DEVELOPMENT OF HIGH-INTERACTION HONEYPOTS

By

MARWA OMAR BABIKIR MOHAMMED AHMED

INDEX NO. 124096

Supervisor

Eng. Tarig Ahmed Khalid

A REPORT SUBMITTED TO
University of Khartoum
in partial fulfillment of the requirements for the degree of
B. Sc. (HONS) Electrical and Electronic Engineering
(ELECTRONICS SYSTEMS SOFTWARE ENGINEERING)
Faculty of Engineering
Department of Electrical and Electronics Engineering

October 2017
DECLARATION OF ORIGINALITY

I declare that this report entitled “Development of high-interaction honeypots” is my own work except as cited in the references. The report has not been accepted for any degree and it is not being submitted currently in candidature for any degree or other reward.

Signature: ____________________

Name: _______________________

Date: ________________________
ACKNOWLEDGMENT

Many thanks to Eng. Tarig for the project idea and the guidance. And I thank each one of my colleagues who supported me, and my relatives who looked forward for the completion of the project successfully.
ABSTRACT

Computer crimes are increasing rapidly and so the security community is always trying to catch up. A honeypots is a security tools which value lies in being attacked, honeypots are used to lure attackers to detect attacks and gather information about them.

Honeypot technologies developed fast through the last decade, high-interaction honeypots became more and more important with their stunning ability to gather information about the blackhat community, but they come with great risk and expense. This project aims to introduce a solution by creating a simple yet high-interaction honeypot.

The honeypot is a mirror website to which the attacker is directed after going through an authentication process, where he will think that he is on the real site but his actions are closely monitored and no harm is done to the real system.
الملخص

الجرائم الإلكترونية في ازدياد متواصل ولذلك فإن مجتمع الأمن المعلوماتي في سعي دائم للحاق بها. الهوني بوت او "قدر العسل" هو أداة حماية إلكترونية تقع فائدتها في قابلتها للاختراق, يستخدم الهوني بوت في اغراء الهاكر بغرض اكتشاف الهجمات وجمع المعلومات عنها.

تقنيات الهوني بوت تطورت سريعا خلال العقد الأخير, الهوني بوت ذو التفاعل العالي يصبح أكثر أهمية مع قدرته المذهلة على جمع المعلومات عن مجتمع القبعة السوداء, ولكنه يأتي مع مخاطر كبيرة وثمن باهظ. هذا المشروع يهدف إلى تقديم حل عن طريق تطوير هوني بوت بسيط ولكنه ذو تفاعل عالي.

الهوني بوت عبارة عن صفحة انترنت مطابقة للتي يريد الهاكر الدخول إليها, يتم توجيهه إليها بعد خوارزمية تحقق من الهوية, ويظن الهاكر أنه دخل إلى النظام الحقيقي بينما في الحقيقة يتم متابعة تحركاته وتسجيلها.
TABLE OF CONTENTS

DECLARATION OF ORIGINALITY ........................................................................... ii

ACKNOWLEDGMENT ............................................................................................. iii

ABSTRACT ............................................................................................................... iv

الملخص ..................................................................................................................... v

TABLE OF CONTENTS .......................................................................................... vi

LIST OF FIGURES ................................................................................................. vii

LIST OF ABBREVIATIONS .................................................................................... viii

CHAPTER ONE: Introduction .................................................................................. 1
  1.1 Overview .......................................................................................................... 1
  1.2 Problem Statement .......................................................................................... 1
  1.3 Objectives ........................................................................................................ 1
  1.4 Thesis Layout ................................................................................................... 2

CHAPTER TWO: Literature Review ......................................................................... 3
  2.1 Honeypots ........................................................................................................ 3
    2.1.1 What Are Honeypots? ................................................................................ 3
    2.1.2 Why Honeypots? ....................................................................................... 4
    2.1.3 History of Honeypots: ............................................................................... 5
    2.1.4 Types of Honeypots: ................................................................................ 6
      2.1.4.1 Production Honeypots: ................................................................. 6
      2.1.4.2 Research Honeypots: ................................................................. 6
      2.1.4.3 Low-Interaction Honeypots: ...................................................... 7
      2.1.4.4 High-Interaction Honeypots: ...................................................... 8
    2.2 Web Applications ............................................................................................ 8
LIST OF FIGURES

