Theoretical and Applied Informatics
Vol. 28 (2016), no. 1&2, pp. 37–55
DOI: 10.20904/281-2037

Analysis of SQL Injection Detection Techniques

Jai Puneet Singh[1*]

[1]CIISE, Concordia University, Montréal, Québec, Canada

Abstract SQL Injection is one of the vulnerabilities in OWASP's Top Ten List for Web Based Application Exploitation. These type of attacks take place on Dynamic Web applications as they interact with databases for various operations. Current Content Management System like Drupal, Joomla or Wordpress have all information stored in their databases. A single intrusion into these type of websites can lead to overall control of websites by an attacker. Researchers are aware of basic SQL Injection attacks, but there are numerous SQL Injection attacks which are yet to be prevented and detected. Over here, we present the extensive review for the Advanced SQL Injection attack such as Fast Flux SQL Injection, Compounded SQL Injection and Deep Blind SQL Injection. We also analyze the detection and prevention using the classical methods as well as modern approaches. We will be discussing the Comparative Evaluation for prevention of SQL Injection.

Keywords SQL Injection; runtime monitoring; Static Analysis

Received 29 JUN 2016 **Revised** 13 JAN 2017 **Accepted** 02 FEB 2017

This work is published under CC-BY license.

1 Introduction

Web Application is widespread today as they have become the necessity of everyday life. There are thousands of security breaches that take place every day. According to Bagchi [1] 75% of the firm's websites and web applications were vulnerable to the Internet Security Breaches. He had analyzed through Gompertz model the growth of Internet Security breaches and the exposure to an attack. The most common attack on the web is through SQL Injection. The classical SQL Injections were easy to prevent and detect as well as a lot of procedures, methodologies were discussed in order to overcome SQL Injections. The various methodologies used to overcome the attack is by writing secure code in accordance with the extensive research by Howard and his team, which relates to the writing of defensive code with proper validation by usage of encoding and decoding techniques [2]. Still, today writing defensive codes is encouraged but it is not enough

*E-mail: ja_ngh@live.concordia.ca

to protect SQL Injection Attacks. The SQLIA's are widespread attacks on the websites which are followed by XSS (Cross Site Scripting) attack. A study by Gartner Group over 300 Internet Web sites has shown that most of them could be vulnerable to SQLIAs [3].

There are numerous types of SQLIA's and each has various approach for attacks onto the website. The different places where SQL Injection can be mounted is on the web form submission of the website, the second popular place to insert SQL Injection is the URL and it often comes in the domain of Blind SQL Injection, thirdly within the login field of the website sometimes it can be inserted in the password field and lastly within the discussion forums.

The complex formation occurs with the combination of SQL Injection and XSS attacks which lead to retrieval of the Database information. Even the SQL Injection attacks are taking place in the Rich Internet Application by finding the assailability in cross domain policies. Most of the modern websites extensively use Rich Internet Application [4] such as Adobe Flash and Microsoft Silver light, for increased user defined functionality. If the care is not taken during the coding of cross site scripts, it can lead to the vulnerability of XSS and SQL Injection Attacks. These types of attacks were not present a few years ago with the advancement in the field of UI/UX design and various other technological changes such as the introduction to JSON, Jquery which resulted in new vulnerabilities. In order to counter these attacks, we will be extensively discussing the modern SQL Injection attacks and the ways to protect and defend against these type of attacks. The negligence at the initial stage can lead to monetary losses at a later stage.

The rest of the paper is organized as follows: Section 2 describes the Background of the SQL Injection Attacks and the concepts related to it. Section 3 details the example application which will be used throughout the paper for the discussion of the advanced SQL Injection Attacks (SQLIA). Section 4 lists the detection and prevention of SQLI Attacks. Section 5 presents the evaluation of the techniques for the prevention and detection of such attacks. We provide the summary and conclusion in Section 6.

2 Background

SQL Injection Attack occurs when the attacker tries to insert malicious code into the Web Application database which is intended for the retrieval or corruption of data. These attacks are moreover used on E-Commerce websites for the extraction of credit card numbers or it is widely used for bypassing the authentication. Su and Wassermann describe SQL Injection thoroughly and formally with an explanation on Code Injection as well as validation using SQL Check [5]. For website fields without proper input validation, an attacker could obtain direct access to the database of the underlying application [6].

The early protection from the SQL Injection attack was with the input validation in which the user was not allowed to enter the special characters. During the few years, the technology had advanced, and the attackers have taken a turn to use more sophisticated and complex techniques to induce the injection attacks. The authors in [7] explains the background history of the SQL Injection attacks. The list defined by William et al. [7] is limited but gives the clear idea from where and how the SQL Injection attack takes, and different researchers have provided various techniques to protect it. We extend the idea and define the new types of attacks and how they can be protected with using the current techniques. The paper gives a view how the latest attacks

take place with the source code and at last explaining which tool can be used to protect such attack. There are various errors in the techniques developed for the protection of SQL Injection attack. The various concern regarding the latest development are:

- The SQL DOM [8] has been developed to protect from the attack but this programming principles and paradigm has to be learned by the developer. It poses a difficulty in the fast changing world of programming.
- The Michael [9] uses a proxy filter to defend against the attack, but this is not a viable solution as it does not provide completeness and accuracy.
- The penetration testing technique of Black Box testing as proposed by Huang and colleagues [10] which used machine learning for testing but it cannot guarantee the completeness as the other techniques used.

3 SQL Injection Attack Types

There are numerous research papers which present various SQL Injection attacks, but most of them discuss the classical SQL Injection whereas the modern SQL Injection attacks are more dangerous. The modern SQL Injection attack can overcome many previously discussed Detection and Prevention techniques. This section is divided into two subsections which are allocated into Classical SQL Injection and Advanced SQL Injection.

