Cross Site Scripting: An Overview

Vishwajit S. Patil
Department of MCA
P.R.M.I.T.& R. Bandera, Amravati

Dr. G. R. Bamnote
Professor & Head,
Department of CSE
P.R.M.I.T.& R. Bandera, Amravati

Sanil S. Nair
Department of MCA
P.R.M.I.T.& R. Bandera, Amravati

ABSTRACT
This paper describes the security attacks and specially focuses on Cross Site Scripting attacks. It further also discusses types and several counter measures. The major problem faced by the web application is the parameter manipulation, through which the attackers are aiming to access the database. Generally web applications maintain same structure and value. In that, required information is being accessed by the identical variables and keywords through web parameters. Parameter manipulation is the major issue in the web application used by the attacker to manipulate the parameter being sent by the browser and executed by the server.

These vulnerabilities occur after the string gets returned to the user's web browser by a susceptible web application. Therefore, to prevent XSS vulnerabilities, it is obligatory to prepare preventative measures to protect the parsing processing in the web browser so that there is no influence even from the effect of the string prepared by the attacker.

1. INTRODUCTION
Web Application have become one of the most important ways of information communication between various kinds of users and service providers .the rapid growth of internet resulted in feature rich, dynamic web application. This increase resulted in the harmful impact of security flaws in such applications. Vulnerabilities leading to compromise of sensitive information are being reported continuously, resulting in ever increasing financial damages.

Cross site scripting better known as XSS, is the most widespread and harmful web application security issue. This flaw occur whenever a web application takes data that originated from user and sends it to a web browser without first validating or encoding that content. This attack could pose a serious security threat. If an attacker made a specially crafted link and sent it to an unsuspecting victim and that victim clicked the link and a piece of Java Script code could be executed which would send the victim's cookie away to a CGI Script, obviously the attack could do some serious damage [Tiwari et al., 2008]. Cross Site Scripting (XSS) is one of the most common application layer attacks that hackers use to sneak into web applications [Shanmugam & Ponnavaikko, 2007]. XSS exploits flaws in web applications which allow attackers to execute arbitrary code without the authorization of the web application. This way, an unaware user can be the victim of an identity theft, electronic fraud or other modalities of cyber-crime [Galan, et al., 2010].

XSS is a web application level vulnerability that can be used by the malicious third party to easily bypass the cookie protection mechanism. Since the vulnerability resides at the web server site, various server side solutions are proposed for protecting users from the XSS attack. But most of them usually degrade the server performance gracefully and cause tremendous configuration overhead [Ismail et al., 2004].

Web Cohort's Application Defense Center research report states that 80% of the web applications are vulnerable to XSS vulnerability. Application worm takes advantage of these XSS vulnerabilities for self replication. The attack involves three primary parties, the malevolent payload, the browser (victim) and the vulnerable web pages in the web server. Web developers are using the mishmash of web technologies to provide better user know-how and to use the bandwidth effectively. Implementation of such new technologies increases the vulnerability of the web applications for XSS attacks [Shanmugam & Ponnavaikko, 2007]. XSS vulnerabilities appear because the structure of the HTML document is modified by the effect of the string that contains the malicious script prepared by an attacker. These vulnerabilities occur after that string is returned to the user's web browser by a susceptible web application. Therefore, to prevent XSS vulnerabilities, it is obligatory to prepare preventative measures to protect the parsing processing in the web browser so that there is no influence even from the effect of the string prepared by the attacker [Iha & Doi, 2009]. Websites that are susceptible to XSS attacks are running some sort of dynamic content, dynamic content is anything that changes due to user interaction or information stored in a database about a user, things such as Forums, web based email and places where information is submitted are vulnerable to XSS attacks.

An HTML-injection attack can occur because the browser fails to sanitize arbitrary input that will be displayed in its History Search results. The attacker's malicious script will have full access to the browsing history.

A cross-site scripting attack can occur via the 'q' parameter of the history search feature.

An origin-validation attack can occur via the browser's preferences configuration option when used in conjunction with the history search feature. Attackers can exploit this issue to, for example, configure a remote proxy or define arbitrary handlers for mail events. An attacker may be able to obtain sensitive information or execute arbitrary local programs within the context of the browser [Shanmugam & Ponnavaikko, 2007].

There are largely two distinct countermeasures for XSS prevention in"real-life" web applications: Input filtering and output sanitation. Input filtering describes the process of validating all incoming data. "Suspicious" input that might contain a code injection payload is either rejected, encoded, or the "offensive" parts are removed using so called "removal filters". The protection approach implemented by these filters relies on removing predefined keywords, such as document. Such filtering approaches are, however, error-prone due to incomplete keyword-lists or non-recursive implementations. If output sanitation is employed, certain characters, such as <, ", or ', are HTML encoded before user-supplied data is inserted into the outgoing HTML. As long as all untrusted data is"disarmed" this way, XSS can be prevented.

