Sensitive Data Exposure Prevention using Dynamic Database Security Policy

Jignesh Doshi
LJ Institute of Management Studies,
Ahmedabad, Gujarat,India

Bhushan Trivedi
GLS Institute of Computer Technology,
Ahmedabad,Gujarat, India

ABSTRACT
Today, the internet has become a 4th necessity for human after air, water and food. The internet is widely used for business. Now-a-days, billions of transactions are done online with the use of different applications. In today's world threats to security is becoming more and more sensitive issue, lots of attacks have taken place in recent years. It is observed that there are many leakages in the security of web applications. Five attacks out of top ten attacks are done using Structured Query Language (SQL). Database attacks mostly affect on Data Theft, Data manipulation and by Pass user authentication. Our study focus is to prevent sensitive data exposure. The authors have proposed dynamic database security policy to prevent sensitive data exposure using Oracle database

General Terms
Database Security Policy.

Keywords
SQL, SQL Injection, Database theft, Sensitive Data exposure, Security policy

1. INTRODUCTION
Use of the internet is almost double in the last 5 years [2] [4] [25]. As per Netcraft survey, approx. 629,939,191 websites are hosted on the server as of January 2013. Approximately, 60% of internet resources are not safe [2]. Growth in attacks is almost exponentially in the last few years [1] [2] [4] [25].

Attackers use different paths for executing their tasks. A wide range of technologies is used for web application development as a result, it is very easy to attack on applications. One attack can do several impacts on business [1] [7]. Application software with security leakages is dejecting our financial, defense, energy, healthcare and other critical infrastructure [1] [8]. Our digital infrastructure gets increasingly complex and interconnected. As a result, it is most critical to secure applications.

1.1 As per OWASP 2013, A New Risk, "Sensitive Data Exposure" has been Identified in 2013 " [1]
Databases are used to store and execute transactions. In order to execute transaction database reveals confidential data at the front end like credit card number, date of birth, social security number, etc. Many applications do not properly protect sensitive data [1] [8] [9]. Attackers may steal or modify such sensitive data to execute the financial fraud, identity theft or other attacks.

Sensitive data need to be protected while storing, retrieving or sharing with the front end. Many techniques are available like encryption, etc. Databases in use have implemented various security policies. Researchers have proposed various solutions to mitigate this Application risk [7] [10] [11] [12].

In this paper, a simple, flexible and dynamic security policy is presented for preventing sensitive data exposure. The model is developed using Oracle and can work on multiple Applications.

This paper is organized as follows: In section 2i problem is defined and Existing Filter solutions are discussed. In Section 3, proposed filter is described, In section 4, testing and results of proposed filter are discussed. The conclusion is provided in section 5 (Refer to figure 1 Paper Map)

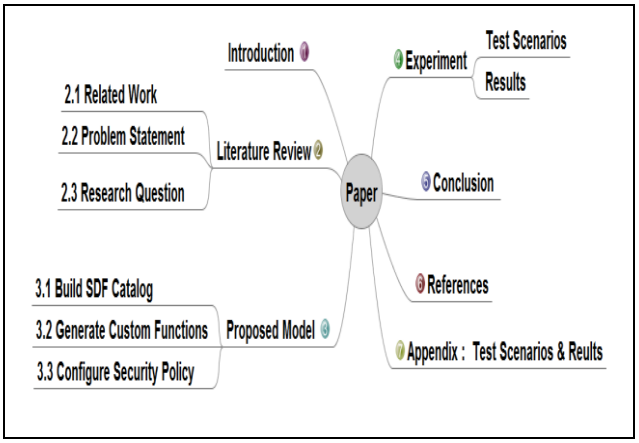

Figure 1: Paper Map

2. LITERATURE SURVEY
2.1 Related Work
As per CIA triangle, Confidentiality, Integrity and availability are 3 key database security goals [5]. The most popular RDBMS has Role Bases Access Control or user level access control mechanism. These security policies provide data access at either table level or row level. When user access data from the database, it retrieves information for all columns. An encryption facility for sensitive data storage is also available.

