CHAPTER 2

LITERATURE SURVEY

2.1 INTRODUCTION

Web application security is a major concern in today’s era for various organizations and banking sectors. Most of the organizations and banking sectors who use internet to provide web based services that protect their sensitive data using firewalls and few access control mechanisms. However, still the organizations data are revealed by internet hackers by some means of purposefully designed SQL queries. Therefore, the current security mechanisms are not sufficient to provide effective security to the web databases. In such a scenario, it is necessary to provide additional protection mechanisms for securing the critical information that are retrieved by SQL queries designed carefully by hackers.

This literature review aims at analysing the latest developments in the area of web security mechanisms for preventing SQL injection and XSS attacks. This analysis has been used for formulating new methods to prevent various types of attacks in web applications.

2.2 WEB APPLICATION SECURITY

Several works have been proposed and implemented in the past by various researchers (Cook et al 2005, James et al 2001, Vidar Kongsli 2006, Viktoria Felmetser et al 2010) for the protection of web applications from SQL injection attacks and cross site scripting attacks. Some of the most important attacks and their limitations are discussed in this thesis and they are
compared with the techniques proposed in this thesis. The analysis is divided into two categories namely the static and dynamic analysis methods.

### 2.2.1 Works on Static Analysis

There are many works that have been proposed in the literature for securing web applications from SQL injection and cross site scripting attacks. In most works, only static analysis methods were proposed and implemented by many researchers. Among the important static analysis methods the static analysis method proposed by Nenad Jovanovic et al (2006, 2010) addressed the problem of vulnerable web applications using static source code analysis. They used a flow sensitive analysis and context sensitive data flow analysis to discover vulnerable points in a program. Moreover, they have enhanced the quality and quantity of the generated vulnerability reports by employing an iterative two phase algorithm for fast and precise resolution of file inclusions. They targeted at general class of taint style vulnerabilities which were applied to detect vulnerability types such as SQL injection, cross site scripting and command injection.

Yao Wen Huang et al (2004) described a sound and holistic approach to ensure web application security that views web application vulnerabilities as a secure information flow problem. They also created a lattice based static analysis algorithm derived from type systems as well as type state and addressed its soundness. Gary Wassermann and Zhendong (2008) discussed the static detection methods proposed by them for finding cross site scripting vulnerabilities. They provided a static analysis method for finding XSS vulnerabilities that directly address weak or absent input validation. Their approach combines the techniques for tainted information flow with string analysis. Moreover, they also provide an extensive evaluation that finds both known and unknown vulnerabilities in real world web applications.
Monga et al (2009) presented a hybrid analysis framework for detecting web applications vulnerabilities that blends together the strengths of static and dynamic approaches for the detection of vulnerabilities in web applications. In their work, a static analysis performed just once is used to reduce the run time overhead of the dynamic monitoring phase. Their system is capable of statically analyzing script language byte codes by searching for dangerous code statements. Moreover, it monitors only the identified statements during the dynamic analysis phase.

Sven Lachmund (2010) proposed auto generating access control policies for web applications using static analysis methods. Using these policies and user input recognition, it is possible to distinguish resource accesses initiated by the application from those initiated by the user. Moreover, it generates an application policy which only contains access rights that are not derived from user interaction. Their policy model also satisfies the principle of least privileges. In their model, the user initiated accesses are handled separately at runtime which are determined and filtered in that system. Andrea Avancini and Mariano Ceccato (2010) proposed a method towards security testing based on taint analysis that performs preliminary investigation on the integration of static analysis with genetic algorithms. Their approach suggests techniques to extract candidate false positives reported by static analysis and provides input vectors that expose actual vulnerabilities.

Another static tainting method proposed by Adam Chlipala (2010) checks the dynamically varying security policies statistically in database backed applications called UrFlow. This system performs static checking of security policies for database backed web applications based on a combination of access control and information flow policies. But their work
is significant only for static checking program analysis and not for dynamic flow analysis. Most of the works present in the literature perform well only for static web applications but does not secure the dynamic web applications from SQL injection and cross site scripting attacks. Therefore, it is necessary to perform dynamic analysis in order to enhance the web security.