3.1 Classical SQL Injection

In this section, we just give short overview of the classical SQL injection which is as follows.

3.1.1 Piggy Backed Queries

Intent of Attack: Retrieval of Information, Denial of Service. In this type of attack the attacker 'Piggy Back' the query with the original query in the input fields present on the web application. The piggybacked can be defined as 'on or as if on the back of another'. The database receives multiple queries [3]. During the execution, the premier query works as in a normal case the second query adjoined with the first query is used for SQL Injection Attack. It is considered as a menacing attack since it fully exploits the database. Using proper prevention and detection technique this type of attack can be prevented. As an Example:

```
SELECT customer_info
FROM   accounts
WHERE  login_id = ''admin''
AND    pass = '123';DELETE FROM accounts WHERE CustomerName = 'albert';
```

After executing the first query the interpreter sees the ';' semicolon and executes the second query with the first query. The second query is malicious so it will delete the all the data of the customer 'Albert'. Hence, these types of the malicious acts can be protected by firstly determining the correct SQL Query through proper validation or to use different detection techniques. This type of attack can be prevented using Static Analysis, Runtime monitoring is not needed.

3.1.2 Stored Procedure

Intent of Attack: Escaping Authentication, Denial of Service. Stored Procedures are widely used as a subroutine in a relational database management system. They are compiled into a single execution plan and extensively used for performing commonly occurring tasks. It is used in businesses as it provides a single point of control while performing the business rules. IT Professionals think that SQL Stored Procedures are the remedy for the SQL Injection as Stored Procedures are placed on the front of the databases the security features cannot be applied to them. The stored procedures do not use the standard Structured Query Language, it uses its own scripting languages which do not have same vulnerability as SQL but different vulnerability related to the scripting language still exist. As an Example:

```
CREATE PROCEDURE user_info
  @username   varchar2
  @pass       varchar2
  @customerid int
AS
  BEGIN
    EXEC(‘SELECT customer_info fromcustomer_table WHERE
      username=‘ ‘‘+@username’’ ’ and pass = ‘ ‘‘+@pass ’’ ’
    GO
```

This type of procedures are vulnerable to the SQL Injection Attack. Any malicious user can enter the malicious data in the fields of username and password. The simple command entered by the user can destroy the whole database, or it can lead to service disruption. It is always advised not to store the critical information on the stored procedures, as it lacks the most important security features.

3.1.3 Union Query

Intent of Attack: Bypassing Authentication, Data Extraction. This type of attack uses Union Operator `U` while inserting the SQL Query. The two SQL query are joined with the Union Operator. The first statement is a normal query after which the malicious query is appended with union operator. Hence, it is used to bypass the prevention and detection mechanism of the system. The example shows how it to proceed. The example shows that the second query is malicious and text following – is disregarded as it becomes comment for the SQL Parser. Taking this an advantage the attacker attacks the web application or website with this query. As an example of the SQLI Attack with the Union Query:

```
SELECT *
FROM   accounts
WHERE  id = ’212’
UNION
SELECT *
FROM   credit_card
WHERE  USER = ’admin’--’ and pass=’pass’
```

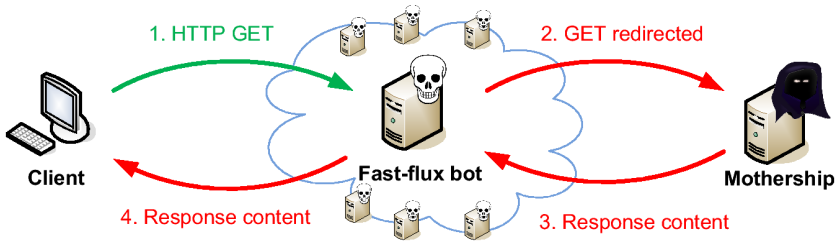

Figure 1 Fast Flux Attack

3.1.4 Alternative Encoding

Intent of Attack: Evading Detection. In this type, the attacker changes the pattern of the SQL Injection so that it goes undetected from the common detection and prevention techniques. In this method, the attacker uses hexadecimal, Unicode, octal and ASCII code representation in the SQL Statement. It goes undetected with the common detection and prevention. As these could not be able to detect the encoded strings and hence, allows these attacks to go undetected.

3.2 Advanced SQL Injection

3.2.1 Blind SQL Injection Attack

Intent of Attack: Data extraction Many Web applications disable to display the SQL error messages. In this attack, the information is inferred by asking true/false questions. If the injection point is completely blind then only way to attack is by using the WAIT FOR DELAY or BENCHMARK command. This type of injection is known as Deep Blind SQL Injection Attack [11].

3.2.2 Fast Flux SQL Injection Attack

Intent of Attack: Data Extraction, phishing. Phishing is a significant security threat to the users of an Internet. The phishing is a social engineering attack in which an attacker fraudulently acquire sensitive information from the user by impersonating as a third party [12]. Traditional phishing host can be detected very easily just by tracking down the public Domain Name Server or the IP address. This trace back technique could lead to the shutdown of the hosting websites. The attackers understood that conducting a loaded attack can have a significant effect on load balancing of server [13]. To counter this action in order to protect its criminal assets, the operator of phishing websites started using Fast Flux technique. Fast Flux is a Domain Name Server technique to hide phishing and malware distribution sites behind an ever-changing network of the compromised host. The fast flux attacking technique can be understood by the diagram on Fig. (1).