Figure 2-1 Different types of honeypots serving the same purpose.................................4
Figure 2-2 a tradeoff between the levels of interaction and complexity.........................8
Figure 2-3 components of a web application.................................................................9
Figure 2-4 software components for java servlet/JSP API..............................................10
Figure 3-1 Agile model.....................................................................................................11
Figure 3-2 Netbeans IDE...............................................................................................12
Figure 3-3 MySQL Workbench......................................................................................12
Figure 3-4 Software Flow Chart.....................................................................................13
Figure 3-5 Main page.......................................................................................................14
Figure 3-6 Login page......................................................................................................14
Figure 3-7 Home page.....................................................................................................15
Figure 3-8 Mirror page.................................................................................................15
Figure 4-1 login test 1....................................................................................................17
Figure 4-2 login test 1 result – success.........................................................................18
Figure 4-3 login test 2....................................................................................................18
Figure 4-4 login test 2 result – failure..........................................................................19
Figure 4-5 authentication test 1 result – granted..........................................................19
Figure 4-6 authentication test 2 result – redirected......................................................20
# LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDS</td>
<td>Intrusion Detection System</td>
</tr>
<tr>
<td>DMZ</td>
<td>Demilitarized Zone</td>
</tr>
<tr>
<td>DTK</td>
<td>Deception Toolkit</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>HTML</td>
<td>Hyper Text Markup Language</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hyper Text Transfer Protocol</td>
</tr>
<tr>
<td>CSS</td>
<td>Cascading Style Sheets</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
<tr>
<td>JSP</td>
<td>Java Server Pages</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>JDK</td>
<td>Java Development Kit</td>
</tr>
<tr>
<td>RDBMS</td>
<td>Relational Database Management System</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>JDBC</td>
<td>Java Database Connectivity</td>
</tr>
</tbody>
</table>
CHAPTER ONE

Introduction

1.1 Overview:

A honeypot is a computer security mechanism set up to detect unauthorized use of information systems. High-interaction honeypots imitates the activities of the production systems that hosts a variety of services and, therefore, an attacker is allowed a lot of services to interact with and waist his time. High-interaction honeypots are complex, because of the vast amount of services they provide, they should not be detected and should provide a great amount of information about the attack. With this being said; high-interaction honeypots are extremely difficult to develop. However, in this project, we will focus on a specific goal, which is redirecting potential unauthorized access to sites with sensitive information to a mirror site, where the attacker is fooled that he is on the real site and information can be gathered about him.

1.2 Problem Statement:

There is no severe problem that this project provide a solution to. However, this doesn’t mean that it has no importance. There are many security tools that can be used to protect systems and websites, and there are many available commercial high-interaction honeypot products, but every situation has its own measures, and what work with some systems might not be a good choice for others. As stated above, high-interaction honeypots are very difficult to develop and install, and commercial ones come with high price. Even free high-interaction honeypots consumes more resources, and a tradeoff should be made between the resources dedicated for them and the actual benefit provided by the honeypot. The project tries to offers a simpler and cheaper solution.

1.3 Objectives:

The main objectives of the project are:
Building a small website that is able to provide access based on a predefined authentication method, and redirect unwanted users to a mirror website which resembles it, this mirror website works as a honeypot.

To introduce the student to the concept of honeypots, the web application development and other security concepts.

### 1.4 Thesis Layout:

This thesis is organized into five chapters and one appendix as follows:

Chapter one gives a brief introduction about the project and its objective.

Chapter two discusses the literature of the disciplines this project concerned about, how it evolved through years and its current statuses. It goes through honeypots, their history and web application development basics.

Chapter three describes the implementation of the project, going through the software development model and construction process.

Chapter four shows the testing phase and results obtained, also a brief discussion is made regarding them.

Chapter five, the last chapter, provides the completion status of the project, future work and limitations.