2.2.2 Works on Dynamic Analysis

In the past, many works pertaining to dynamic analysis have been proposed and implemented by various researchers (Wei Xu et al 2005, William G.J. Halfond 2008) to secure web applications from SQL injection and cross site scripting attacks. Most of the important works that focus on dynamic analysis are discussed in this thesis.

Yao Wen Huang et al (2003) proposed techniques for web application security assessment mechanisms in order to identify poor coding practices that render web applications vulnerable to attacks such as SQL injection and cross site scripting attacks. They used software testing techniques (including dynamic analysis, black box testing, fault injection, and behavior monitoring), for testing the security in web applications.

The dynamic analysis method developed by Doudalis et al (2007) was an effective memory protection technique that uses dynamic tainting method to detect illegal memory accesses. According to this method, it is possible to check whether the memory allocated at runtime is tainted both in the memory and the corresponding pointer using the same taint mark. Moreover, this technique has been implemented at the binary level and hence it helps to handle applications using third party libraries whose source code is unavailable.
The system called Dytan was developed by Clause J. Li et al (2007) that uses generic dynamic taint analysis framework to address the limitations of existing work and to foster experimentation with dynamic tainting technique. This work provides a general framework for dynamic tainting that is highly flexible as well as customizable and allows for performing both data flow and control flow based tainting conservatively. Balzarotti et al (2008) proposed a novel approach for the analysis of the sanitization process. They developed this approach by combining both static and dynamic analysis techniques to identify faulty sanitization procedures that can be bypassed by an attacker.

Raymond Wu and Masayuki Hisada (2010) explained their work by using macro and micro views. The macro view oversees syntax structure and identification, while micro view envisions metadata messaging and parser automation. The coherence of macro and micro views forms the web security framework for tracking and validation of the code. Their work can be used to perform the security services through fraud detection. It also demonstrates metadata messaging for tracking and code generation techniques for validation which bridges the gap between static and dynamic analysis. Viktoria Felmetsger (2010) proposed a technique towards automated detection of application logic vulnerabilities in web applications. They used dynamic analysis to observe the normal operation of a web application and hence to infer a set of behavioral specifications. Their system is capable of finding unknown logic vulnerabilities.

A new approach called dynamic multi process information flow tracking for web applications was proposed by Susanta Nanda et al (2007). This model presents the design, implementation and evaluation of a dynamic checking compiler called WASC, which automatically adds checks into web applications used in three-tier internet services to protect them from the most
common two types of web application attacks. Moreover, Inyong Lee et al (2010) proposed a very simple and effective detection method for SQL injection attacks. Their method removes the value of an SQL query attribute of web pages when parameters are submitted and they have compared it with a predetermined one. Their method uses combined static and dynamic analysis.

Venkatakrishnan et al (2010) proposed a framework for the prevention of SQL injection and XSS attacks on web applications. Their framework incorporates techniques based on static and dynamic analysis, symbolic evaluation and execution monitoring to retrofit existing web applications to be resilient to these attacks. All the dynamic tainting methods reported in the literature focus only on securing the web applications from SQL injection and cross site attacks by dynamic tainting analysis. However, most of these techniques are capable of only known attack patterns and hence the attackers are able to carry out new attacks based on new attack patterns. Hence, it is necessary to propose and implement new and effective detection techniques which are capable of detecting both known and novel attacks. For this purpose, a new detection and prevention model that provides both static and dynamic analysis has been proposed and implemented in this research work.

2.3 RELATED WORKS ON XSS ATTACKS

There are several works that have been proposed and implemented in the past by various researchers (Mike Ter Louw et al 2008a, 2010, Prithvi Bisht et al 2010c) to secure web application from cross site scripting attacks. Some of the most important works are cited below.

Benjamin Livshits and Ulfar Erlingsson (2007) proposed a web application construction framework to protect the web data against code injection attacks. Their work suits well for detecting script injection
vulnerabilities which have been built on top of AJAX development framework but limited to other kind of applications. Garcia Alfaro and Navarro Arribas (2007) presented a survey on detection techniques to prevent cross site scripting attacks on current web applications. In their work, they focused on the specific problem of preventing XSS attacks against web applications which are based on the use of X.509 certificates and eXtensible Access Control Markup Languages (XACML) for the expression of authorization policies. Their method proves to be efficient for some specific applications since it can specifically express its security requirements from the server side as well as on the requirements of client side.