The massive SQL Injection which means many queries are used for an attack at the same time using fast flux. It can take place using the Asprox botnet. In Fast Flux mode, the DNS (Domain Name server) simultaneously hosts varied malware infected IP's and the IP's constantly changing. The first fast flux SQL The injection was detected in banner82.com which now has been closed, but it was studied by researchers thoroughly. It was infecting new hosts to be added

1	67.161.224.204	banner82.com	banner82.com
2	74.78.23.64	banner82.com	banner82.com
3	69.140.230.80	banner82.com	banner82.com
4	71.11.244.47	banner82.com	banner82.com
5	99.254.31.140	banner82.com	banner82.com
6	67.167.252.180	banner82.com	banner82.com
7	69.231.247.143	banner82.com	banner82.com
8	68.158.153.62	banner82.com	banner82.com
9	98.211.79.238	banner82.com	banner82.com
10	76.180.52.119	banner82.com	banner82.com

(a)

1	68.154.33.242	banner82.com	banner82.com
2	99.227.84.105	banner82.com	banner82.com
3	98.200.11.115	banner82.com	banner82.com
4	97.80.36.252	banner82.com	banner82.com
5	66.169.212.252	banner82.com	banner82.com
6	97.81.100.62	banner82.com	banner82.com
7	75.38.112.73	banner82.com	banner82.com
8	67.201.212.83	banner82.com	banner82.com
9	172.136.93.52	banner82.com	banner82.com
10	69.247.201.61	banner82.com	banner82.com

(b)

Figure 2 (a) Phase 1 of IP Addresses for a SQL Injection Attack in Fast Flux; (b) Phase 2 of IP Addresses for a SQL Injection Attack in Fast Flux [14]

to the botnet. Banner82.com (now a closed domain) has a tiny iFrame that's attempting to load *dll64.com/cgi-bin/index.cgi?admin* where the NeoSploit malware exploitation kit is serving MDAC ActiveX code execution (CVE-2006-0003) expl [14]. Fig. (2) show that in Fast Flux Attacks the IP address is continuously changing so it is very difficult to trace down the website and to stop from spreading the malware. The fast flux attacks in SQL Injection is difficult to predict and defend. Therefore, a very little research for protection and detection has been done for the Fast Flux SQL injections. The one SQL statement is used to attack the ASP/IIS using Asprox.

3.2.3 Compounded SQL Injection Attack

Compounded SQL Injection Attack is the mixture of the two or more attacks which attack the website and cause more serious effect than the previously discussed SQL Injections. Compounded SQL Injection has come into the place due to rapid development of prevention and detection techniques against various SQL Injections. To overcome, the malicious attackers developed a technique called compounded SQL Injection. Compounded SQL Injection is derived from the mixture of SQL Injection and other Web Application Attacks which can be detailed as follows.

SQL Injection + DDoS Attacks DDoS (Distributed Denial Of Service) is defined as the attack that is used to hang a server, exhaust the resources so that the user is not able to access it. It can be categorized as Web Application DDoS. In SQL we can create extremely complex queries in order to get an output in the desired manner. There are various commands which can be used in SQL Injection in order to pursue with DDoS Attack using advanced SQL commands such as encode, compress, join etc. A very little research on this topic for prevention has been conducted as it is a very complex attack to understand in security viewpoint. In order to pursue with this type of attack, there are the basic steps to be followed which can be done by finding the vulnerability, preparing for the vulnerability and finally, the complex code is used for an attack. The greater the number of columns and rows in the database it will be easier for the SQL DDoS attack. Hence the sample code is used to make SQL DDoS attack on the website is as follow:

```
SELECT tab1
FROM   (SELECT Decode(Encode(CONVERT(Compress(post) USING latin1),
                                    Concat(post, post, post, post)),
                                    Sha1(Concat(post, post, post, post))
        AS tab1
        FROM   table_1)a;
```

If we find using Union SQL Injection that the website is vulnerable to the SQL Injection but we got to know that only 3rd column is vulnerable, so we will try to inject the payload into the website as sample presented.

```
HTTP:exploitable-web.com/link.php?id=1'
     UNION
     SELECT 1,2,
            tab1,
            4
     FROM   (SELECT decode(encode(CONVERT(compress(post) using latin1),
                        des_encrypt(concat(post,post,post,post),8)),
                  des_encrypt(sha1(concat(post,post,post,post)),9))
               AS tab1
             FROM   table_1)a--
```

We can use the sleep command present in SQL to make connections live for long that will help to do the task. Using Sleep we can also do pooling of the connection in ASP.net or many other programming languages where by default maximum 100 or 150 connections are allowed at the time of 30 seconds. If we can make our connection live using Sleep command it won't allow the server to reply other users. Hence, our DDoS attack using the SQL will be achieved.

SQL Injection + DNS Hijacking Ex-filtration of the data using Blind SQL Attack is usually slow. So, the attackers came out with DNS attack which is much faster and less noisy than the blind SQL Attack. DNS are more allowed than any other command to access the database and connect to the arbitrary host. The attacker main goal is to embed the SQL Query in DNS request and to capture it and makes its way onto the internet.

The term DNS Hijacking does not mean web hacking of a DNS (Domain Name Server) but it relates to the modification of the DNS entries, Exploiting the administration of the web of domain registers. When DNS Hijacking has been achieved then the second part comes into an

effect which is SQL Injection attack with the DNS lookup. Conceptually, the attack would be as shown in the figure where the website used is a dummy website.

```
do_dns_lookup(
        (SELECT TOP 1
              password
       FROM   users) + '.inse6140.net' );
```

The SELECT statement is used to obtain the password hash that the attacker is interested and appended a domain name which we have control to the end of it (e.g. inse61400.net) which is done with the help of DNS Hijacking. At last, we perform a DNS lookup (address-based lookup for a dummy hostname). Then we run a packet sniffer on the name server for our domain and wait for the DNS record containing our hash [15]. Below is the another example for the SQL Injection with the DNS Hijacking in real time.

```
someserver.example.com.1234 > ns.inse6140.net.53 a? 0x1234ABCD.inse6140.net
```

The string '`0x1234ABCD`' here represents the password hash we hope to extract using our SELECT statement.