Appendix A shows the source code of the application.
CHAPTER TWO

Literature Review

2.1 Honeypots:

2.1.1 What Are Honeypots?

There are many definitions of honeypots, because they can take on different manifestations. A definition that can be applicable to all the honeypots is:

A honeypot is a security resource whose value lies in being probed, attacked or compromised. [1]

Honeypots can be used to detect attacks, deter them or capture and analyze information about hackers and worms. They come in many different forms and can achieve many different goals, they can be anything from a router, a script, a port listener to an actual system deployed in the network specifically to be compromised. But all of them –unless probed or attacked- has no value. Most security tools are designed to address a specific problem. For example, block unauthorized activity –firewalls- or detect attacks in the network –IDS-, but we need to know how the hacker actually attacks the system to fix its vulnerabilities, and this is the role of honeypots. [2]
2.1.2 Why Honeypots?

Organizations gather a vast amount of data every day which can be overwhelming, and usually it’s very difficult to get any useful information out of it. Since honeypots have no production value, any interaction with them is considered to be a threat. They gather a small amount of data with high importance.

While they work as a security tools, the greatest benefit of honeypots lies in the research area. They are a powerful tool to silently gather information about the blackhat community. Those do not contribute directly in the security of a specific organization, instead they are used to research the threats that the organizations face and thus learn how to better protect against those threats.
2.1.3 History of Honeypots:

Honeypots where used before 1990, within military, government and some commercial organizations. But nothing was a public knowledge until 1990. First book written was by Clifford Stoll, titled *The Cuckoo’s Egg* [3]. The second is the whitepaper "An Evening with Berferd in Which a Cracker Is Lured, Endured, and Studied" [4].

DTK was the first public honeypot solution, it was developed by Fred Cohen and published in 1997. It was one of the first free honeypot solutions for everyone to download and try. The following list summarize the key events in the history of honeypots: [1]


- **1997** - Version 0.1 of Fred Cohen's Deception Toolkit was released, one of the first honeypot solutions available to the security community.

- **1998** - Development began on CyberCop Sting, one of the first commercial honeypots sold to the public. CyberCop Sting introduces the concept of multiple virtual systems bound to a single honeypot.

- **1998** - Marty Roesch and GTE Internetworking begin development on a honeypot solution that eventually becomes NetFacade. This work also begins the concept of Snort.

- **1998** - BackOfficer Friendly is released: a free, simple-to-use Windows-based honeypot that introduced many people to honeypot concepts.

- **1999** - Formation of the Honeynet Project and publication of the "Know Your Enemy" series of papers [5]. This work helped to increase the awareness and validate the value of honeypots and honeypot technologies.
2000/2001- Use of honeypots to capture and study worm activity. More organizations adopting honeypots for both detecting attacks and for researching new threats.

2002 - A honeypot is used to detect and capture in the wild a new and unknown attack, specifically the Solaris dtspcd exploit.

2.1.4 Types of Honeypots:

Honeypots can be classified according to the purpose or the level of interaction. According to purpose:

2.1.4.1 Production Honeypots:

They are systems that help mitigating the risk in the organization. They work together with other tools to provide better security. We will examine the value they add to the three categories of security which are defined by Bruce Schneier in “Secrets and Lies” [6].

**Prevention:** if attackers knew that an organization is using honeypots, they might be scared off. But this add a little value to prevention, and thus there is no need to spend time and resources in deploying honeypots just for the sake of prevention.

**Detection:** while honeypots add a little value to prevention, they add extensive value to detection. They are very useful in avoiding false positives -when systems falsely alert suspicious or malicious activity- and false negatives -when systems fails to detect an attack-. They are also the best choice to avoid data aggregation, which means having a vast amount of data with no valuable information.

**Response:** When a honeypot is compromised, the only real activity on the system is the activity of the attacker, helping to maintain its integrity. This helps in collecting evidences and avoiding data pollution, then planning for response strategies.

2.1.4.2 Research Honeypots:

One of the greatest challenges the security community faces is the lack of information on the enemy. The intelligence and counterintelligence community spend billions of dollars
on information-gathering capabilities because knowledge is such a critical asset. Research honeypots offer extensive value in information gathering by giving us a platform to study cyber threats. What better way to learn about the bad guys than to watch them in action, to record step by step as they attack and compromise a system?