Both of the above protections are known to frequently fail [Johns et al., 2008] either through erroneous implementation, or because they are not applied to the complete set of user-supplied data

2. LITERATURE REVIEW

Integrating security throughout the life cycle can improve overall web application security [Meir, 2006]. Web applications are widely adopted in today's life. More and more individuals and organizations strongly depend on their correct functioning, resulting in an increasing demand for reliability and security [Desmet et al., 2008]. There are number of web applications available and day by day the use of it increasing as numbers of users are increasing. So that users are facing problem of security. Our studies will emphasis on identifying the security problems and try to solve it.Providing document security in XML-based Web services requires access control models that offer specific capabilities suggested by authers[Bhatti et al., 2004]. Their XML-based access control specification language addresses a new set of challenges that traditional security models do not address. In [Chien, 2006] author proposed a new digital signature scheme, and claimed the scheme can resist the forgery attack without using one-way hash function and any redundancy padding. This claim is very interesting to all designers, because conventionally a one-way hash function is required to resist the attacks. Their article shows an existential forgery attack on the scheme, and shows that the scheme would still be insecure even if a secure one-way function were adopted in the scheme. [Curphey & Arawo, 2006] studied Web applications and Web services for security vulnerabilities, along with each type's advantages and disadvantages. [Desmet et al., 2008] focused on one specific type of implementation vulnerability, namely, broken dependencies on session data. This vulnerability can lead to a variety of erroneous behavior at runtime and can easily be triggered by a malicious user by applying attack techniques such as forceful browsing. Their paper shows how to guarantee the absence of runtime errors due to broken dependencies on session data in Web applications. The proposed solution combines development-time program annotation, static verification, and runtime checking to provably protect against broken data dependencies. They have developed a prototype implementation of our approach, building on the JML annotation language and the existing static verification tool ESC/Java2, and successfully applied their approach to a representative J2EE-based e-commerce application.

[Galan, et al., 2010] depicted that a novel multi-agent system for the automated scanning of web sites to detect the presence of XSS vulnerabilities exploitable by an stored-XSS attack. The rate of detection of the system is evaluated in two different scenarios over here. [Gebre et al., 2010] suggested a server-side ingress filter that aims to protect vulnerable browsers which may treat non-HTML files as HTML files. The filter over here examines user uploaded files against a set of potentially dangerous HTML elements (a set of regular expressions). The result of the experiment shows that the proposed automata-based scheme is highly efficient and more accurate than existing signature-based approach. [Halfond et al., 2008] expressed a new highly automated approach for protecting Web applications against SQL injection that has both conceptual and practical advantages over most existing techniques. From a conceptual standpoint, the approach is based on the novel idea of positive tainting and on the concept of syntax-aware evaluation. They claimed that this technique is precise and efficient, has minimal deployment requirements, and incurs a negligible performance overhead in most cases. They have implemented techniques in the Web application SQL-injection preventer (WASP) tool, which they used to perform an empirical evaluation on a wide range of web applications that were subjected to a large and varied set of attacks and legitimate accesses. They further claimed that WASP was able to stop all of the otherwise successful attacks and did not generate any false positives.

[Holz et al., 2006] describes some of the new threats like workstations: client-side attacks have increased because direct attacks on servers aren't so easy any more. Moreover, as new defenses are raised, information flows are increasingly embedded into Web applications, making them extremely valuable as well, and, thus, the next target. [Hondo et al., 2002] proposed a mechanism for the client to provide authentication data, based on the service definition, and for the service provider to retrieve those data. They also demonstrated how XML Digital Signatures and encryption can be exploited to achieve a level of trust. [Iha & Doi, 2009] proposed a binding mechanism, which is comparable to the binding mechanism for SQL. Furthermore, this paper shows the evaluation results of this mechanism by implementing this mechanism into the web browser (Firefox 3.0). [Ismail et al., 2004] proposed a client-side system that automatically detects XSS vulnerability by manipulating either request or server response. The system also shares the indication of vulnerability via a central repository. The purpose of the proposed system was twofold, to protect users from XSS attacks, and to warn the web servers with XSS vulnerabilities.[Jovanovic et al., 2006] presented a solution that provides complete automatic protection from XSRF attacks. The approach is based on server side proxy that detects and prevents XSRF attacks in a way that is transparent to users as well as to the web application itself. [Johns et al., 2008] advised a passive detection system to identify successful XSS attacks. Based on a prototypical implementation, we examine our approach's accuracy and verify its detection capabilities. The author compiled a data-set of 500.000 individual HTTP request/response-pairs from 95 popular web applications for this, in combination with both real word and manually crafted XSS-exploits; our 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.