SQL Injection + XSS The manager of IBM Dewey [16] says about SQL Injection + XSS attack 'When you get down to the nuts and bolts of it, this is a cross-site scripting attack. SQL injection was just a vehicle to get there'. In his statement, he means that SQL Injection is the way for setting up an attack, the rest of the work is done by XSS (Cross Site Scripting). These attacks are known as third wave attacks as they are not typically the old way of attacking, but they are the commands from hiding from Network Monitoring devices.

XSS (Cross Site Scripting) can be defined as the client side code injection attack wherein the attacker can inject malicious code into to legitimate website or an application. The script is usually inserted into the input fields of a website. After inserting the scripts are executed as it is and the role of the attacker fulfills. The Fig. (3) shows the normal content of the file, the content of the file after adding the script into the input fields of the website and finally the XSS Attack with the MySQL Injection. The script will try to connect to the database of the website. Hence it's a difficult and complex task. As JavaScript is a client side language whereas accessing the database is usually handled by server side languages. If the connection is successful then the attacker will have access to the database but through client side language. It is mainly used for extraction of the data. The implementation for inserting and modifying the data will become extremely difficult. The further modification for extraction of the data can be done using the code defined below. There are innumerable websites which are vulnerable to the XSS + SQL Injection Attack [17]. These are complex attacks, and their prevention and detection become really difficult as most of the websites usually use JavaScript. The development of JavaScript is in paces such as Node.js and much more. The developers are unaware of the type of vulnerability it carries.

SQL Injection attack using Cross Domain Policies of Rich Internet application Today majority of websites uses Adobe Flash and Microsoft Silver Light for boosting and increasing the user interactivity of the websites. Using this type of applications is vulnerable to the SQL Injection

```
print "<html>"
print "<h1>Most recent comment</h1>"
print database.latestComment
print "</html>"
```

(a)

```
print "<html>"
print "<h1>Most recent comment</h1>"
<iframe src="http://evil.com/xss.html">
print "</html>"
```

(b)

```
print "<html>"
print "<h1>Most recent comment</h1>"
<script>
var connection =new ActiveXObject("ADODB.Connection");
var connectionstring="Data Source=<server>;
Initial Catalog=<catalog>;
User ID=<user>;
Password=<password>;
Provider=SQLOLEDB";
connection.Open(connectionstring);
var rs = new ActiveXObject("ADODB.Recordset");
rs.Open("SELECT * FROM table", connection);
rs.MoveFirst
while(!rs.eof)
{
   document.write(rs.fields(1));
   rs.movenext;
}
rs.close;
connection.close;
</script>
print "</html>"
```

(c)

Figure 3 (a) Normal code within the website for displaying the comments [18]; (b) Addition of the iframe command which is used for phishing attack (XSS Attack)l (c) SQL Injection attack using the XSS

and another type if care is not taken during programming a code. They use cross-domain policies to run the website. Misinterpretation and not a proper use of the cross-domain policies give rise to the vulnerability in Rich Internet Application. Cross-Domain policies is an XML file which gives permission to web client to handle data in multiple domains [19]. Cross-domain policies define the list of RIA hosting domains that are allowed to retrieve content from the content provider's

domain. It was first defined by the Internet Storm Center in the year of 2008, when the legitimate sites were being attacked by Asprox Injection String. It first determines the browser being used which is either Firefox or Internet Explorer. Then Java Script file is run to determine the version of the flash player being used which is usually done for inducing the SQL Injection attack.

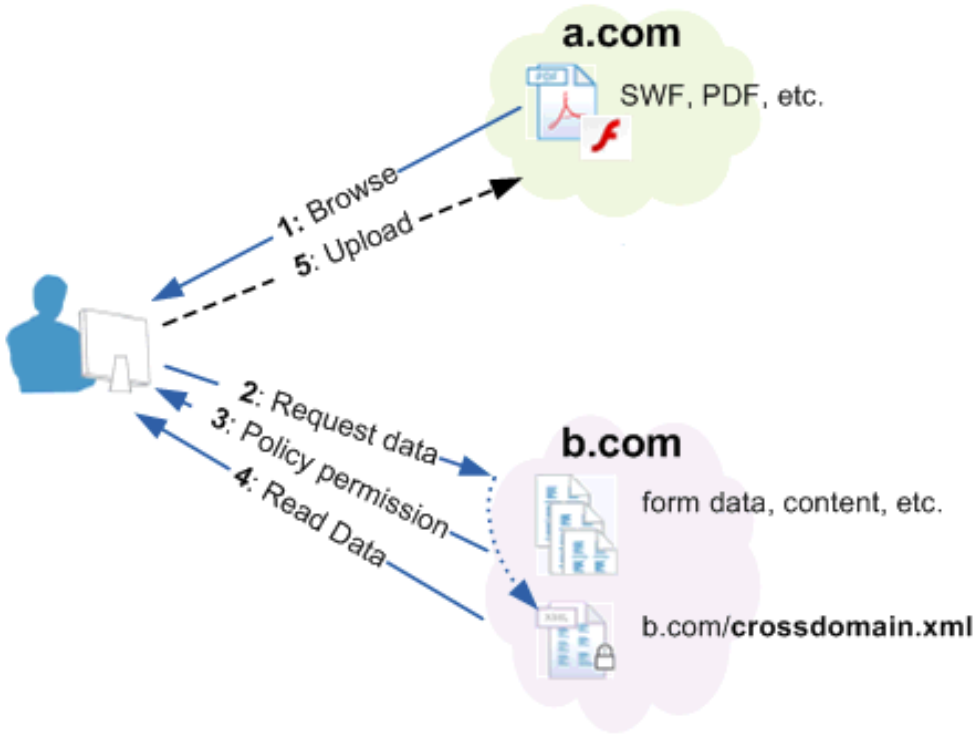

Figure 4 Explanation of Cross-Domain Policy

Georgios [4] in his paper describes how weak statements are written when programming for Rich Internet Application that is vulnerable to the attack. The first example in Fig. (5) shows the right code for the cross-domain policy. If this code is written then the website is not vulnerable to SQL Injection for the Rich Internet Applications. The other example shows if the coding style of the programmer changes which is shown in the figure then a code is vulnerable to the SQL Injection Attack as well as it is vulnerable to other types of attack. There are key points in order to prevent SQL Injection which we will discuss in Prevention and Detection section.