Research honeypots can be used for the following:

- To capture automated threats, such as worms or auto-rooters.
- As an early warning mechanism, predicting when future attacks will happen. Data collected from research honeypots can be used for statistical modeling, predicting future attacks. Attacks can then be identified and stopped before they happen.
- To capture unknown tools or techniques.
- To better understand attacker’s motives and organization.
- To gain information on advanced blackhats.

According to the level of interaction:

**2.1.4.3 Low-Interaction Honeypots:**

Those are mostly production honeypots deployed to increase the security of a specific organization. They are easy to install, configure and maintain because of their simple design and basic functionality. They emulate a variety of services and the attacker is limited to interacting with these predesigned services. They can’t provide detailed information about the attacks, and are limited to transactional information such as the date of the attack, the source and destination port and IP. Examples of low-interaction honeypots are BackOfficer Friendly, specter and honeyed.

Although their functionalities are limited, low-interaction honeypots can still be of a great use in detecting attacks, and thus can be used as an additional tool along with other security tools. They are also good as a starting point for individuals or organizations who have never
worked with honeypots before. However, they provide very little information about the attack and can’t be used in the research area, where high-interaction honeypots come in use.

2.1.4.4 High-Interaction Honeypots:

As their name implies, they provide the attacker with a full access to real systems with no restrictions. They provide a vast amount of information about the attack tools and the attacker techniques, they are the extreme of honeypot technologies. They are a very powerful weapon, providing limitless possibilities, they discover new tools and identify new vulnerabilities in operating systems or applications. With all these abilities however, they come with a great risk, and they are extremely difficult and time consuming to create, install and maintain. Examples of high-interaction honeypots are ManTrap and honeynets.

<table>
<thead>
<tr>
<th>Level of Interaction</th>
<th>Work to Install and Configure</th>
<th>Work to Deploy and Maintain</th>
<th>Information Gathering</th>
<th>Level of Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Easy</td>
<td>Easy</td>
<td>Limited</td>
<td>Low</td>
</tr>
<tr>
<td>Medium</td>
<td>Involved</td>
<td>Involved</td>
<td>Variable</td>
<td>Medium</td>
</tr>
<tr>
<td>High</td>
<td>Difficult</td>
<td>Difficult</td>
<td>Extensive</td>
<td>High</td>
</tr>
</tbody>
</table>

*Figure 2-2 a tradeoff between the levels of interaction and complexity.*

2.2 Web Applications:

2.2.1 Definitions:

1- A web application is a set of web pages that are generated in response to user requests. There are many types of web applications, such as search engines or online stores [7]. A web application consists of a client and server communicating with each other using a specific protocol. The client send requests to and get responses from the server using the web browser.
2- Static web pages are pages that stay the same each time they are viewed, they don’t change in response to user’s input.

3- Dynamic web pages are HTML documents that are generated by a web application. They change according to parameters sent to the web application by the web browser.

4- HTML is the language that the web browser converts into the web pages of a web application.

5- HTTP is the protocol that web browsers and web servers use to communicate.

6- CSS is a stylesheet language used to describe the presentation of a document written in HTML or XML.

![Figure 2-3 components of a web application](image)

2.2.2 Java Servlets and JSP:

It is the java API for web applications. Servlets store the java code that does the server-side processing, while JSPs store the HTML that defines the user interface. Java servlet/JSP API gives the developer a high degree of control over the HTML, CSS and JavaScript.

The software components to work with java servlet/JSP API are: A web server that runs a servlet/JSP engine -also known as servlet/JSP container-, the JDK which contains the java complier and the core classes for working with java. Web applications which store their data in a database need a database server too.
2.2.3 Apache Tomcat server:

The Apache Tomcat® software is an open source implementation of the Java Servlet, Java Server Pages, Java Expression Language and Java WebSocket technologies [8].

It is the most popular server for java web applications. It runs on almost all the modern operating systems, and it includes a web server named Coyote and a servlet/JSP engine named Catalina.