As per the Open Web Application Security Project (OWASP), the new threat added since 2010 is "A6-Sensitive Data Exposure" [1]. Breaches with more than 10 Million Identities Exposed in 2013, which is 700% more compare to 2012. Top 3 out of top 10 types information breached were real names, birth dates and social security numbers (government numbers) [7].

Attackers most commonly use Structured Query Language (SQL) [1] [9]. An attacker craft SQL statements to manipulate (add, /modify/delete) data or steal information. Key impact as a result of such attacks is loss of data or sensitive data

exposure. Several solutions exist to detect and prevent such attacks caused using SQL [7] [10] [11] [12].

Several solutions have been proposed by Researchers for database security like:

a) Watermark based techniques is implemented to secure database [13]

b) Encryption based techniques is implemented to keep the data in the database secure [14] [15]

c) Cryptography is implemented to keep the data secure in the database by encrypting the data [16] [17].

d) Steganography is implemented to hide critical data and prevent them from unauthorized and direct access[18] [19]

e) Access control based techniques are implemented to hide critical data and prevent them from unauthorized and direct access[20[21] [22] [23] [24]

2.2 Problem Statement

Key Problems with existing Sensitive data exposure risk solutions [13],[14], [15],[16], [17], [18],[19],[20], [21], [22] [23] [24] are summarized as :

Need complex logic to store and retrieve plain text

Works same for all users

Requires more space to store small data

Performance is a major concern

Requires more efforts and skill to deal with techniques

Most places no encryption is done or if done it is key generated are weak [1]

The research problem is defined as:

Research Question	Which concepts, are to be implemented to prevent sensitive data theft even at column level?
Objective	Prevent Sensitive Data loss without performance loss
Variables studied	Database Security, Application Security, Application users and Flow
Resources	Data Base , Database Administrator, Users

Four out top 10 attacks are related to database (SQLI) and caused using SQL[01],[10].[11] so our focus will be on restricting data extraction using SELECT statement of SQL in this paper.

3. PROPOSED MODEL: SENSITIVE DATA FILTER (SDF)

The authors have proposed highly dynamic and flexible automated approach to prevent sensitive data exposure. This solution is deployed at database level on Oracle database(11g).

Figure 2 shows the architecture of proposed solution. Major 3 phases of the proposed solution are Build SDF catalog, creating custom security functions/policies and configuring custom policies using fine grained access control mechanism of oracle database.

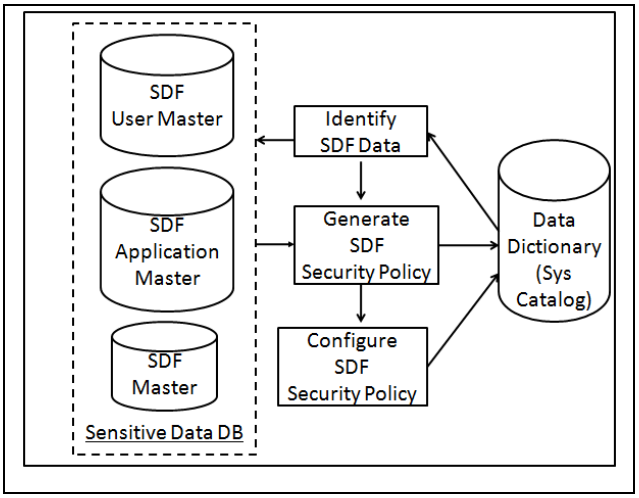

Figure 2. Sensitive Data Filter architecture

3.1 Build SDF Catalog

In this phase, tables to store objects on which security policy need to be are created. There are mainly three objects identified: 1) SDF User Master, 2) SDF Application Master and 3) SDF Policy Master. Which is used to store identified sensitive data, i.e. identifies tables and columns which contain sensitive data, e.g. credit card number, date of birth etc. and stored information into database objects created. To automate above step, JSP based User interface is developed.

Table 1. SDF Data Dictionary :

3.1.1 Sensitive Application Master

App id	User Name
11	PAYROLL
12	SMITH

Explanation: The Sensitive application master will be used to store details of sensitive applications.