SQL Injection + Insufficient Authentication This type of Compounded Attack is associated with the Insufficient Authentication where the user or the site administrator is a novice. The security parameters have not been initialized where the application fails to identify the location of user, service or application. It can also refer to the website which allows the attacker to access the sensitive content without verifying the identity of the user. This advantage is taken by the attacker to induce the SQL Injection Attack. Hence, this type of attack is very easy in comparison with the other types of attacks. The first step is to find whether the website has insufficient

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM
http://www.concordia.ca/files/docs/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
<allow-access-from domain="sub1.domain1.com"/>
<allow-access-from domain="domain3.com"/>
</cross-domain-policy>
```

(a)

```
<allow-access-from domain="*.sub1.domainA.com"/>
<allow-access-from domain="*.domainC.com"/>
<allow-access-from domain="*"/>
```

(b)

Figure 5 (a) A Valid Code for Cross-Domain Policy; (b) A Week Code for the Cross-Domain Policy

authentication. If this is the case then the SQL Injection attack can take place.

4 Detection and Prevention

Detection and Prevention is a difficult task if there is proper understanding about the SQL Injection Attacks types then it is easier to prevent the attack. To prevent modern SQL Injection Attack, it is always advised to use the prepared statements [20] as it is fixed and cannot be modified by the user of a website or web application. The techniques like `magic_quotes()` and `add_slashes()` cannot protect the Web Application or Web Site from the SQL Injection Attack. In this section, we will discuss various techniques for the detection and prevention of modern SQL Injection.

4.1 Blind SQL Injection Detection and Prevention

There are plenty of research papers for the Blind SQL Injection where they describe various Detection and Prevention Techniques. As Blind SQL Injection are difficult to detect and prevent but researchers were aware of Blind SQL Injection for many years. The most popular technique used is AMNESIA [7] which stands for Analysis and Monitoring for Neutralizing SQL-injection attacks. This tool is only applicable to protect Java Based Applications, and it uses runtime monitoring mechanism. Komiya et al. [21] came out with the better method for preventing SQL Injection. They encouraged to use Machine Learning Algorithms in order to improve the prevention and Detection of Blind SQL Injection. They obtained the results and verified that prevention and detection were better than SQLCheck [5] and AMNESIA [7].

4.2 Fast Flux SQLI Detection and Prevention

The major attacks where client-side security fails and an emerging phenomenon which is not widely known. Fast Flux SQL Injection attacks have been faced at Indiana University, US and even the security of FBI are concerned about this kind of attack. The best way to protect from Fast Flux attack is to make the servers safe [22]. The Fast Flux can be protected by using the technique by which URL's point to the Javascript delivery hits can be blacklisted if they are identified in a quick fashion. Alper et al. [23] discusses in their paper regarding the Fast Flux Monitor (FFM) that can detect as well as can classify a Fast Flux Service Networks in the order of minutes for using both active and passive DNS monitoring, which complements long-term surveillance of FFSNs. After the Fast-Flux Networks has been classified we can use our SQL Injection Techniques in order to stop SQL Injection which can be stopped by Monitoring. The secure coding techniques should be taken place as a suggestive measure. As attackers are becoming smarter and finding the ways to crack into the system, even the researchers came out with the new techniques to countermeasure the un-detection by the Fast Flux Monitor. Holz et al. [24] came out with a research to detect the Fast Flux networks and SQL Injection attacks by using the Expert Systems. The further improvement was done by Stalsman and Irwin [25] by developing a more suitable method for the detection of the Fast Flux Networks and SQL Injection Attacks. They added that Machine learning methods that can be used for detection. Among many machine learning techniques, they have put emphasis on C5.0 Classifier and Naive Bayesian Classifier for the detection of Fast Flux and SQL Injection Attack. Prevention of the Fast Flux is a really complicated task and many researchers are finding the right techniques to counter the Fast Flux SQL Injection Attacks.

4.3 SQLI XSS Detection and Prevention

Adam et al. [26] discuss in their paper the automatic creation of the SQL Injection and XSS in order to bypass and enter into the database in order to find the vulnerability. They discuss the Ardilla Tool which is primarily an attacking tool in which the user chooses to attack (SQLI, first order XSS or second-order XSS). The tool is used for the detection of the SQLI + XSS attack. It has two modes to check the validity of the attack i.e. strict mode and lenient mode whereas SQLI has only one mode. Ardilla Tool uses Taint Based approaches and static analysis techniques, in this if the preconditions are not met, it will suggest filters and other sanitization methods in order to fulfil the precondition which is the requirement for the detection of vulnerability. The other tools are not as efficient as ARDILLA. Therefore, In order to protect our system firstly, XSS has to be detected and prevented. Secondly, the SQLI detection and prevention methods have to be applied in order to achieve the task.

