

Comparison And Implementation Of Random4 Algorithm And Hirschberg Algorithm Using Open Source Software For Prevention Of SQL Injection Attack

¹Mohammed Ahmed, ²Sayyed Saima, ³Shaikh Shagufta, ⁴Shaikh Tazreen
(Department of Computer Engineering)

M.H. Saboo Siddik College of Engineering, Clare Road, Byculla, Mumbai-400008, India.

ABSTRACT

SQL injection attacks are easy way to attack the web applications and gain access to the private data in a database. Using different types of the SQL attacks one can easily gain access, modify the database or can even remove it. Details such as fields and table names are required for a hacker to modify a database. Hence, to provide the increased amount of security to users and their data, different types of techniques are used such as Random4 algorithm, which is based on randomization and used to encrypt the user input, Hirschberg algorithm which is a divide and conquer technique used to match the query pattern. In this paper we are providing a comparative study and implementation of these prevention techniques using open source software.

Keywords

SQL injection, Random4, Hirschberg.

1. INTRODUCTION

According to the White Hat report on the web security vulnerabilities in the year 2011 it is shown that nearly 14-15 % of web application attacks account for SQL Injection [1].

SQL Injection Attack is a type of code-injection attack [2] which penetrates in the web applications and gets illegal access to the databases, it is very easy for the attackers to inject the code in the web page and gain access to all the databases, or to modify or even delete the databases. This type of attack is mainly caused due to improper validation of user input. [3].

With the leading attacks on the web applications it is very important to prevent them. So, different techniques are used for the prevention, named as Random4 Algorithm which is based on randomization. It will take user input and encrypt it using Random4 Look up table, the encryption of the current characters will be based on the next character. Second technique is Hirschberg Algorithm which is a divide and conquer version of the Needleman-Wunsch algorithm [4] and is based on pattern matching of the query. Any input given to the web application goes in the form of query, so using Hirschberg one standard query format will be set and if the received pattern of

the query from web page is not matched with the set query pattern then no one can get access to the web page or the databases.

The following prevention techniques are chosen for comparison and implementation named as Random4 Algorithm and Hirschberg Algorithm.

IMPLEMENTED ALGORITHMS:

I– RANDOM4 ALGORITHM:

```
Input:
input String inp[]
Output: Encrypted String en[]
N: Length of inp[]
R[]: Random values of character
For i=1 to N
  If(inp[i+1]==null || inp[i+1]=lowercase)
  Then en[i]=R[lower]
End { if }
Else If( inp[i+1]=uppercase)
  Then en[i]=R[upper]
End { Else if }
Else If( inp[i+1]=number)
  Then en[i]=[number]
End { Else if }
Else If( inp[i+1]=special_character)
  Then en[i]=[special]
End { Else if }
End { For }
Return en[]
```

Random4 is based on the randomization and is used to convert the input into the cipher text. Any input in the web forms will be containing numbers, uppercase, lowercase or special characters. Using the look up table, the user input will be converted into cipher text. Each character in the input can have 72 combinations. Hence there will be 726 combinations possible for 6 character inputs possible. To encrypt the input, each input character is given four random values. A sample look up table is given below. The first column in the table is for lower case, second is for upper case, third is for numbers and fourth is for special characters now based on the next input character, one of these four values is substituted for a given character. For example, if the input is provided in the form of username as xY@2 then the first character 'x' is given four Random values in the look up table based on the next input character which is upper case here, the value "R[Upper]" is chosen.

If the next character was lower case then the value "R[lower]" is chosen. If the next character was number, the value "R[number]" is chosen. If the next character was special character then value "R[Special]" is chosen.

I	R[Lower]	R[Upper]	R[Number]	R [Special]
A	F	#	4	R
....				
Z	J	5	%	D
A	T	A	S	8
....				
Z	&	Q	3	o
0	U	\$	B	.
....				
9	6)	^	g
@	-	2	C	(
....				
-	K	#	L	4

Table 1: Look Up table

II- HIRSCHBERG ALGORITHM:

Hirschberg’s algorithm is a divide and conquer version of Needleman-Wunsch algorithm. In this algorithm one standardized form of an SQL Query is set for the login.

This algorithm works on the logic 1 and 0. Each word in the login query is considered as a column and row wise for checking. When the user enters the input in the web page, it goes into the form of query. Each word of the query will be verified with the predefined set query and if the query word is perfectly matched then it will be marked as 1 otherwise 0. If in the whole table a single 0 is found then it will be considered as an attack and system will block the access.