William Halfond et al (2006, 2008) developed the Web Application Security Provider (WASP) that protects web applications using positive tainting and syntax aware evaluation. Moreover, WASP protects several web applications and subjected them to a large and varied set of attacks and legitimate accesses. Abdul Bashah Mat Ali et al (2011) discussed the development of a new web scanning mechanism with enhanced features that is able to conduct efficient penetration test on PHP based websites to detect SQL injection vulnerabilities. This method automates the penetration test process in order to make it easy even for those who are not aware familiar about hacking techniques.

A multi module vulnerability analysis model for securing web based applications has been proposed by Davide Balzarotti et al (2007). Their approach characterizes both the extended state and the intended workflow of a web application which is able to account inter module relationships as well as interaction of an application’s modules with back end databases. They have implemented their technique in a prototype model and tested it on several applications, identifying both known and new vulnerabilities. Djordjevic and Dimitrakos (2007) introduced dynamic security perimeters for inter enterprise
service integration. Their underpinning security architecture enables the federated management and distributed enforcement of dynamic security perimeters for virtual communities of services, and on resources that span across administrative and enterprise boundaries.

Michelle Zhou and Prithvi Bisht (2010) proposed an approach to strengthen Cross Site Request Forgery (CSRF) defenses for legacy web applications using the white box analysis and transformation. Monica et al (2008) described a language called Program Query Language (PQL) that allows users to provide queries to extract information flow patterns sufficiently and declaratively. For this purpose, they developed a static context sensitive, but flow insensitive information flow tracking analysis that can be used to find all the vulnerabilities in a program. Their language is useful for analysing the input vectors that expose the vulnerability. However, most works consider only numeric constraints. On the other hand, in the current web scenario, symbolic manipulations are necessary to detect and prevent novel attacks. In order to achieve this, it is necessary to propose new algorithms that use symbolic constraints for effective detection of attacks.

2.3.1 Works on Symbolic Constraints

There are many works in the literature that deal with the use of symbolic constraints to test the inputs. In addition, there are few works that discuss about symbolic constraint satisfaction techniques for effective rule management. Among them, Emmi et al (2007) presented a dynamic test input generation algorithm for database applications. This algorithm can track symbolic constraints across language boundaries and uses these constraints in conjunction with a novel constraint satisfaction algorithm to generate both program inputs and corresponding database states. Xiang Fu and Kai Qian (2008) proposed a method SQL injection scanner that uses symbolic
execution for detecting SQL injection vulnerabilities. Their method instruments the code in applications and use symbolic execution to statically inspect security vulnerabilities.

A simple alternative mechanism for preventing script injection called Browser Enforced Embedded Policies (BEEP) was proposed by Trevor Jim et al (2007). According to them, a web site can embed a policy in its pages that specifies which scripts are allowed to run. They have added BEEP support to several browsers and provided mechanisms to simplify adding policies to web applications. Adam Kieyzun et al (2009) presented a technique for finding security vulnerabilities in web applications. Their work presents an automatic technique for creating inputs that expose SQL injections and XSS vulnerabilities that generates sample inputs for analysis. These inputs are used by the algorithm for symbolically tracking taints through execution and mutation to produce concrete exploits. However, the limitation of their system is that it has been designed only for PHP based applications and hence are not suitable for other applications.

Mike Ter Louw et al (2007) proposed an extensible web browser security model that uses code integrity checking techniques to control the extension installation and loading process. They also proposed methodologies for runtime monitoring of extension behavior that provide a foundation for defending threats due to installed extensions. In another work, Mike Ter Louw et al (2008b) presented an analysis on hypertext isolation techniques for XSS prevention which focuses on security, browser compatibility and other important qualities.

Moreover, MikeTer Louw et al (2010) proposed novel techniques for Practical enforcement of confidentiality and integrity policies on web applications by addressing issues related to security threats posed by third party advertisements. Their mechanism support fine grained policy
specification and enforcement, and does not affect the user experience of interactive advertisements. Evaluation of their frameworks suggests the compatibility with several mainstream ad hoc networks security policies that saves the network from many threats.