By using the Cross Scripting Attacks the attacker can attack many different parts of the Web Application. The common being the stealing of cookies which can further lead to vulnerability, loss of critical information and SQL Injection. Stealing of cookies can be prevented by using Dynamic Cookies Rewriting techniques which have been discussed by Rattipong et al. [27]. In his paper, he discusses the creation of the random data and changing the name when storing in the cookies table. As discussed in our above section about the XSS Attack with the SQL Injection which is mostly done with the help of Java Script. In order to prevent these type of

attacks Zhang et al. [28] came out with the Execution flow mechanism in order to protect from JavaScript based XSS attack which serves the purpose of protecting against SQL Injection in XSS. In this prevention technique, they have used the finite state automata to analyze client side JavaScript and it prevents any malicious script to enter or retrieve the data from the database. As it uses the machine learning algorithm which improves with its experience and highly depends on the data sets but it does not guarantee full protection, and it has a significant performance overhead. According to Vogt et al. [29] promises that Dynamic Data Tainting is a technique which is used for the detection and prevention of the XSS Attack, and then SQL Injection is automatically protected but Nikiforakis et al. [30] has counter-reaction as they say there are many hidden channels which remain undetected and hence cannot be prevented from attack by using the Dynamic Data Tainting.

The other tool very popular tool used to mitigate the XSS attack is the Noxes tool [31]. The developers of this tool were inspired by the Windows Firewall. It has certainly helped in protection against XSS attacks + SQL Injection attacks but it fails to prevent the attack completely as discussed by Nikiforakis et al. [30] as they consider that attacker can use HTML Tags instead of Script Tags for an Attack. It takes care about HTTP request and prevents the modification done on the HTTP header and has the functionality to set cookies.

4.4 SQLI DNS Detection and Prevention

In SQL + DNS Prevention and Detection, the rules of dividing will apply. In this approach, DNS Hijacking is detected, and afterwards, SQL Injection prevention and detection takes place. DNS Hijacking can be prevented by not downloading the free utilities from the websites as they mostly contain the vulnerabilities. Diter gollman [32] describe DNS rebinding which tries to capture the router settings of the client or user. In order to prevent the DNS Hijacking the Nikiforakis et al. [30] came out with the session shield that is lightweight client side protection mechanism. Stampar [20] in his paper discusses the usage of SQLMap in the protection of the SQLI + DNS Attack. The SQLMap has the feature of the DNS Ex-filtration and there are many command lines specially designed for DNS prevention and detection. It is compatible with most of the SQL Database versions.

4.5 SQLI Cross-Domain Policies Detection and Prevention

These types of attacks according to the Kontaxis et al. [4] can be protected by using proper policy implementation and reducing the usage of Any Subdomain Weakness and domain Weakness. The method developed by Steven and his colleagues is FlashOver [33] which is used to detect and prevent the XSS attacks in the Rich Internet Applications. As we know that if it is XSS Vulnerable then it would be SQLI vulnerable as SQL Injections can take place with the help of XSS. In their method, they have used the static as well as dynamic code analysis in order to achieve the protection of the SQLI cross-domain Policies. In this method using the Static Analysis, they retrieve the Potentially exploitable variables (PEV) which are later used as an attack vector in FLASHOVER. The method DEMACRO [34] was proposed by Sebastian and his colleagues at SAP Research Center, Germany. Their system for the prevention does not need any training or machine learning methodology. DEMACRO detects the malicious cross-domain

No.	Technique	Blind SQL Injection	FF_SQLI	SQLI_XSS	SQLI_DNS	SQLI_CDP	SQLI_DDoS	SQLI_In_Authen
1.	Crypto Graphical Hash Functions	p	×	×	×	×	×	+
2.	Dynamic Cookies Rewriting	×	×	+	p	×	×	p
3.	Execution Flow Mechanism	×	×	*	×	×	×	×
4.	Static Code Analysis	○	×	○	×	*	×	×
5.	Dynamic Data Tainting	×	×	*	×	×	×	×
6.	Run Time Monitoring	p	+	*	×	×	×	×
7.	Machine Learning	×	○	*	×	×	○	×

Table 1 The various techniques for Detection and Prevention of an Attack

requests and tries to un-authenticate them. They did extensive research and came out with the prevention policy for the Cross-Domains.

4.6 SQLI DDoS Detection and Prevention

DDoS Attacks are well understood by the security professionals, but even though it's well-discussed topic, there are some loopholes which attacker uses to attack the system. The DDoS and SQL Injection Detection was well discussed by Lee and his colleagues who gave the idea of detecting the DDoS Attack using the cluster analysis [35]. The cluster analysis methodology helps to detect DDoS Attacks and can easily identify the type of attack on the system. Yu shui [36] came forth with the discussion of the survey of various detection techniques of the DDoS Attacks. After the detection phase, mitigation comes which is comparatively much easier in the case of the SQLI DDoS attacks. Hence, the research for the DDoS SQLI is widespread. The thing needed at the present time is to utilize the techniques in a proper manner so that we can secure our web servers, web applications and websites from this type of attacks.

4.7 SQLI Insufficient Authentication

In order to protect from the SQLI Insufficient Authentication attack without getting into the trouble of Authentication. The administrator can use the technique of Crypto-graphical Hash functions as used by Singh et al. [37] for protection from SQLI + Insufficient Authentication. In this method, two extra attributes are added which are hash functions for the username and password field. The hash functions are automatically generated using Hash Algorithms. Now, when the client enters the username and password, then hash function is generated and is transferred to the server side for verification. Everything which takes place over here is in encrypted form. If the username and password are same as stored in the database which is matched with its hash functions. Hence, there is a negligible chance for intrusion into the database.