3.1.2 Sensitive Object Master

AppId	Table Id	Table Name
11	1	EMLOYEE
11	2	PAY SLIP
11	3	BKP_EMP

Explanation: The Sensitive object master will be used to store application wide sensitive objects

3.1.3 SDF Security Row Level Policy

AppId	ObjId	Grant access to user	Number of rows access granted (max.)	Condition
11	1	SCOTT	10	
11	1	HR	5	
11	1	AR		Deptno=10
11	2	AR	0	

Explanation: SDF Row level security policy will be used to store user wise table wise row level number of rows access granted.

3.1.4 SDF Security Column Level Policy

AppId	ObjId	Columns	SDF_Rule	Hide_flg
11	1	Sal	Deptno=10	
11	3	Comm.		Y

Explanation: The SDF Column level security policy will be used to store column level user access on sensitive columns.

In the next section, how this data have effect when Filter works is explained.

3.2 Generate Custom SDF Security Policy Function

Create Custom stored function for each Security policy defined in above security row and column level tables.

Example: Custom security function Security Row Level Policy will be as below:

Dynamic Custom Security Function

```
CREATE OR REPLACE FUNCTION sdf_row
    ( v_schema IN VARCHAR2,
      v_objname IN VARCHAR2) RETURN
VARCHAR2 AS
  con VARCHAR2 (50);
  rowcnt number(5);
BEGIN
 begin
  select norows, sdf_rule
   into rowcnt,con
  from sdf_appl_mst a,
        sdf_obj_mst b,
        sdf_obj_row_rules c
  where a.appid = b.applid
   and b.applid = c.applid
   and b.objid = c.objid
   and appname = upper(v_schema)
   and objname= upper(v_objname)
   and sdf_user = user;
  exception
   when no_data_found
   then con :='';
  end;
  if rowcnt is not null then
    con :='rownum <= '||rowcnt;
  end if;
   RETURN (con);
END ;
```

Example: Custom security function Security Column Level Policy will be as below:

```
CREATE OR REPLACE FUNCTION hide_sal_data
    v_schema IN VARCHAR2,
     v_objname IN VARCHAR2)
       RETURN VARCHAR2 AS
  con VARCHAR2 (200);
BEGIN
  con := 'deptno=10';
  RETURN (con);
END hide_sal_data;
```

Configure Security policy for sensitive data columns:

Here, defined security policy to restrict display of salary column will be configured. The custom Security function created in the previous step is implemented using dbms_rls package of oracle [26].

Example: SDF Column Level Security Policy

```
BEGIN
 DBMS_RLS.ADD_POLICY (
  object_schema     => 'SCOTT',
  object_name       => 'EMP',
  policy_name       => 'hide_sal2',
  policy_function   => 'hide_sal_data',
  sec_relevant_cols => 'sal',
   sec_relevant_cols_opt => dbms_rls.ALL_ROWS);
END;
```

3.3 How SDF Filter Works?

Whenever any query is executed, the fine grain policy will check whether any security policy is set on objects used in the query,

If yes, it will modify the query and run else run query as it is (refer to Figure 3) .

If no security policy based objects were found in the query, it will execute the query as it is.

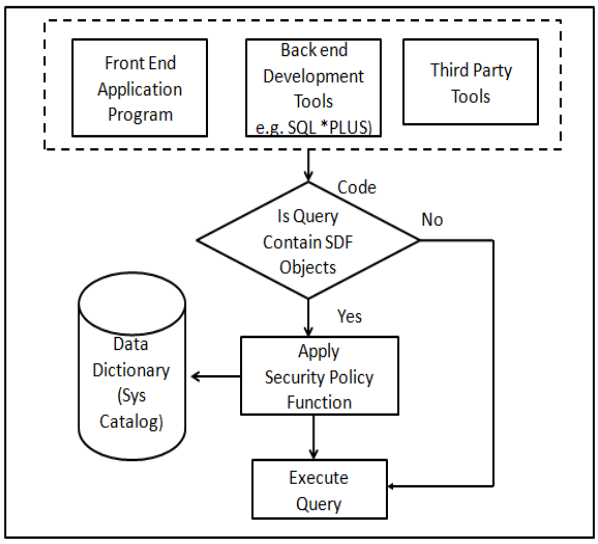

Figure 3. SDF Filter work flow

Example: Custom security function Security Row Level Policy will be as below:

Case 1	
Original Query	Select * from employee;
Filtered query	Select * from employee Where rownum<=10;

Explanation: Case 1: maximum 10 row display access is set on the EMPLOYEE table PAYROLL schema to SCOTT user.