According to Gary Wassermann and Zhendong Su (2007) specifications can be avoided by queries for which user input changes the intended syntactic structure of the generated query as attacks. They have implemented the proposed technique for PHP and this method successfully discovered unknown and subtle vulnerabilities in real world programs. Avik Chaudhuri and Jeffrey S. Foster (2010) presented a symbolic security analysis on rails for web applications in order to detect cross site scripting attacks. They have considered cross site request forgery, insufficient authentication, leaks of secret information, insufficient access control and application specific security properties in their work.

Yusuke Takamatsu et al (2010) proposed a technique for automatically detecting session fixation vulnerabilities in web applications. They have tested this technique using an attack simulator that executes a real session fixation attack and checks whether it is successful or not. However, most of the works either focused only on constraint management or security management. In this research work, symbolic constraints with constraint satisfaction and propagation using rules has been proposed and implemented for providing effective security.

2.3.2 Intelligent Techniques for Detection of XSS Attacks

Juan Jose Garcia Adeva and Juan Manuel Pikatza Atxa (2007) focused on detecting attempts of either gaining unauthorized access or misusing a web application. They introduced an intrusion detection software component that has been developed using text mining techniques. They used
text categorization techniques capable of learning the characteristics of both normal and malicious user behavior from the log entries generated by the web application server for effective decision making. The detection of misuse in the web application is achieved in their work without the need for any explicit programming or code writing, hence improving the system maintainability.

Kenneth Lingham et al (2007) described a method to use Deterministic Finite Automata (DFA) induction method to detect malicious web requests. They have used DFA in combination with rules for reducing variability among requests and heuristics for filtering and grouping anomalies. With this setup, a wide variety of attacks have been detected with few false positives even when the system is trained on data containing benign attacks. Moreover, Lieven Desmet et al (2006) proposed a solution for bridging the gap between Web Application Firewalls and web applications through a combination of static and dynamic verification. According to these authors, WAFs can formally guarantee the absence of certain kinds of erroneous behavior in web applications. Therefore, they have developed a prototype implementation of their approach using an existing static verification technique.

Another approach for blocking malwares was presented by Weiqing Sun et al (2008) which uses effective filtering mechanisms for providing security. Their technique was able to block malicious code without breaking non malicious ones. Chengying Mao (2009) experienced in security testing for web based applications. His test results show that the security testing framework can provide effectual direction for testing practice and reveal valuable security flaws.

Shahriar and Zulkernine (2011) presented a taxonomy and classification approaches employed for monitoring program vulnerability exploitations. They classified the existing approaches based on a set of
characteristics which are common in online attack detection approaches. Moreover, they presented this taxonomy by classifying the approaches based on monitoring aspects that primarily differentiate among the approaches. In another work, Meier (2006) presented new web application security engineering methodologies that are based on empirical evidence from consulting with hundreds of customers. The author suggested the use of real world scenarios with real project constraints and security concerns across a variety of scenarios and by putting them into practice so that it can follow the security techniques that the experts know. However, the protection systems have been developed to take care of either client side or server side attacks.

2.3.3 Client Side Protection

Engin Kirda et al (2009) proposed a Client side solution to mitigate cross site scripting attacks. They developed a new methodology which uses both manual and automatically generated rules to mitigate possible cross site scripting attempts. Their system effectively protects information leakage from the user’s environment and requires only a minimal user interaction and customization effort. The major limitation of this system is that it protects only on the client side but not on the server side. Philipp Vogt et al (2007) proposed a solution that prevents XSS attacks on the client side by tracking the flow of sensitive information inside the web browser.