2.2.4 MySQL Database:

MySQL is an open source RDBMS that uses SQL, which is the most popular language for adding, accessing and managing content in a database. Many java web applications use the MySQL server to manage database [9].
CHAPTER THREE

Methodology

3.1 Software Development Tools:

3.1.1 Agile Development Model:

Agile development model was used in the process of building this small software, it is a type of Incremental model. Software is developed in incremental, rapid cycles. This results in small incremental releases with each release building on previous functionality. Each release is thoroughly tested to ensure software quality is maintained. It is used for time critical applications [10].

![Agile Model Diagram](image)

*Figure 3-1 Agile model*

3.1.2 Development IDE:

Netbeans IDE was used, it is an open source software development platform written in java, it is primary intended for the development of java applications but can also support other languages [11].

To develop web applications using Netbeans, java web plugin must be installed and the server which will be used to load the application must also be installed in Netbeans.
For the database, MySQL Workbench was used. It’s a user friendly IDE which works with MySQL server to create and manage database.
3.2 Software Flow Chart:

![Software Flow Chart]

3.3 Construction Process:

3.3.1 Creation of Static HTML pages:

The application consists of 4 pages: welcome page, login page, home page and home mirror page. All written in HTML. The pages are shown in the figures below.
Figure 3-4 Main page

Figure 3-5 Login page
Figure 3-6 Home page

Figure 03-7 Mirror page
3.3.2 Creation of java servlets and classes:

For the website to be dynamic, JSPs only won’t be enough. The login page take information from the user, so it’s mapped with a login servlet which compares it with the database table entries. If they matched, it goes to the second authentication phase with the IP address. The user is granted access after his IP address is accepted and the attributes are passed to an HTTP session.

There is also the logout servlet, which deletes the attributes from the HTTP session and take the user to the login page again.

For the database connection, a java class which loads the JDBC driver is created. The class maps the database connection to the web application using the user name and password of the connection.

3.3.3 How The Authentication Works:

The idea behind the IP authentication is as follows: suppose that a user usually logs in from Sudan, and if any user changed location he informs the bank beforehand. If a user has the correct login information but a different location –i.e. different IP range- it can mean a potential identity theft attack. Thus the user is redirected to a mirror site that works as a high-interaction honeypot. It has the same layout and functionality of the Home website. And the user actions are monitored.
CHAPTER FOUR

Results and Discussion

4.1 Testing:

4.1.1 Testing Environment:

PC used is a Dell Latitude E6410 with 4GB of RAM, 2.40GHz Intel Core i5 CPU, 64-bit architecture and windows 7 operating system. The browser is Firefox version 56.0.

4.1.2 Testing Procedure:

Login testing:

- With an existing account in the database:

Figure 04-1 login test 1
Figure 04-2 login test 1 result - success

- With an account that doesn’t exist:
IP authentication testing:
- Setting the IP range to 192.168:
- Setting IP range to 2.16:

![Successfully logged in](image)

*Figure 04-6 authentication test 2 result - redirected*

### 4.2 Result and Discussion:

As seen in the testing, the software is working as intended with no problems. This software is too simple to be deployed in a real system. It just presents the basic concept and idea behind the project goal. And it introduce the student to web application development and security some concepts.
CHAPTER FIVE

Conclusion and Future Work

5.1 Conclusion:

The two main objectives of the project, which are building a small website with an authentication method, and introducing the student to the honeypot technology was met successfully.

5.2 Limitations:

The software only offers a login scenario with a simple authentication method, to present a basic idea. However, to be used in any real system, it must be upgraded.

5.3 Future Work:

As stated above, the project is just the first step and need to be upgraded as follows:

The website must be a complete online banking site, which users can use for money transfer or online shopping.