Case 2	
Original Query	Select * from payslip;
Filtered query	Select * from payslip Where rownum<=0;

Explanation: Case 2: No (zero) row access is set on salary slip (payslip).

Case 3	
Original Query	Select * from bkp_emp;
Filtered query	Select * from bkp_emp Where deptno=10;

Explanation: Case 3: Here, SCOTT user can see the salary (sal) of department 10 only

4. EXPERIMENT: TEST AND RESULTS

Filter testing was executed from all three sources (application, Back end and using tools). And the authors were able to restrict sensitive data exposure. We tested security policy on local host web application, JSP based custom reporting forms and SQLl*Plus. The model is able to restrict sensitive data exposure using dynamic security policy. Test scenarios and results are summarized in Appendix A.

Performance observed is as below:

Executing User	Execution Time
SYS	00:00:00.02
SCOTT	00:00:00.02
AR	00:00:00.02
HR	00:00:00.02

5. CONCLUSION

The Sensitive Data exposure filter/security policy is developed successfully. SDF Filter was successfully implemented and thoroughly tested. SDF Filter found to be very effective.

Key Benefits of Model are:

- Easy to implement
- Dynamic as per user/application preferences
- Endless configurations helps in reducing human efforts
- It does not require any specialized infrastructure for implementations
- No extra coding efforts required as it works at the database level

SDF provides a highly configurable environment for application development. User can use it to create even for multiple applications. Such a setup does not require any additional infrastructure and can be used along with existing built-in security techniques (like MAC, DAC and RBAC) in application development to prevent database security.

There is no impact on performance of security policies.

6. REFERENCES

[1] OWASP: https://www.owasp.org /index.php/ Top_10_2013-A1-Injection: accessed 31st May 2014

[2] Internet hosting statistics : http:/ /www.netcraft.com/internet-data-mining: accessed 31st May 2014

[3] Common Weakness Enumeration: http:// cwe.mitre.org/data/definitions/89.html : accessed 3rd June 2013

[4] Internet users : http://www.internetlivestats.com /internet-users: accessed 14th June 2014

[5] Nina Godbole, "Information Systems Security: Security Manaement, Metrics, Frameworks and Best Practices", Wiley India Pvt. Ltd, First Edition 2009

[6] Vulnerability Analysis: http://www.cert.org/vulnerability-analysis/publications/index.cfm:, accessed 14th June 2014

[7] http://www.symantec.com/content/en/us/enterprise/other_resources/b-istr_main_report_v19_21291018 .en-us.pdf

[8] Nina Godbole and Sunit Belapure, "Cyber Security: Understanding Cyber Crimes, Computer Forensic and Legal Perspective", Wiley India Pvt. Ltd, First Edition 2011

[9] Michael Howard, David LeBlanc and John Viega. "24 Deadly Sins of Software Security". "Sin 1: SQL Injection." Page 3-27. McGraw-Hill. 2010

[10] Rahul Johri and Pankaj Sharma " A Survey on Web Application Vulnerabilities(SQLIA and XSS) Exploitation and Security Engine for SQL Injection", IEEE 2012

[11] Diallo Abdoulaye and Al-Sakib Khan Pathan, " A Survey on SQL Injection: Vulnerabilities, attacks AND Prevention Techniques", IEEE 15th International Symposiam on Consumer Electronics, 2011

[12] Ramya Dharam and Sajjan G Shiva, "A framework for development of Runtime Monitors", 2012 International Conference on Computer & Information Science (ICCIS), 2012, IEEE, 978-1-4673-1938-612

