IJCSE International Journal of Computer Sciences and Engineering Open Access
Survey Paper Volume-3, Issue-11 E-ISSN: 2347-2693

A Survey on Content Injection Attacks

Sandeep D Sukhdeve[1*] and Hemlata Channe[2]

[1*]*Post Graduate Department of Computer Science and Engineering (PICT), Pune University, India*
[2]*Assistant Professor , Pune Institute of Computer Technology(PICT),Pune University, India*

www.ijcseonline.org

Received: Nov /05/2015 Revised: Nov/14/2015 Accepted: Nov/20/2015 Published: Nov/30/2015

Abstract— We are increasingly relying on web, and performing important transactions online through it. At the same time, quantity and impact of security vulnerabilities in such applications has grown as well. This work presents a survey of web security research which is the emerging domain that implements various detection prevention techniques for hinder content injection attacks on web applications. This paper provides a classification of the research areas on the content injection attacks. In this paper, we analyze important aspects in content injection attacks. In addition, this paper presents a survey of various security mechanisms adopted by web browsers to defend content injection attacks. The goals of this survey paper are two-fold:
i) Serve as a guideline for researchers, who are new to web security and want to contribute to this research area, and
ii) Provides further research directions required into content injection attack prevention.

Keywords— XSS(Cross-site scripting), SQLI(Structural Query Language Injection),Content Injection,SQL queries.

I. INTRODUCTION

Nowadays many activities are performed by dynamic web applications. For example users pay their utility bills, book the hotels or air tickets by dynamic websites to save time and money. It is crucial that user data must be kept secret. That is, confidentiality and integrity of user data must be provided by developers of the web application but unfortunately there is no such guarantee for preserving the underlying web application from various content injection attacks. Content injection attacks can compromise confidentiality and integrity of information in the vulnerable web application.

According to the Symantec security threat report of 2015 [29], cross-site scripting is listed on number two out of top ten web application vulnerabilities. Web sites vulnerable to content injection attacks are from various categories such as blogging, hosting, entertainment, shopping, sports, etc. According to the Open Web Application Security Project (OWASP) 2013 top ten web application vulnerability report [23], injection attack is a number one web application vulnerability and cross-site scripting listed at number three among the top ten web application vulnerabilities.

In 2011, the National Institute of Standards and Technology National Vulnerability Database [18] reported 289 SQL injection vulnerabilities (7 percent of all vulnerabilities) in websites, including those of IBM, Hewlett-Packard, Cisco, WordPress, and Joomla. In December 2011, SANS Institute security experts reported a major SQL injection attack (SQLIA) that affected approximately 160,000 websites using Microsoft's Internet Information Services (IIS), ASP.NET, and SQL Server frameworks [10].

Corresponding Author: *Sandeep Sukhdeve, sukhdevesandeep@gmail.com*
Post Graduate Department of Computer Science and Engineering(PICT), University of Pune, India.

Inadequate validation and sanitization of user inputs make websites vulnerable to content injection, and re-searchers have proposed various ways to address this problem, ranging from simple static analysis to complex dynamic analysis.

A. *Content injection and its types*

Content injection attack refers to inserting malicious content into a legitimate site. There are two types of con-tent injection attacks namely SQL injection and cross-site scripting.

SQL Injection: SQL Injection (SQLI) attack is a prevalent attack technique that allows attackers to gain direct access to the web application database and extract sensitive information from the victim web application's database [33].
Cross-site Scripting (XSS): Cross-site scripting (XSS) is a type of web application vulnerability that enables attackers to inject client-side script into web pages viewed by other users. Attackers can use cross-site scripting vulnerability to bypass access controls such as the same-origin policy (SOP) [2], [32]. Some examples of real-world cross-site scripting attacks can be found at [6], [16], [25], [35], [36].

B. *Observations*

We observed that the aim of content injection attack to gain illegal access to user data. The Structural Query

Language Injection (SQLI) attack occurs when an attacker changes the logic, semantics or syntax of a SQL query by inserting new SQL keywords or operators. SQL Injection attack is a class of content injection attacks that occurs when there is no input validation mechanism deployed by web developers in the web application.