For example if the username and password entered as ‘Soni’ and ‘1234’ respectively. Then the query will be formed as

```
SELECT * FROM logintable WHERE username='Soni' AND password='1234';
```

In this query each format of the query will be matched and it will be considered as a correct user.

Now suppose the SQL injection query

Username: soni and password: ‘abcd’ OR ‘1’=’1

Here after ‘abcd’ it will be detected as SQL injection because after password’s quotes (“”) it will not get AND statement instead it will get OR statement which is not in the predefined query. So in this way the attack is prevented.

(I,J)	Select	*	From	Logintable	Where	User Name	=	'	'	And	Password	=	'	'	'	;
Select	1															
*		1														
From			1													
Logintable				1												
Where					1											
User Name						1										
=							1									
'								1								
				DIVIDE	AND	CONQUER										
'									1							
And										1						
Password											1					
=												1				
'													1			
															1	
'																
;																1

Table 2: Look Up table

3.IMPLEMENTATION:

I – RANDOM4 ALGORITHM:

```

package com.java.utility;
public class Random4 {

    // a-z 26 A-Z 26 0-9 10 Special 25 Total 87
    public char[] mainList=
    {
        'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z',

        'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z',
        '0','1','2','3','4','5','6','7','8','9',
        '!','@','#','$','%','^','&',*','(',')','-','\|',;','"','\'';
    };
    public char[] firstList=
    {
        '!','\|',;','"','\'';','-','(',')','(','*','&',^','%','$','#','@','!',
        '9','8','7','6','5','4','3','2','1','0',

        'Z','Y','X','W','V','U','T','S','R','Q','P','O','N','M','L','K','J','I','H','G','F','E','D','C','B','A',
        'z','y','x','w','v','u','t','s','r','q','p','o','n','m','l','k','j','i','h','g','f','e','d','c','b','a'
    };
}

```

```

public char[] secondList=
    {
        '0','1','2','3','4','5','6','7','8','9',
        '!', '@', '#', '$', '%', '^', '&', '*', '(', ')', '-', '\\', ';', ':', '"', '\'',
        'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z',

        'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z',
    };
public char[] thirdList=
    {

        'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm',

        'z', 'y', 'x', 'w', 'v', 'u', 't', 's', 'r', 'q', 'p', 'o', 'n', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M',
        '9', '8', '7', '6', '5', '4', '3', '2', '1', '0',
        '!', '\'', '"', ';', '\\', '-', ')', '(', '*', '&', '^', '%', '$', '#', '@', '!'

    };
public char[] fourthList=
    {

        '5', '6', '7', '8', '9', '0', '1', '2', '3', '4',

        'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M',
        '!', '@', '#', '$', '%', '^', '&', '*', '(', ')', '-', '\\', ';', ':', '"', '\'',

        'm', 'l', 'k', 'j', 'i', 'h', 'g', 'f', 'e', 'd', 'c', 'b', 'a', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'
    };

//Encryption module
public String encrypt(String data)
{
    String encdata = "";
    for(int i=0;i<data.length();i++)
    {
        char get = data.charAt(i);
        String getcase = "lower";
        if(i!=data.length()-1)
        {
            char findcase = data.charAt(i+1);
            getcase = getCase(findcase);
        }
        int position = getPosition(get);
        if(position == -1)
        {
            encdata = encdata + get;
        }
        else
        {

```

```

        if(getcase.equals("lower"))
        {
            encdata = encdata + firstList[position];
        }
        else if(getcase.equals("upper"))
        {
            encdata = encdata + secondList[position];
        }
        else if(getcase.equals("number"))
        {
            encdata = encdata + thirdList[position];
        }
        else if(getcase.equals("special"))
        {
            encdata = encdata + fourthList[position];
        }
        System.out.println("Case : "+getcase+" "+get+" a : "+encdata);
    }
}

return encdata;
}
public String getCase(char ch)
{
    String charcase = "";
    int val = (int)ch;

    if(val>64 && val<91)
    {
        charcase = "upper";
    }
    else if(val>96 && val<124)
    {
        charcase = "lower";
    }
    else if(val>47 && val<58)
    {
        charcase = "number";
    }
    else
    {
        charcase = "special";
    }

    return charcase;
}

public int getPosition(char ch)
{
    int position = -1;

```

```
        for(int i=0;i<mainList.length;i++)
        {
            if(ch == mainList[i])
            {
                position = i;
                break;
            }
        }
        return position;
    }
}
```