5 Evaluation

In this section, we will evaluate the techniques presented in the earlier section. We have made two tables depicting in this section. In Table 1. we have shown which technique is used to detect

No.	Tools	Blind SQL Injection	FF_SQLI	SQLI_XSS	SQLI_DNS	SQLI_CDP	SQLI_DDoS	SQLI_In_Authen
1.	Ardilla Tool	×	×	○	×	○	×	×
2.	Noxes	×	×	p	×	+	×	×
3.	Session Shield	×	×	∗	p	×	×	p
4.	AMNESIA	∗	×	p	×	×	×	×
5.	SQLMap	○	×	○	p	×	○	○
6.	Fast Flux Monitor	×	○	×	○	×	×	×

Table 2 The Evaluation of various tools for Detection and Prevention of an Attack

and prevent from the modern SQL Injection Attacks. In table 2. we discuss the various detection and prevention tools used.

In this section, we will use an asterisk '∗' for showing that it is used for both Detection and Prevention in respect with the modern SQL Injections on the table. The circle '○' is used for showing that it is only used for the Detection mechanism. The plus '+' is used for depicting only prevention corresponding to the modern SQL Injection. The symbol times '×' is used to depict that techniques or tools do not correspond with the modern SQL Injection (in terms of Prevention and Detection). The symbol 'p' depicts the incompleteness which means after applying the specific method the other method has to be applied in order to achieve complete detection and prevention.

5.1 Detection and Prevention Techniques

In Tab. 1, we have taken various techniques which can be used in order to detect and prevent the attacks. These techniques will help the researchers and security professionals to take proper action or use the specified techniques to solve the crises arisen inside the organization due to an attack. The techniques described here can be used to develop a system with the optional functionality in order to protect the system from any kind of these modern attacks that corresponds to the Compounded SQL Injection and Fast Flux SQL Injection attacks. According to the survey performed, we observed that the Static Code Analysis and Machine Learning are the best among the others but other techniques have various other advantages.

5.2 Basis of Detection and Prevention Tools

In Tab. 2, we have discussed various tools for the detection and prevention of the modern attacks. These tools are ready-made and some are open source which can be downloaded from the internet. Most of these tools were developed for the research purposes, but due to its significant advantages they are being used in the commercial sectors. The tools are discussed giving the broad overview that which of these tools can be used for the particular type of attacks. According to our survey performed we observe that the **Noxeus** and **SQLMap** are latest and have better prevention and detection mechanism.

6 Conclusion

The challenge we have faced writing this survey paper is very little scientific research being done in the field of Fast Flux SQL Injection and Compounded SQL Injection. As it was very hard to determine proper tools for prevention and detection. These topics are difficult to understand, and tools should be quite sophisticated in order to find the deviation from the normal SQL statements. We discussed the modern SQL Injection attack which is less known to the general world as well as many researchers. They are a very typical attack which is done on the web applications and websites. They take a considerable amount of time to understand as they are quite complex when compared with the classical SQL Injection Attacks. We have discussed the prevention and detection techniques of these attacks and numerous techniques discussed to prevent these type of attacks. The prevention and detection techniques discussed are limited due to very few research papers done on these types attacks. These attacks can overcome the previous detection and prevention techniques. Hence, sometimes proper coding of Web Application holds very little value as it can overcome easily. The developer should have the good knowledge of these type of attacks. Otherwise, the attackers can destroy the web application and whose implication can affect the businesses of an organization.

Lastly, we have come up with the Evaluation of various detection and prevention techniques in which we compared it and came out with the general characteristics of the tools used. The future research or evaluation can be done on finding and researching on the optimized algorithm to protect from Fast Flux SQL Injection attack and the compounded SQL Injection attack.

Acknowledgemnts I want to thank Mitacs Inc., Canada for their generous support in the research. Without Mitacs, I would not have been able to complete my research.

References

[1] K. Bagchi and G. Udo. An analysis of the growth of computer and Internet security breaches. *Communications of the Association for Information Systems*, 12(1):46, 2003.

[2] M. Howard and D. LeBlanc. *Writing secure code*. Pearson Education, 2003.

[3] W. G. Halfond, J. Viegas, and A. Orso. A classification of SQL-injection attacks and countermeasures. In *Proceedings of the IEEE International Symposium on Secure Software Engineering*, volume 1, pages 13–15. IEEE, 2006.

[4] G. Kontaxis, D. Antoniades, I. Polakis, and E. P. Markatos. An empirical study on the security of cross-domain policies in rich internet applications. In *Proceedings of the Fourth European Workshop on System Security*, page 7. ACM, 2011. DOI: 10.1145/1972551.1972558.

[5] Z. Su and G. Wassermann. The essence of command injection attacks in web applications. *ACM SIGPLAN Notices*, 41(1):372–382, 2006. DOI: 10.1145/1111320.1111070.

[6] K. Wei, M. Muthuprasanna, and S. Kothari. Preventing SQL injection attacks in stored procedures. In *Australian Software Engineering Conference (ASWEC'06)*. Institute of Electrical and Electronics Engineers (IEEE), 2006. DOI: 10.1109/aswec.2006.40.