In cross-site scripting attack, the attackers fold malicious content into the content being delivered from the compromised site. When the resulting combined content arrives at the client-side web browser, it has all been delivered from the trusted source, and thus operates under the permissions granted to that system. By finding ways of injecting malicious scripts into web pages, an attacker can gain elevated access-privileges to sensitive page content, session cookies, and a variety of other information maintained by the browser on behalf of the user [32]. The successful XSS attack is a result of lack to provide input validation in the web application by the developers.

Too many existing techniques are either not publicly available or are difficult to adopt. Readily available tools would motivate more developers to combat content injection attacks. Developer's unawareness of security mechanisms and content injection sanitization can result in data loss or corruption, lack of accountability, or denial of access. Injection can sometimes lead to complete host takeover. Therefore, it is important to provide a solution that protects web applications from SQLI and XSS attacks. This paper performs the survey of various techniques proposed to protect web applications from these attacks.

C. *Contributions*

In summary, this paper makes the following contributions:

1) Classifies state-of-the-art research performed on detection and prevention of content injection attacks.
2) Provides analysis of important aspects in content injection attacks.
3) Provides guidelines and further research directions required in the content injection attacks prevention.

The rest of this paper is organized as follows: Section II explains content injection attack vectors (SQL injection and Cross-site Scripting attack) with the help of example. Section III describes our motivation. Section IV discusses the literature survey. Section V suggests open research question in content injection, and we conclude the paper in Section VI.

II. BACKGROUND

A. *An Example of SQL Injection Attack*

Typically users are requested to provide some input data on web pages (for example, username and passwords) and web applications make a SQL query to the database based on the information received from the user. Malicious user can send crafted input to change the SQL statement structure and execute arbitrary SQL commands on the vulnerable system.

Let's consider an example of web application that accepts username and password for users in order to allow authenticate users to login to the web site. When user input is received at server side by the web application, following SQL query is created and executed by the web application to verify credentials provided by the user:

SELECT * FROM usertable WHERE userID = 'Sandeep' and password = 'abc123'

Assume malicious users provided following crafted input in the password input box:
' or 1=1 --

The SQL query in the web application will become:

SELECT * FROM usertable WHERE userID = Sandeep and password = '' or 1=1 --'

The **"or 1=1"** will make the query TRUE and results in returning all the records in the "usertable" to the malicious user. The **"-"** comments out the last **'** character appended by the web application.

B. *An Example of Cross-site Scripting Attack*

Attacker can inject scripts in a web page by reflection. For example, when asked for a non-existent page on the web server, many websites try to produce a helpful not found response that includes the URL of the non-existent page that was requested. Therefore, if the developers of website are not carefully sanitizing the user inputs, an occurrence of the text < script > …. < /script > in the URL can be executed in the visitor's browser when it renders the not found page. To exploit this, an attacker can try to entice victims to follow URLs with targets that include scripts, e.g.,
http://trusted.site/<script>document.location='http://malicioussite.com/?'+document.cookie</script>

The attacker could place the URL in a spam e-mail, in a blog comment on trusted.site, or even on another website. If a victim follows the link, the script will run in the not found page served by trusted.site, retrieve the user's trusted.site cookie, and send it to malicioussite.com.

III. MOTIVATION

Traditionally, content injection was limited to personal computing environments. However, the increasing use of smart phones, tablets, and other portable devices has extended this problem to mobile and cloud computing environments, where vulnerabilities could spread much faster and become much easier to exploit. We were motivated to perform this survey in order to enumerate and compare state-of-the-art research that proposed techniques to prevent content injection attacks. This study paper provides analysis and summarizes various proposed solution for prevention of SQL injection and cross-site scripting attacks. This survey can become the starting point for anyone trying to understand, evaluate and develop techniques for web security.

IV. LITERATURE SURVEY

A. SQL Injection Detection and Prevention Solutions

Several solutions that mitigate the risk posed by SQL Injection attacks have already been proposed [1], [7]–[9]. All of these solutions have been successful in mitigating SQL Injection attacks. However, none of these solutions address the actual SQL injection attack that exists in the source code. A common way to remove SQL injection vulnerability is to separate the SQL structure from the SQL input by using prepared statements. Stephen et.al. [30] proposed a prepared statement replacement algorithm and a corresponding tool for automated fix generation.