The authors [Kieyzun et al., 2009] advised an automatic technique for creating inputs that expose SQLI and XSS vulnerabilities. The technique generates sample inputs, symbolically tracks taints through execution (including through database accesses), and mutates the inputs to produce concrete exploits. This technique creates real attack vectors, has few false positives, incurs no runtime overhead for the deployed application, works without requiring modification of application code, and handles dynamic programming-language constructs. The author also implemented the technique for PHP, in a tool Ardilla. [Louw & Venkatakrishnan, 2009] presented a XSS defense strategy designed to be effective in widely deployed existing web browsers, despite anomalous browser behavior. Their approach seeks to minimize trust placed on browsers for interpreting untrusted content. This approach was implemented in a tool called BLUEPRINT that was integrated with several popular web applications. The authors evaluated BLUEPRINT against a barrage of stress tests that demonstrate strong resistance to attacks, excellent compatibility with web browsers and reasonable performance overheads.

[Meir, 2006] shares a way to improve Web application security by integrating security throughout the life cycle. The ideas he presented here are based on empirical evidence from consulting with hundreds of customers - real-world scenarios with real project constraints and security concerns - across a variety of scenarios and putting into practice the security techniques that the experts know. The result is an approach that has evolved and refined itself over time. [Nichols & Peterson, 2007] had broken an applications life cycle into three main phases: design, deployment, and runtime. By organizing metrics according to life cycle in addition to OWASP type, insight from the derived quantitative results can potentially point to defective processes and even suggest strategies for improvement. [Schneier, 2007] focused on these problems—risk analyses, ROI models, audits—yet critical technologies that still remain uninstalled and important networks that remained insecure. The entire IT security industry is an accident: a piece of how the computer industry developed.

[Scott & Sharp, 2003] investigated new tools and techniques which address the problem of application-level Web security. They 1) described a scalable structuring mechanism facilitating the abstraction of security policies from large Web-applications developed in heterogeneous multiplatform environments; 2) presented a set of tools which assist programmers in developing secure applications which are resilient to a wide range of common attacks. [Shanmugam & Ponnavaikko, 2007] proposed signature based misuse detection approach. It expresses a security layer on top of the web application, so that the existing web application remain unchanged whenever a new threat is introduced that demands new security mechanisms. They claim that this approach is very effective as it addresses the vulnerabilities at a granular level of tags and attributes, in addition to addressing the XSS vulnerabilities.

The behavior based anomaly detection approach was proposed by [Shanmugam & Ponnavaikko, 2007]. They introduced a security layer on top of the web application, so that the existing web application remain unchanged whenever a new threat is introduced that demands new security mechanisms. Further application level parameters are also introduced to reduce the processing time. [Shanmugam & Ponnavaikko, 2007] illustrated a thread based solution for efficient process utilization of the web server and to prevent XSS threats. [Tiwari et al., 2008] introduced a client side solution that uses a step by step approach to detect XSS, without degrading much the user's web browsing experience. [Wei et al., 2010] addressed localization attacks against ITM systems in which an attacker impairs the effectiveness of an ITM system by identifying the locations of ITM monitors. They proposed an information-theoretic framework that models localization attacks as communication channels. Based on this model, they generalized all existing attacks as "temporal attacks," derived closed formulas of their performance, and proposed an effective attack detection approach. The information-theoretic model also inspires a new attack called a spatial attack and motivates the corresponding detection approach.

[Wurzinger et al., 2009] recommended SWAP (Secure Web Application Proxy), a server-side solution for detecting and preventing cross-site scripting attacks is introduced. SWAP comprises a reverse proxy that intercepts all HTML responses, as well as a modified Web browser which is utilized to detect script content. SWAP can be deployed transparently for the client, and requires only a simple automated transformation of the original Web application. Using SWAP, the author was able to correctly detect exploits on several authentic vulnerabilities in popular Web applications. [Xie & Yu, 2009] focused on the detection for new DDoS attacks, a scheme based on document popularity was introduced. An Access Matrix is defined to capture the spatial-temporal patterns of a normal flash crowd. Principal component analysis and independent component analysis are applied to abstract the multidimensional Access Matrix. A novel anomaly detector based on hidden semi-Markov model is proposed to describe the dynamics of Access Matrix and to detect the attacks. The entropy of document popularity fitting to the model is used to detect the potential application-layer DDoS attacks. Numerical results based on real Web traffic data are presented to demonstrate the effectiveness of the proposed method. In this [Zhang et al., 2010] recommended an execution-flow analysis for JavaScript programs running in a web browser to prevent Cross-site Scripting (XSS) attacks. The authors constructed a Finite State Automaton (FSA) to model the client-side behavior of Ajax applications under normal execution. Here the system is deployed in a proxy mode. The proxy analyzes the execution flow of client-side JavaScript before the requested web pages arrive at the browser to prevent potentially malicious scripts, which do not conform to the FSA. [Zhenyu et al., 2007] proposed a client-side system that automatically detects XSS vulnerability by manipulating either primitive detection mode or advanced detection mode. Through the modified model the system has concluded that the input invalidation is the most important aspect which causes XXS. The system also shares the vulnerability information via a central database.