Stephen Chong et al (2007, 2009a) developed a compiler that uses policies to automatically partition the program into JavaScript code running in the browser, and Java code running on the server. Their system protects the code and data that are placed on the client side. Moreover, this code and data can be replicated across the client and server to provide both security and performance. Moreover, they also proposed a method to build secure web application via automatic portioning.
Mike Ter Louw and Venkatakrishnan (2009) presented a new XSS defense strategy that is widely deployed in existing web browsers for providing effective security. This approach relaxes the trust placed on browsers for interpreting un-trusted content. They have evaluated blueprints against a barrage of stress tests that demonstrates strong resistance to attacks and provides excellent compatibility with web browsers with reasonable performance overheads. Collin Jackson et al (2009) proposed and implemented a solution for protecting browsers from Domain Name System (DNS) rebinding attacks. Their primary focus is in the design of strong defenses against DNS rebinding attacks that protect modern browsers. Nuo Li et al (2010) proposed a new approach to generate test inputs for User Input Vectors (UIV) based on the analysis of client side information. They used input field information to generate valid inputs and then perturb valid inputs to generate invalid test inputs. They conducted an empirical study which results in comparison to existing vulnerability scanners. Their approach is more effective than the existing vulnerability scanners in finding semantics related vulnerabilities of UIV in web applications.

2.3.4 Server Side Protection

Works pertaining to secure web applications at the server side protection was proposed by few researchers (Pietraszek et al 2006, Thorsten et al 2006). Among them Martin Johns et al (2008) proposed a passive detection system to identify successful XSS attacks. In their work, they have compiled a data set of 500000 individual HTTP request/response pairs from 95 popular web applications in combination with both real word and manually crafted XSS exploits. Moreover, their detection approach results in a total of zero false negatives for all tests, while maintaining an excellent false positive rate for more than 80% of the examined web applications.

Wurzinger et al (2009) introduced secure web application proxy for server side solution to detect and prevent cross site scripting attacks. Their
system comprises a reverse proxy that intercepts all HTML responses, as well as a modified Web browser which is utilized to detect script content. Another work named ‘Better Abstractions for Secure Server Side Scripting’ proposed by Dachaun Yu et al (2008) is a combination of a high level language declarative server side scripting and a low level language reflecting realistic web programming concepts and scenarios, thus articulating the computation model behind web programming.

In another work, Shahriar and Zulkernine (2009) applied the idea of mutation based testing technique to generate adequate test data sets for testing XSS vulnerabilities. They addressed XSS vulnerabilities related to web applications that use Hypertext Preprocessor (PHP) and JavaScript code to generate dynamic HTML contents and their mutants. Their approach also includes a robust mechanism for identifying scripts at the server side and removes any script in the output that is not intended by the web application.

The defense mechanisms for cross site scripting attacks was proposed by Prithvi Bisht and Venkatakrishnan (2008) based on input validation that can effectively prevent XSS attacks on the server side. They also discussed several new XSS attacks and analyzed the reasons for the failure of filtering mechanisms in defending these attacks. They also proposed a new framework called XSS GUARD that has been designed to prevent XSS attacks on the server side. XSS GUARD works by dynamically learning the set of scripts that a web application intends to create for any HTML request. Another approach proposed by Prithvi Bisht et al (2010a) attempts to automatically detect potential server side vulnerabilities in web applications through black box analysis. Their approach has been used to test and discover several previously unknown vulnerabilities in a number of open source web applications and live web sites.

In this work, to secure web applications from XSS attacks, user inputs, cookies information, session details are properly validated and are
monitored by intelligent agents which are deployed at the client side. Similarly, the web application and its data in server side are secured by providing filtering techniques and an intelligent agent that prevents XSS attacks. This work considers both client and server side attacks for prevention.

2.4 RELATED WORKS ON SQL INJECTION ATTACKS

Research on securing web applications from SQL injection attacks has been carried out in the past by many researchers (Sarika Agarwal 2002, Sekar et al 2003, Roberto et al 2008, Toan et al 2010, Kulkarni and Kadeeskar 2010). William G.J Halfond and Alessandro Orsa (2008) proposed a highly automated approach for dynamic detection and prevention of SQL injection attacks. Their approach works by identifying “trusted” strings in an application and by allowing only trusted strings that create the semantically relevant parts of a SQL query such as keywords or operators.

Another approach proposed by Anley (2002) discussed about advanced SQL injection in SQL server applications for protection of web applications. Zhendong Su and Gary Wassermann (2006) explained about command injection attacks in the context of web applications and proposed a sound and complete algorithm for preventing them based on context free grammars and compiler parsing techniques. In order to achieve this, the intended syntactic structures of the query are identified and attacks are detected. However, it does not provide sufficient mechanism to prevent new form of attacks in SQL injections.