Cristian [22] et.al. presented a hybrid approach based on the Adaptive Intelligent Intrusion Detector Agent (AIIDA-SQL) for the detection of SQL injection attacks. The AIIDA-SQL agent incorporates a Case-Based Reasoning (CBR) engine which is equipped with learning and adaptation capabilities for the classification of SQL queries and detection of malicious user requests. To carry out the tasks of attack classification and detection, the agent incorporates advanced algorithms in the reason-ing cycle stages. Concretely, an innovative classification model based on a mixture of an Artificial Neuronal Network together with a Support Vector Machine is applied in the reuse stage of the CBR cycle. This allowed classification of SQL queries.

Michelle [27] et. al. proposed a technique that is based on automatically developing a model for a SQL query such that the model captures the dependencies between various components (sub-queries) of the query.

The authors analyzed the model using CREST test-case generator and identify the conditions under which the query corresponding to the model is deemed vulnerable. The authors further analyzed the obtained condition set to identify its subset; this subset being referred to as the causal set of the vulnerability. The technique proposed by the authors considers the semantics of the query conditions, i.e., the relationship between the conditions, and as such complements the existing techniques which only rely on syntactic structure of the SQL query. In short, the technique proposed by the authors can detect vulnerabilities in nested SQL queries.

B. Cross-site Scripting Detection and Prevention Solutions

Mozilla has a feature called signed scripts [26]. Scripts are signed when they require additional privileges, such as writing to local files, and the absence of a signature does not constrain scripts. Server-side techniques to protect against script injection attacks have been reported extensively in the literature. A systematic approach to filtering injected attacks involves partitioning trusted and untrusted content into separate channels and subjecting all untrusted content to application defined sanitization checks [21]. Su and Wassermann [28] develop a formal model for command injection attacks and apply a syntactic criterion to filter out malicious dynamic content. Applications of taint checking to server programs that generate content to ensure that untrustworthy input does not flow to vulnerable application components have also been explored [17], [34]. UserCSP [20] is a Mozilla tool that allows security savvy users to specify and en-force content security policy to protect themselves from cross-site scripting attacks. The tool automatically infers content security policies for the websites user visits and enforces them to protect users from XSS attacks. Other solutions [4], [5] need browser modifications to identify untrusted or malicious scripts from trusted scripts.

MashupOS [31] makes the browser a multi-principal operating system for Web applications. BEEP [13] lets Web sites restrict the scripts that run in each of their pages. ConScript [14] enforces application-specified security policies. OMash [3] restricts communication to public interfaces declared by each page. Kailas [12], [19] proposed a solution to isolate untrusted scripts included in web applications from the trusted scripts. It allows isolation of Javascript context for scripts from different origins. In addition, it also provides different privileges of read and write to scripts running in isolated Javascript contexts.

BrowserShield [24] propose to defeat JavaScript-based attacks by rewriting scripts according to a security policy prior to executing them in the browser. In BrowserShield, the rewriting process inserts trusted JavaScript functions to mediate access to the document tree by untrusted scripts.

Jackson et al. [11] describe several unexpected repos-itories of private information in the browsers cache that could be stolen by XSS attacks. They advocate applying a refinement of the same-origin policy [15] to cover aspects of browser state that extend beyond cookies. By allowing the server to explicitly specify the scripts that it intentionally includes in the document, our approach can also be thought of as an extension of the same-origin policy. Other research efforts [37], [38] proposed various security solutions.

V. RESEARCH QUESTIONS

To solve research problems in web security needs to address the research challenges in prevention of cross-site scripting, and SQL injection attacks. The need to web security solutions generates a number of important research questions:

- To provide rich user interface and web 2.0 features, how can user tasks be modeled and analyzed automatically by the system?
- How user intent is determined correctly? Does it implicitly determine by the system or system needs explicit user interactions.
- How to isolate trusted inputs from untrusted inputs? Where to implement protection solution (application level, network level, client-side or database)?
- How to include third-party content or untrusted content safely in web applications.
- How to identify user input and web application data.