II – HIRSCBERG ALGORITHM:

```
package com.java.utility;
import java.util.StringTokenizer;
import java.util.Vector;

public class Hirshberg {
boolean analysis = false;//not allowed
//true allowed
Vector<Object> xAxis;
Vector<Object> yAxis;
Vector<Object> result;
Vector<Object> output;
String applicationQuery;

// Setting the predefined Query pattern

public Hirshberg() {
    xAxis = new Vector<Object>();
    yAxis = new Vector<Object>();

    keywords();
}

public void keywords() {

    xAxis.add("SELECT");
    xAxis.add("*");
    xAxis.add("FROM");
    xAxis.add("LoginTable");
    xAxis.add("WHERE");
    xAxis.add("username");
    xAxis.add("=");
    xAxis.add("\");
    xAxis.add("");
    xAxis.add("\");
    xAxis.add("AND");
    xAxis.add("password");
}
```

```

xAxis.add("=");
xAxis.add("\");
xAxis.add("");
xAxis.add("\");
xAxis.add(";");
}

//pattern matching module
public boolean Checker(String query) {
    try {
        result = new Vector<Object>();
        applicationQuery = query.trim();
        StringTokenizer st = new StringTokenizer(applicationQuery, " ");
        while (st.hasMoreElements()) {
            String e = st.nextToken();
            if (!e.contains("")) {
                result.add(e);
            } else {
                String r = e;
                StringTokenizer stk = new StringTokenizer(r, "");
                while (stk.hasMoreElements()) {
                    result.add("");
                    result.add(stk.nextElement());
                    result.add("");
                }
            }
        }
    }

    System.err.println(result.size());
    for (int z = 0; z < result.size(); z++) {
        if (result.elementAt(z).toString().equals("")) {
            if (result.elementAt(z + 2).toString().equals("")) {
                result.setElementAt("", (z + 1));
                z = z + 3;
            }
        }
    }
    System.out.println(xAxis);
    System.err.println(result);

    //Analysis Module

    output = new Vector<Object>();

    int count = 0;
    for (int i = count; i < xAxis.size(); i++) {
        for (int j = count; j < result.size(); j++) {
            String x = xAxis.elementAt(i).toString();
            String y = result.elementAt(j).toString();

```



```
System.out.println("x== " + x + " y== " + y);
if (x.equalsIgnoreCase(y)) {
    output.add("1");
    analysis = true;
    count++;
    break;
} else {
    analysis = false;
    count++;
    break;
}
}
if (!analysis) {
    break;
}
}
} catch (Exception e) {
    e.printStackTrace();
}
return analysis;
}
}
```