[13] Jun Ziang Pinn and A. Fr. Zung:A NEW WATERMARKING TECHNIQUE FOR SECURE DATABASE, International Journal of Computer Engineering & Applications, Vol. I, No. I

[14] Dr. Anwar Pasha Abdul Gafoor Deshmukh, Dr. Riyazuddin Qureshi: Transparent Data Encryption-Solution for Security of Database Contents, International Journal of Advanced Computer Science and Applications, Vol. 2, No.3, March 2011, page 25

[15] Samba Sesay, Zongkai Yang, Jingwen Chen, Du Xu, "A Secure Database Encryption Scheme", Second IEEE Consumer Communications and Networking Conference (CCNC), 3-6 Jan. 2005, pp. 49- 53

[16] Lianzhong Liu and Jingfen Gai, "A New Lightweight Database Encryption Scheme Transparent to Applications", 6th IEEE International Conference on Industrial Informatics, 13-16 July 2008, pp.135-140.

[17] Hasan Kadhem, Toshiyuki Amagasa, Hiroyuki Kitagawa, "A Novel Framework for Database Security based on Mixed Cryptography", Fourth International Conference on Internet and Web Applications and Services, 24-28 May 2009, pp.163-170

[18] M. Naseem, Ibrahim M. Hussain, M. Kamran Khan, Aisha Ajmal, "An Optimum Modified Bit Plane Splicing LSB Algorithm for Secret Data Hiding", International Journal of Computer Applications, Vol. 29, No. 12, 2011. Foundation of Computer Science, New York, USA, pp. 36-43

[19] Wen-Chung Kuo, Dong-Jin Jiang, Yu-Chih Huang, "A Reversible Data Hiding Scheme Based on Block Division", Congress on Image and Signal Processing, Vol. 1, 27-30 May 2008, pp. 365-369

[20] S Rizvi, A Mendelzon, S Sudarshan, Prasan Roy, "Extending query rewriting techniques for fine-grained access control", Proceedings of the ACM SIGMOD international conference on Management of data, 2004, pp.551–562.

[21] Q Wang, T Yu, N Li, J Lobo, E Bertino, "On the Correctness Criteria of Fine Grained Access Control in Relational Databases", Proceedings of the 33rd international conference on Very large data bases, 2007, pp. 555-566.

[22] S Chaudhuri, T Dutta, S. Sudarshan, "Fine Grained Authorization Through Predicated Grants", IEEE 23rd International Conference on Data Engineering, 15-20 April 2007, pp.1174-1183.

[23] Hong Zhu and Kevin Lü, "Fine-Grained Access Control for Database Management Systems", Data Management. Data, Data Everywhere, vol. 4587, 2007, pp. 215-223

[24] Zheng Zhang and Alberto O. Mendelzon, "Authorization Views and Conditional Query Containment", International Conference on Database Theory, vol. 3363, 2005, pp. 259-273

[25] Internet and social network penetration worldwide: http://wearesocial.net/tag/stats/ visited on 25th July 2014.

[26] DBMS_RLS :docs.oracle.com: visited on 25th July 2014

7. APPENDIX A : TEST CASES AND RESULTS

Sr#	Security Level	Executing User	Test Scenario	Test Result	Test Pass / Fail
1	Row Level (Rule 1)	SYS	Select * From employee;	Should return 1000 rows as per security policy	Pass
2		SCOTT		Display only 10 rows	Pass
3	Row Level (Rule 2)	SYS	Select * From employee;	Should return 1000 rows as per security policy	Pass
4		HR		Display 5 rows	Pass
5	Row Level (Rule 3)	SYS	Select * From employee;	Should return 10000 rows as per security policy	Pass
6		AR		Should display dept=10 rows	Pass
7	Row Level (Rule 4)	SYS	Select * From payslip;	Should return 1000 rows as per security policy	Pass
8		AR		No rows will be displayed	Pass
9	Column Level (Rule 1)	SYS	Select * From emp;	Should return 1000 rows as per security policy	Pass
10		SCOTT	Select * From emp;	Display salary only for rows related dept=10	Pass