A new model to detect illegal queries before they are executed on the database has been proposed by William G.J. Halfond and Alessandro Orso (2005a). Their work is based on static as well as dynamic parts that perform program analysis to automatically build a model of the legitimate queries that are generated by the applications. Moreover, it uses runtime monitoring to
inspect the dynamically generated queries and to check them against the statically built model. Gregory Buehrer (2005) described techniques to prevent SQL manipulation for SQL injection vulnerabilities. Their technique is based on comparing the parse tree of the SQL statement before inclusion of user input with that of the result obtained after inclusion of input. Their solution is efficient and useful to reduce the cost of query processing.

Mcclure and Kruger (2005) proposed the SQL: Document Object Model (DOM) which uses a set of classes that are strongly typed to describe a database schema. Instead of string manipulation, these classes are used to generate SQL statements. They also show how to extract the SQL: DOM automatically from an existing database schema and demonstrate its applicability to solve the problems and to evaluate its performance. William G.J. Halfond and Alessandro Orso (2005b, 2006) proposed new methods to counter SQL injections attacks using positive tainting and syntax aware evaluation. Their work focused mainly on static part and hence it has scalability problem.

2.4.1 Recent Works on SQL Injection Attacks

A provable protection technique against web application vulnerabilities related to session data dependencies was proposed by Lieven Desmet and Pierre Verbaeten (2008). Their technique shows how to formally guarantee that no data dependencies are broken in a given web application. Moreover, this method guarantees the absence of run time errors due to broken data dependencies on session data. Alexer Yip et al (2009) presented a new method for improving application security with data flow assertions called RESIN. RESIN’s runtime checks data flow assertions by propagating policy objects along with data that moves through the application. Moreover, it filters objects when data crosses a data flow boundary. Using RESIN, web
application programmers can prevent a range of problems from SQL injection and cross site scripting to inadvertent password disclosure and missing access control checks.

Angelo Ciampa et al (2010) developed a heuristic based approach for detecting SQL injection vulnerabilities in web applications. Their approach is based on pattern matching of error messages and on outputs produced by the application under test. This system relies upon an extensible knowledge base consisting in a large set of templates. However, it has some limitations in the detection algorithm that decreases the performance of web applications. Konstantinos Kemalis and Theodores Tzeeramanis (2008) developed a prototype named SQL-IDS that implements a specification based approach for SQL injection detection that monitors Java based applications and detects SQL injection attacks in real time. They have also proposed a query specific detection technique that allows the system to perform analysis at negligible computational overhead without producing false positives or false negatives.

Martin Bravenboer et al (2010) developed methodologies for preventing SQL injection attacks with syntax embeddings in programming. Their approach embeds the grammars of the guest languages into the host language that automatically generates code for mapping the embedded language that constructs the host language which reconstructs the embedded sentences by adding escaping functions appropriately. Xiang Fu and Kai Qian (2008) developed the system for detecting SQL injection vulnerabilities resident in web applications. This system instruments the bytecode of java web applications and utilizes symbolic execution to statically inspect security vulnerabilities.

The framework for mining programmer intended queries that dynamically evaluate runs over benign candidate inputs was proposed by
Prithvi Bisht et al (2007, 2010b). Their mechanism is based on inferring intended queries to consider the symbolic query computed on a program run. This approach has been named as CANDID that retrofits web applications written in Java to defend them against SQL injection attacks. In addition to this, they also proposed new approach to automatically prepare safe SQL queries which are executed in the database. Artzi et al (2010) developed a solution for finding bugs in web applications using dynamic test generation and explicit state model checking in web applications. However, their work is limited only for PHP applications.

Stephen Thomas et al (2009b) presented an algorithm of automated PREPARED statement generation for removing SQL Injection Vulnerabilities (SQLIV) by replacing SQL statements with PREPARED statement. The PREPARED statement has a static structure, which prevents SQL injection attacks from changing the logical structure of a PREPARED statement. They created a PREPARED statement replacement algorithm and a corresponding method for automated fix generation. In this work, new techniques have been proposed for effective detection and prevention of SQL injection attacks.