4. RESULTS:

⇒ Normal Login

Online Banking
Security Awareness Program

Home About Services Contact Login Register

Login Page

Technique: Normal Login

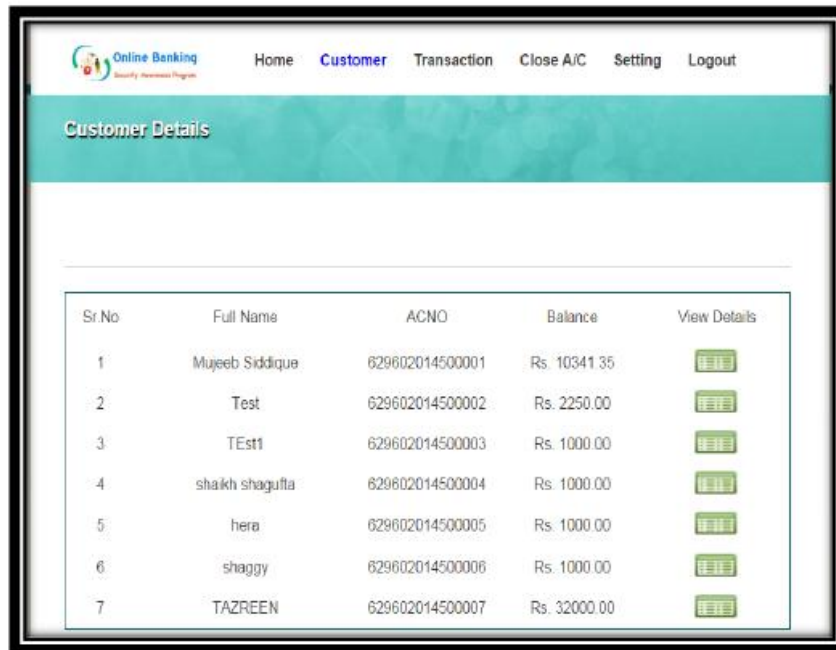
Username: QR 1=1--

Password: *****

Submit

Not registered? Click [Here](#) to Register

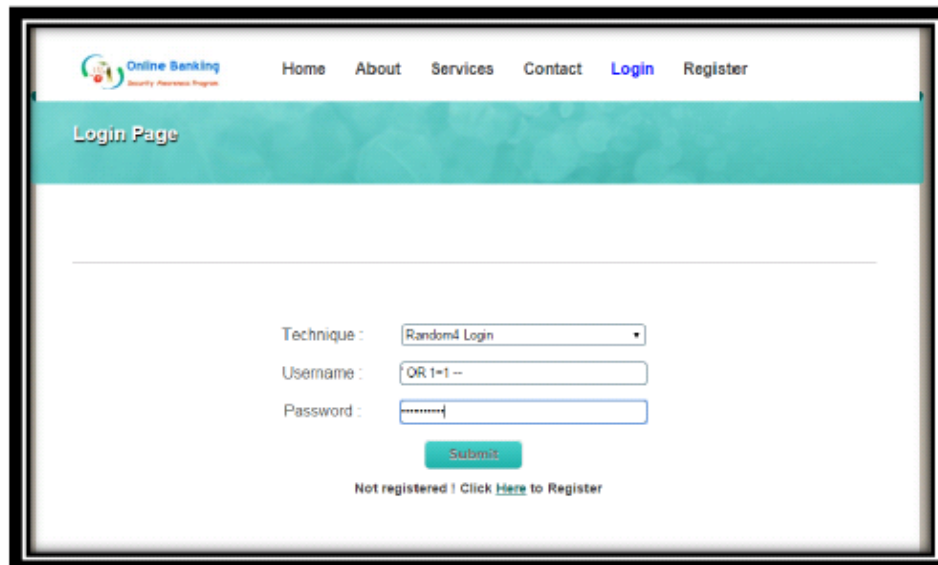
⇒ **Attacker gets illegal access to database.**



The screenshot shows the 'Customer Details' page of an online banking system. The page has a teal header with the 'Online Banking' logo and navigation links: Home, Customer, Transaction, Close A/C, Setting, and Logout. Below the header is a table with the following data:

Sr.No	Full Name	ACNO	Balance	View Details
1	Mujeeb Siddique	629602014500001	Rs. 10341.35	
2	Test	629602014500002	Rs. 2250.00	
3	TEst1	629602014500003	Rs. 1000.00	
4	shaikh shagufta	629602014500004	Rs. 1000.00	
5	hera	629602014500005	Rs. 1000.00	
6	shaggy	629602014500006	Rs. 1000.00	
7	TAZREEN	629602014500007	Rs. 32000.00	

⇒ **Login using Random4 Algorithm**



The screenshot shows the 'Login Page' of an online banking system. The page has a teal header with the 'Online Banking' logo and navigation links: Home, About, Services, Contact, Login, and Register. Below the header is a login form with the following fields and options:

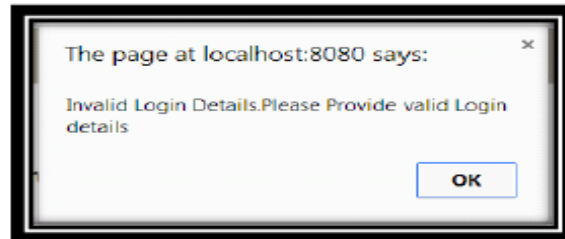
Technique :

Username :

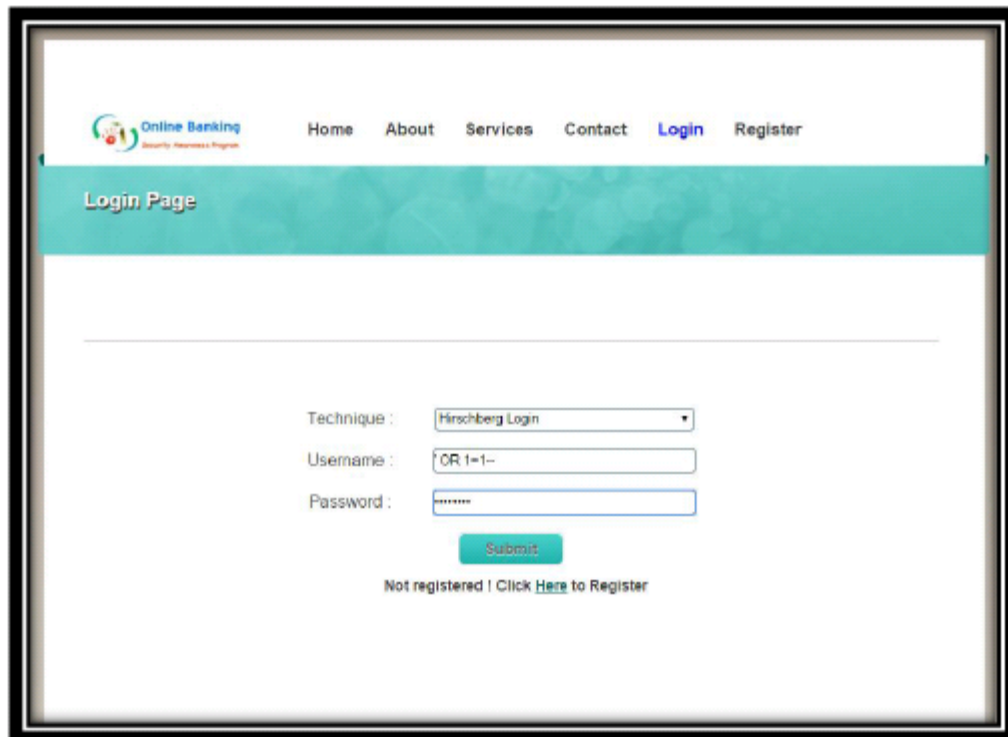
Password :