Research in web application security is crucial because it is a fusion of a research in many disjoint areas.

VI. CONCLUSION

Traditionally, content injection was limited to personal computing environments. However, the increasing use of smart phones, tablets, and other portable devices has extended this problem to mobile and cloud computing environments, where vulnerabilities could spread much faster and become much easier to exploit.

In this paper, we presented a survey of SQL injection and Cross-site scripting prevention research. This paper analyzed important aspects in content security systems. This survey paper serves as a guideline for researchers who are new to web security and want to contribute to this research area.

REFERENCES

[1] G. Buehrer, B.W. Weide, and P.A.G. Sivilotti. "Using parse tree validation to prevent sql injection attacks". In Proceedings of the 5th International Workshop on Software Engineering and Middleware, **2005.**

[2] CGIsecurity. The cross-site scripting (xss) faq. http://www.cgisecurity.com/xss-faq.html.

[3] S. Crites, F. Hsu, and H. Chen. Omash: "Enabling secure web mashups via object abstractions". In Proceedings of the International Conference on Computer and Communications Security (CCS), **2008.**

[4] Xinshu Dong, Kailas Patil, Xuhui Liu, Jian Mao, and Zhenkai Liang. "An entensible security framework in web browsers". Technical Report TR-SEC-2012-01, Systems Security Group, School of Computing, National University of Singapore, **2012.**

[5] Xinshu Dong,Kailas Patil, Jian Mao, and Zenkai Liang. "A comprehensive client-side behavior model for diagnosing attacks in ajax applications". In proceedings of the 18th International Conference on Engineering of Complex Computer systems (ICECSS), **2013.**

[6] Dennis Fisher. Persistent XSS bug on twitter exploited by worm http://threatpost.com/en us/blogs/persistent-xss-bug-twitter-being-exploited-092110

[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, **2005.**

[8] W.G.J. Halfond and A. Orso. "Combining static analysis and runtime monitoring to counter sql-injection attacks". In Proceed-ings of the Third International Workshop on Dynamic Analysis, **2005.**

[9] W.G.J. Halfond, A. Orso, and P. Manolios. "Using positive tainting and syntax-aware evaluation to counter sql-injection at-tacks". In Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering, **2006.**

[10] Mark Hofman. Sql injection attack happening atm. isc.sans.org/diary/SQL+Injection+Attack+happening+ATM/12127.

[11] Collin Jackson, Andrew Bortz, Dan Boneh, and John C. Mitchell. "Protecting browser state from web privacy attacks". In Proceedings of the International Conference on World Wide Web (WWW), **2006.**

[12] Patil Kailas, Dong Xinshu, Li Xiaolei, Liang Zhenkai, and Jiang Xuxian. "Towards fine-grained access control in javascript contexts". In Proceedings of the International Conference on Distributed Computing Systems, **2011.**

[13] Ziqing Mao, Ninghui Li, and Ian Molloy. "Defeating cross-site request forgery attacks with browser-enforced authenticity protection". In Financial Cryptography and Data Security, 13th International Conference, **2009.**

[14] Leo A. Meyerovich and Benjamin Livshits. "ConScript: Specify-ing and enforcing fine-grained security policies for javascript in the browser". In Proceedings of the IEEE Symposium on Security and Privacy (IEEE S & P), **2010.**

[15] Mozilla Same origin policy for javascript. https://developer.mozilla.org/En/Same_origin_policy_for_javascript.