2.4.2 Works on Classification Techniques

In the past, few works have been proposed and implemented to secure web applications using classification techniques. Among them Zhou et al (2002, 2003) proposed the discovery of classification rules by using gene expression programming with linear representation. The antecedent of discovered rules may involve many different combinations of attributes. Their approach is also noise tolerant and is able to deal with both numeric and nominal attributes.

Expression Programming for detecting web application attacks. They considered the malicious activity of an intruder against applications as attack which uses a database for storing data. Peter Kok Keong and Deepak Subramaniam (2010) introduced fuzzy classification metrics for scanner assessment and vulnerability reporting based on soft computing techniques for the development of metrics for validating vulnerabilities reports. These metrics help to derive a level of assurance that supports security management decisions and enhance effective remediation efforts and thus serve as security design metrics. In their work, the intrusion detection problem is transformed into classification problem, where the objective is to classify SQL queries between either normal or malicious queries.

In this thesis, classification techniques are used for preventing SQL injection attacks since classification of the user as normal and abnormal and transformation of static query and dynamic query is important in preventing SQL injection attacks. Hence, new techniques have been proposed in this work to classify the user queries as normal and abnormal based on the past history stored in the knowledge base which uses rules to take decision on such queries. These classified user queries are again checked with intelligent prevention agents that detects and prevents SQL injection attacks in web applications.

2.5 ANOMALY INTRUSION DETECTION SYSTEMS

Anomaly intrusion detection systems gained importance during the last decade since they are effective in preventing external attackers. For this purpose, behavior analysis and access control techniques have been employed in the past.
2.5.1 Works on Anomaly Intrusion Detection Techniques

In the past, many techniques have been proposed and implemented to detect and prevent web based attacks by many researchers (Axelsson 2003, Wang et al 2009, Khaleel Ahamed et al 2011). Some of the important cited works have reported here.

A signature based method for detecting web server attacks was proposed by Almgren et al (2000). Their design was flexible in signatures used to detect malicious behavior that are not limited to simple pattern matching of dangerous CGI scripts. They included mechanisms to reduce the number of false alarms. Another work that operates on signatures extracted from execution profiles was proposed by El-Ghali et al (2009). In their work, they first analysed the execution of profiles of a subject application that identifies suspicious combinations and their corresponding signatures. Denning (1987) presented an Intrusion Detection Model that is based on the hypothesis that security violations can be detected by monitoring a system's audit records for abnormal patterns of system usage. Their model includes profiles for representing the behavior of subjects with respect to objects in terms of metrics and statistical models, and rules for acquiring knowledge about this behavior from audit records and for detecting anomalous behavior.

Kymie M. C. Tan et al (2002) presented a method that identifies the weaknesses of an anomaly based intrusion detector, and shows how an attacker can manipulate common attacks to exploit those weaknesses. Their work explores the implications of this threat, and suggests possible improvements for existing and future anomaly based intrusion detection systems. Christopher and Giovanni (2003) proposed a new technique called anomaly intrusion detection to detect web based attacks in the server. Their technique analyses the length, structure of the parameters and associated profiles in web applications in order to reduce the false positives. Moreover,
David Wagner and Paolo Soto (2002) discussed the mimicry attacks on host based intrusion detection systems that introduce the notion of a mimicry attack, which allows a sophisticated attacker to cloak their intrusion to avoid detection by the IDS. They also show how to break the security of one published IDS with these methods.

Christopher Kruegel et al (2005) proposed a multi model approach to detect web based attacks. Their model analyzes client queries that refer server side programs to create models for wide range of different features of these queries. Their work is related to three different areas of intrusion detection, namely learning based anomaly detection, application level intrusion detection, and, more specifically, detection of attacks against web servers. But their work is limited to the simulation of web applications and not to real time web applications which also has more false positive ratio.

Wes Masri and Andy Podgurski (2008) presented an approach to detect and prevent software security failures. Their approach is based on online capture of executions and offline execution replays, profiling and analysis that employs fine grained dynamic information flow analysis in conjunction with anomaly detection. Their approach is capable of detecting a variety of security failures, including both ones that involve violations of confidentiality or integrity requirements.