[7] W. G. J. Halfond and A. Orso. AMNESIA: analysis and monitoring for NEutralizing SQL-injection attacks. In *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering - ASE '05*. Association for Computing Machinery (ACM), 2005. DOI: 10.1145/1101908.1101935.

[8] R. A. McClure and I. H. Krüger. SQL DOM: Compile time checking of dynamic SQL statements. In *Proceedings of the 27th International Conference on Software Engineering*, ICSE '05, pages 88–96, New York, NY, USA, 2005. ACM. DOI: 10.1145/1062455.1062487.

[9] M. J. Beranek. HTTP caching proxy to filter and control display of data in a web browser, 2005. US Patent 6,886,013.

[10] Y.-W. Huang, S.-K. Huang, T.-P. Lin, and C.-H. Tsai. Web application security assessment by fault injection and behavior monitoring. In *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, pages 148–159, New York, NY, USA, 2003. ACM. DOI: 10.1145/775152.775174.

[11] F. Mavituna. Deep blind SQL injection. *White Paper*, 2008.

[12] T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer. Social phishing. *Commun. ACM*, 50(10):94–100, 2007. DOI: 10.1145/1290958.1290968.

[13] L. Wichman. Mass SQL injection for malware distribution. Technical report, SANS Institute, 2010.

[14] D. Danchev. http://www.zdnet.com/article/fast-fluxing-sql-injection-attacks-executed-from-the-asprox-botnet, 2008.

[15] PenTestMonkey. http://pentestmonkey.net/blog/mssql-dns, 2016.

[16] K. J. Higgins. http://www.darkreading.com/third-wave-of-web-attacks-not-the-last/d/d-id/1129488, 2008.

[17] P. Kaur and K. P. Kour. SQL injection: Study and augmentation. In *2015 International Conference on Signal Processing, Computing and Control (ISPCC)*, pages 102–107, 2015. DOI: 10.1109/ISPCC.2015.7375006.

[18] acunetix Vulnerability Scanner Webpage. http://www.acunetix.com/websitesecurity/cross-site-scripting/, 2015.

[19] Adobe Developer Connection Inc. main page. http://www.adobe.com/devnet/articles. Technical report, Adobe, 2010.

[20] M. Stampar. Data retrieval over DNS in SQL injection attacks. *arXiv preprint arXiv:1303.3047*, 2013.

[21] R. Komiya, I. Paik, and M. Hisada. Classification of malicious web code by machine learning. In *2011 3rd International Conference on Awareness Science and Technology (iCAST)*, pages 406–411, 2011. DOI: 10.1109/ICAwST.2011.6163109.

[22] Y. Shin, S. Myers, and M. Gupta. A case study on Asprox infection dynamics. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 1–20. Springer Nature, 2009. DOI: 10.1007/978-3-642-02918-9_1.

[23] A. Caglayan, M. Toothaker, D. Drapeau, D. Burke, and G. Eaton. Real-time detection of fast flux service networks. In *2009 Cybersecurity Applications & Technology Conference for Homeland Security*. Institute of Electrical and Electronics Engineers (IEEE), 2009. DOI: 10.1109/catch.2009.44.

[24] T. Holz, Ch. Gorecki, K. Rieck, and F. C. Freiling. Measuring and detecting fast-flux service networks. In *NDSS*, 2008.

[25] E. Stalmans and B. Irwin. A framework for DNS based detection and mitigation of malware infections on a network. In *2011 Information Security for South Africa*. Institute of Electrical and Electronics Engineers (IEEE), 2011. DOI: 10.1109/issa.2011.6027531.

[26] A. Kieyzun, P. J. Guo, K. Jayaraman, and M. D. Ernst. Automatic creation of SQL injection and cross-site scripting attacks. In *2009 IEEE 31st International Conference on Software Engineering*. Institute of Electrical and Electronics Engineers (IEEE), 2009. DOI: 10.1109/icse.2009.5070521.

[27] R. Putthacharoen and P. Bunyatnoparat. Protecting cookies from cross site script attacks using dynamic cookies rewriting technique. In *Advanced Communication Technology (ICACT), 2011 13th International Conference on*, pages 1090–1094. IEEE, 2011.

[28] Q. Zhang, H. Chen, and J. Sun. An execution-flow based method for detecting cross-site scripting attacks. In *Software Engineering and Data Mining (SEDM), 2010 2nd International Conference on*, pages 160–165. IEEE, 2010.

[29] P. Vogt, F Nentwich, N. Jovanovic, E. Kirda, Ch. Kruegel, and G. Vigna. Cross site scripting prevention with dynamic data tainting and static analysis. In *NDSS*, volume 2007, page 12, 2007.

[30] N. Nikiforakis, W. Meert, Y. Younan, M. Johns, and W. Joosen. SessionShield: Lightweight protection against session hijacking. In *Lecture Notes in Computer Science*, pages 87–100. Springer Nature, 2011. DOI: 10.1007/978-3-642-19125-1_7.

[31] E. Kirda, Ch. Kruegel, G. Vigna, and N. Jovanovic. Noxes: a client-side solution for mitigating cross-site scripting attacks. In *Proceedings of the 2006 ACM symposium on Applied computing - SAC '06*. Association for Computing Machinery (ACM), 2006. DOI: 10.1145/1141277.1141357.

[32] D. Gollmann. Securing Web applications. *Information Security Technical Report*, 13(1):1–9, 2008. DOI: 10.1016/j.istr.2008.02.002.

[33] S. Van Acker, N. Nikiforakis, L. Desmet, W. Joosen, and F. Piessens. FlashOver: Automated discovery of cross-site scripting vulnerabilities in rich internet applications. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security - ASIACCS '12*, pages 12–13. Association for Computing Machinery (ACM), 2012. DOI: 10.1145/2414456.2414462.

[34] S. Lekies, N. Nikiforakis, W. Tighzert, F. Piessens, and M. Johns. DEMACRO: Defense against malicious cross-domain requests. In *Research in Attacks, Intrusions, and Defenses*, pages 254–273. Springer Nature, 2012. DOI: 10.1007/978-3-642-33338-5_13.

[35] K. Lee, J. Kim, K. H. Kwon, Y. Han, and S. Kim. DDoS attack detection method using cluster analysis. *Expert Systems with Applications*, 34(3):1659–1665, 2008. DOI: 10.1016/j.eswa.2007.01.040.

[36] S. Yu. DDoS attack detection. In *Distributed Denial of Service Attack and Defense*, pages 31–53. Springer Nature, 2013. DOI: 10.1007/978-1-4614-9491-1_3.

[37] S. P. Singh, U. NathTripathi, and M. Mishra. Detection and prevention of SQL injection attack using hashing technique. *International Journal of Modern Communication Technologies & Research*, 2, 2014.