATION

This is to certify that the paper entitled

"Security Threats To Mobile Multimedia Applications"

Authored by

J SRIVIDYA

From

CMR TECHNICAL CAMPUS, Hyderabad

Has been published in

JES JOURNAL, VOLUME 13, ISSUE 06, JUNE- 2022

IMPACT FACTOR 6.54

ACADEMIC EXCELLENCE

UGC APPROVED

Dr. D. K. Umber

Editors-In-Chief

Dr. D.Karthikaran Umber

JES PUBLICATION

DOI: 10.15433/JES

crossref member

CROSSREF.ORG THE CITATION LINKING BACKBONE

JES
Journal of Engineering Sciences

Journal of Engineering Sciences

ISSN:0377-9254, Web:www.jespublication.com

CERTIFICATE OF PUBLICATION

This is to certify that the paper entitled

"Security Threats To Mobile Multimedia Applications"

Authored by

PHANI CH

From

CMR TECHNICAL CAMPUS, Hyderabad

Has been published in

JES JOURNAL, VOLUME 13, ISSUE 06, JUNE- 2022

IMPACT FACTOR 6.54

UGC APPROVED

Editors-In-Chief

Dr. D.Karthikaran Umber

JES PUBLICATION

DOI: 10.15433/JES

crossref member

CROSSREF.ORG THE CITATION LINKING BACKBONE

JES
Journal of Engineering Sciences

Journal of Engineering Sciences

ISSN:0377-9254, Web:www.jespublication.com

CERTIFICATE OF PUBLICATION

This is to certify that the paper entitled

"Security Threats To Mobile Multimedia Applications"

Authored by

ANIKETH D

From

CMR TECHNICAL CAMPUS, Hyderabad

Has been published in

JES JOURNAL, VOLUME 13, ISSUE 06, JUNE- 2022

IMPACT FACTOR 6.54

ACADEMIC EXCELLENCE

UGC APPROVED

Dr. D.K Umber

Editors-In-Chief

Dr. D.Karthikaran Umber

JES PUBLICATION

DOI: 10.15433/JES

crossref member

CROSSREF.ORG THE CITATION LINKING BACKBONE

JES
Journal of Engineering Sciences

Journal of Engineering Sciences

ISSN:0377-9254, Web:www.jespublication.com

CERTIFICATE OF PUBLICATION

This is to certify that the paper entitled

"Security Threats To Mobile Multimedia Applications"

Authored by

ROHITNIVAS B

From

CMR TECHNICAL CAMPUS, Hyderabad

Has been published in

JES JOURNAL, VOLUME 13, ISSUE 06, JUNE- 2022

IMPACT FACTOR 6.54

ACADEMIC EXCELLENCE

UGC APPROVED

Editors-In-Chief

Dr. D.Karthikaran Umber

JES PUBLICATION

DOI: 10.15433/JES

crossref member

CROSSREF.ORG
THE CITATION LINKING BACKBONE