Not registered ! Click [Here](#) to Register

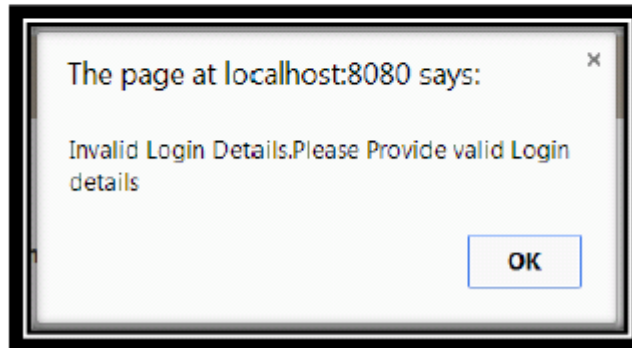
➔ **Attacker doesn't get access to database.**



⇒ **Login using Hirschberg Algorithm**



➔ **Attacker doesn't get access to database.**



5. COMPARATIVE STUDY:

Parameters	Hirschberg	Random4
Proxy Overhead	No	No
Time Complexity	Less	More
Space Complexity	Less	Bit more
Encryption	No encryption (query pattern is matched only)	User Input
Computational Overhead	Less	Bit more

Table 3: Comparative Study

6. CONCLUSION:

This paper presents the methods for preventing the web applications from SQL injection attack. And by the above comparative study, we have concluded that Hirschberg algorithm is easy to implement, it has less time and space complexity and no proxy or computational overhead. On the other side Random4 is also a powerful technique because of encryption but due to that time

and space complexity is more than Hirschberg Algorithm. It is easy to implement, and it has no proxy or computational overhead.

REFERENCES:

- [1] <https://www.whitehatsec.com/resource/stats.html>
- [2] W.G halfond and A. orso AMNESIA: Analysis and monitoring for neutralizing SQL injection attacks.in proceedings of the IEEE and ACM International Conference on Automated Software Engineering (ASE-2005), Long Beach, CA, USA, NOV 2005.
- [3] Random4: An Application Specific Randomized Encryption Algorithm to prevent SQL Injection Avireddy, S. Dept. of Inf. Technol., Anna Univ. Chennai, India Perumal, V. ; Gowraj, N. ; Kannan, R.S. ; Thinakaran, P. ; Ganapathi, S..Gunasekaran, J.R. Prabhu,S. IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. 60, NO.5, MAY 2012
- [4] 2009 IEEE International Advance Computing Conference (IACC 2009) Patiala, India, 6-7 March 2009 Combinatorial Approach for Preventing SQL Injection Attacks ,R. Ezumalai, G. Aghila Department of Computer Science, Pondicherry University, Puducherry, India.

AUTHORS:

Mohd. Ahmed is currently Assistant professor at the M.H. Saboo Siddik College of Engineering and has completed his M.E. from Vidyalkar Institute of Technology (VIT), Mumbai and formerly trainee hardware & network engineer in DSS Mobile communication, Mumbai. He is having 9 years teaching and 1 year industry experience.



Shaikh Tazreen is currently pursuing B.E. from M.H. Saboo Siddik College of engineering, Mumbai.



Shaikh Shagufta is currently pursuing B.E. from M.H. Saboo Siddik College of engineering, Mumbai.



Sayed Saima is currently pursuing B.E. from M.H. Saboo Siddik College of engineering, Mumbai.