[16] The clickjacking meets xss: a state of art. http://www.milw0rm.com/papers/265, **2008.**

[17] Anh Nguyen-tuong, Salvatore Guarnieri, Doug Greene, Jeff Shirley, and David Evans. "Automatically hardening web appli-cations using precise tainting". In Proceeding of the 20th IFIP International Information Security Conference, **2005.**

[18] National Institute of standards and technology. National vulnerability database (nvd) http://web.nvd.nist.gov/view/vuln/search

[19] Kailas Patil Ensuring session integrity in the browser environment http://scholarbank.nus.edu.sg/bitstream/handle/10635/49161/ThesisHT080141L.pdf?sequence=1, **2013.**

[20] Kailas Patil, Tanvi Vyas, Fredrik Braun, and Mark Goodwin. "Usercsp- user specified content security policies". SOUPS'13 POSTER, **2013.**

[21] Tadeusz Pietraszek, Chris V, and En Berghe. "Defending

against injection attacks through context-sensitive string evaluation". In Proceeding of the Recent Advances in Intrusion Detection, **2005.**

[22] Cristian Pinzn, Javier Bajo Juan F. De Paz, lvaro Herrero, and Emilio Corchado. "Aiida-sql: An adaptive intelligent intrusion detector agent for detecting sql injection attacks". In Proceedings of the 10th International Conference on Hybrid Intelligent systems **2010.**

[23] OWASP-The open web application security project. OWASP top ten project. https://www.owasp.org/index.php/Top_10_2013-Top_10

[24] Charles Reis, John Dunagan, Helen J. Wang, Opher Dubrovsky, and Saher Esmeir. "Browsershield: Vulnerability-driven filtering of dynamic html". In Proceedings of the Symposium on Oper-ating Systems Design and Implementation (OSDI), **2006.**

[25] RSnake. Xss(cross site scripting) cheat sheet esp: for filter evasion. http://ha.ckers.org/xss.html.

[26] Jesse Ruderman. Signed scripts in mozilla. http://www.mozilla.org/projects/security/components/signed-scripts.html.

[27] Michelle Ruse, Tanmoy Sarkar, and Samik Basu. "Analysis & detection of sql injection vulnerabilities via automatic test case generation of programs". In Proceedings of the Annual International Symposium on Applications and the Internet, **2010.**

[28] Zhendong Su and Gary Wassermann. "The essence of command injection attacks in web applications". In Proceedings of the ACM Symposium on Principles of Programming Languages (POPL), **2006.**

[29] Symantec. Internet security threat report volume 20. https://www4.symantec.com/mktginfo/whitepaper/ISTR/21347932 GA-internet-security-threat-report-volume-20-2015-social v2.pdfg, **April 2015.**

[30] Stephen Thomas, Laurie Williams, and Tao Xie. "On auto-mated prepared statement generation to remove sql injection vulnerabilities". In Proceedings of the Elsevier Journal on the Information and Software Technology, **2009.**

[31] H. J. Wang, X. Fan, J. Howell, and C. Jackson. "Protection and communication abstractions for web browsers in mashupos". In Proceeding of the SOSP, **2007.**

[32] Wikipedia Cross site scripting. https://en.wikipedia.org/wiki/Cross-site_scripting

[33] Wikipedia SQL injection https://en.wikipedia.org/wiki/SQL_injection

[34] Yichen Xie and Alex Aiken. "Static detection of security vulnerabilities in scripting languages". In Proceedings of the USENIX Security Symposium, **2006.**

[35] xssed.com. Myspace.com hit by a permanent xss. http://www. xssed.com/news/83/Myspace.com hit by a Permanent XSS/.

[36] xssed.com. New orkut xss worm by brazilian web secu-rity group. http://www.xssed.com/news/77/New Orkut XSS worm by Brazilian web security group/.

[37] K.S.Wagh, Vishal Jotshi, Harshal Dalvi, Manish Kamble. "Reversed proxy based XSS filtering". In Proceeding of the International Journal on Computer Science and Engineering (IJCSE). Vol -3, Issue-5,Page No(175-180) May **2015**.

[38] Jyotsnamayee Upadhyaya, Namita Panda, Arup Abhinna Acharya. "Attack Generation and Vulnerability Discovery in Penetration Testing using Sql Injection". In Proceeding of the International Journal on Computer Science and Engineering (IJCSE). Vol -2, Issue-3,Page No(167-173) March **2014**.

AUTHORS PROFILE

Sandeep Sukhdeve, is currently pursuing final year M.E. in Computer Science and Engineering from Pune Institute of Computer Technology (PICT), Pune.

Mrs.Hemlata Channe,
Assistant Professor,M.E.(CE), Pune Institute of Computer Technology(PICT),Pune.