Rather than just the location, the authentication method must use machine learning algorithms to detect the patterns of users, so that potential hackers can be identified.
References


Appendix A: Software Source Code:

Main page: Index.html

```html
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="index_style.css">
    <title>eBanking</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  </head>
  <body>
    <h1 id="welcome">Welcome to eBanking</h1>
    <h2><a href="LogIn.jsp" id="loginlink">Log In</a></h2>
  </body>
</html>
```
Login page: LogIn.jsp

```html
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" href="index_style.css">
    <title>Banking - Log In</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
    <h1 id="login-header">Login</h1>
    <form action="LogIn" method="post">
        Username : <input type="text" name="name" required>
        <br><br>
        Password : <input type="password" name="pass" required>
        <br><br>
        <input type="submit" value="Log In" name="login">
    </form>
</body>
</html>
```
Home page: Home.jsp

```html
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="index_style.css">
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Home</title>
  </head>
  <body>
    <% 
      if(session.getAttribute("name") == null) {
        response.sendRedirect("LogIn.jsp");
      }
    %>
    <h2>Successfully logged in</h2>
    <h3>Enjoy our online banking service</h3>
    <form action="LogOut">
      <input type="submit" value="Log Out" name="logout">
    </form>
  </body>
</html>
```
Mirror page: Home_mirror.jsp

```html
<%@ page contentType="text/html" pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="index_style.css">
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Home Mirror</title>
  </head>
  <body>

  <% if(session.getAttribute("name") == null){
    response.sendRedirect("LogIn.jsp");
  } %>

  <h2>Successfully logged in</h2><br>
  <h3>Enjoy our online banking service</h3><br>
  <form action="LogOut">
    <input type="submit" value="Log Out" name="logout">
  </form>

  </body>
</html>
```
CSS file: index_style.css

/*
Author: marwa
*/
body{
  font-family: sans-serif;
  height: 450px;
  width: 400px;
  margin-left: auto;
  margin-right: auto;
  padding: 50px 10px 10px 10px;
  border-color: black;
  border-width: 2px;
  border-style: solid;
  text-align: center;
}
#welcome {color:#006666}
#loginlink {color:#6699ff;}
#login-header {color:#006666}
Login servlet: LogIn.java

```java
import java.io.IOException;
import java.util.*;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/LogIn")
public class LogIn extends HttpServlet {

    String ip_range = "192.168";

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String name = request.getParameter("name");
        String pass = request.getParameter("pass");

        //verify access
        MySqlConnection dao = new MySqlConnection();
        if(dao.loginTrue(name, pass)){
            String ip = request.getRemoteAddr();
            StringTokenizer toke = new StringTokenizer(ip, ".");
            int dots = 0;
            String byte1 = "";
            String byte2 = "";
            String client;
            while (toke.hasMoreTokens()){  
                ++dots;
                if(dots == 1){
                    byte1 = toke.nextToken();
                }else{
                    byte2 = toke.nextToken();
                    break;
                }
            }
            client = byte1 + "." + byte2;

            if(client.equals(ip_range)){
                HttpSession session = request.getSession();
                session.setAttribute("name", name);
                response.sendRedirect("Home.jsp");
            }else{
                HttpSession session = request.getSession();
                session.setAttribute("name", name);
                response.sendRedirect("Home_mirror.jsp");
            }
        }else{
            response.sendRedirect("LogIn.jsp");
        }
    }
}
```
Logout servlet: LogOut.java

```java
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebServlet(value="/LogOut")
public class LogOut extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        HttpSession session = request.getSession();
        session.removeAttribute("name");
        session.invalidate();
        response.sendRedirect("LogIn.jsp");
    }
}
```
Database connection: MysqlConnection.java

```java
import java.sql.*;

public class MysqlConnection {

    String query = "select * from accounts where userName=? and userPass=?";
    String url = "jdbc:mysql://localhost:3306/ebanking";
    String accountName = "root";
    String mysqlPass = "#mysql**t57";

    public boolean loginTrue(String userName, String userPass) {
        try{
            Class.forName("com.mysql.jdbc.Driver");
            Connection con = DriverManager.getConnection(url, accountName, mysqlPass);
            PreparedStatement st = con.prepareStatement(query);
            st.setString(1, userName);
            st.setString(2, userPass);
            ResultSet rs = st.executeQuery();
            if(rs.next()){
                return true;
            }
        }catch(Exception e){
            e.printStackTrace();
        }
        return false;
    }
}
```