Giovanni et al (2009) proposed a new technique that combines the web based anomaly detector and SQL anomaly detector in order to increase the detection rate of their system which reduces the errors in web based attacks by anomaly intrusion detection system through combined analysis of web request and SQL queries. Marco Cova et al (2010) presented a novel approach for the detection and analysis of malicious JavaScript code. Their approach combines anomaly detection with emulation to automatically identify malicious Java Script code and to support its analysis. Their system
uses a number of features and machine learning techniques to establish the characteristics of normal JavaScript code. During detection, the system is able to identify anomalous JavaScript code by emulating its behavior and comparing it to the established profiles. In this work, a new modified classification technique has been proposed to detect and prevent intrusions in web applications.

### 2.5.2 Policy Based Access Control Techniques

In the past, few works have been proposed and implemented by researchers (Marian et al 2003, Miller 2006). Among them Marian Ventuneac et al (2003) proposed a policy based security framework to secure web applications. This method enforces policies for authentication, access control, security management, identity administration and accountability. David Scott and Richard Sharp (2002, 2003) proposed a new method to analyse the code in web applications. This method performs code analysis in order to find the bugs and vulnerabilities in web applications by executing the specifications and policies provided in this new technique. In addition to this, they have also proposed the idea of developing secure web applications by abstracting code in web applications in the application level itself.

The attack prevention framework for securing web applications was proposed by Garcia Alfaro et al (2005b). Their method make use of an access control mechanism which are embedded into the operating system’s kernel in order protect the web applications and its data which are compromised by an attacker. Frank S. Rietta (2006) examined the threat from SQL injection attacks by intrusion detection system. They proposed a model to detect anomalous SQL which observes the database traffic from the perspective of the database server itself. Their proposed anomaly model is used in conjunction with the existing methods to secure the database server from SQL injection attacks.
Fredrik Valeur et al (2006) developed an anomaly based system that learns the profiles of the users and access to the database performed by web applications to detect SQL attacks. This model allows for the detection of unknown attacks with reduced false positives and limited overhead. Moreover, their approach uses multiple models to characterize the profiles of normal access to the database, where profiles are learned automatically during a training phase by analyzing a number of sample database accesses. In addition to this, they proposed an approach that composes a web based anomaly detection system with reverse HTTP proxy. Their approach is based on the assumption that a web site's content can be split into security sensitive and non sensitive parts, which are distributed to different servers. They also developed a prototype of their approach and evaluated its applicability with respect to several existing web based applications, showing that their approach is both feasible and effective.

Manar H. Alalfi et al (2009) proposed a verification framework for access control in dynamic web applications. A reverse engineering process is performed by them over a dynamic web application to extract a role based access control security model. They applied formal analysis on the recovered model to check access control security properties. This framework can be used to verify that a dynamic web application conforms to access control polices specified by a security engineer. Guillaume et al (2009) proposed a new technique which combines policy-based intrusion detection and information flow control that analyses the information flow in the operating system level to detect wide range of web abased attacks in web applications.

In this thesis, an Intelligent Anomaly Intrusion Detection System that accurately detects attacks intrusions against web applications with low performance overhead which provides accuracy of detection and low false alarm rates has been proposed. This newly proposed technique has higher
probability to reduce false positives which are the drawbacks in the earlier systems. Finally, a novel framework for developing web applications that are secure by construction against many common classes of attacks is presented in this thesis.

2.6 PROPOSED WORK

There are many major contributions that are involved in thesis work with respect to SQL injections and cross site scripting attacks. First, this work focuses on preventing cross site scripting attacks both at the client and the server side by introducing new techniques at the client and server with intelligent agents to effectively prevent such attacks. Second, this thesis focuses on preventing SQL injection attacks at the server by proposing a new algorithm called modified decision tree algorithm in order to prevent such attacks by classifying the user queries as normal and abnormal queries. Third, this thesis proposes a role based access control module that provides fuzzy rules and access control manger to prevent SQL injections and anomaly intrusions in web applications. Finally, an architectural frame work has been proposed that provides techniques and intelligent rules to secure web applications from SQL injections and cross site scripting attacks.