3. EXISTING COUNTERMEASURES

3.1 Binding mechanism in the web browser
3.2 A multiagent scanner
3.3 Reverse proxy
3.4 Signature based model
3.5 Robust defense
3.6 Model based detection system
3.7 Serevr side solution
3.8 Client side solution
3.9 Automatic detecton/collection system
3.10 An execution flow based method

4. CONCLUSION

From the above paper we can depict that Cross-Site Scripting is extremely dangerous. It identifies the theft and act as impersonation. Cross-Site Scripting causes due to missing or insufficient input validation. We can prevent our web application by implementing XSS prevention in application, not assuming input values are benign. The server should not trust on client side validation rather it should check and validate all inputs before processing. In all application a conceptual solution should be used.

5. REFERENCES

[1] R.Bhatti, E.Bertino, A. Ghafoor, J.B.D Joshi, "Xml-Based Specification for Web Services Document Security," Computer , vol.37, no.4, pp. 41- 49, April 2004.

[2] Hung-Yu Chien, "Forgery Attacks On Digital Signature Schemes Without Using One-Way Hash and Message Redundancy," Communications Letters, IEEE , vol.10, no.5, pp. 324- 325, May 2006.

[3] M.Curphey, R. Arawo, "Web Application Security Assessment Tools," Security & Privacy, IEEE , vol.4, no.4, pp.32-41, July-Aug. 2006

[4] L. Desmet, P. Verbaeten, W. Joosen, F.Piessens, "Provable Protection against Web Application Vulnerabilities Related to Session Data Dependencies," Software Engineering, IEEE Transactions on , vol.34, no.1, pp.50-64, Jan.-Feb. 2008

[5] E.Galan, A. Alcaide, A. Orfila, J. Blasco, "A Multi-Agent Scanner To Detect Stored-XSS Vulnerabilities," *Internet Technology and Secured Transactions (ICITST), 2010 International Conference for* , vol., no., pp.1-6, 8-11 Nov. 2010.

[6] M.T Gebre,.; Kyung-Suk Lhee; ManPyo Hong; , "A Robust Defense Against Content-Sniffing XSS Attacks," *Digital Content, Multimedia Technology and its Applications (IDC), 2010 6th International Conference on* , vol., no., pp.315-320, 16-18 Aug. 2010.

[7] W.G.J Halfond, A.Orso, P. Manolios, "WASP: Protecting Web Applications Using Positive Tainting and Syntax-Aware Evaluation," Software Engineering, IEEE Transactions on , vol.34, no.1, pp.65-81, Jan.-Feb. 2008

[8] T. Holz, S. Marechal, F. Raynal, "New Threats and Attacks on the World Wide Web," Security & Privacy, IEEE , vol.4, no.2, pp.72-75, March-April 2006.

[9] M. Hondo, N. Nagaratnam, A. Nadalin, "Securing Web Services," IBM Systems Journal , vol.41, no.2, pp.228-241, 2002.

[10] G.Iha, H. Doi, "An Implementation of the Binding Mechanism in the Web Browser for Preventing XSS Attacks: Introducing the Bind-Value Headers," *Availability, Reliability and Security, 2009. ARES '09. International Conference on* , vol., no., pp.966-971, 16-19 March 2009.

[11] O.Ismail, M. Etoh,Y. Kadobayashi,S. Yamaguchi, "A Proposal and Implementation of Automatic Detection/Collection System for Cross-Site Scripting Vulnerability," *Advanced Information Networking and Applications, 2004. AINA 2004. 18th International Conference on* , vol.1, no., pp. 145- 151 Vol.1, 2004

[12] M.Johns, B.Engelmann, J.Posegga, "XSSDS: Server-Side Detection of Cross-Site Scripting Attacks," *Computer Security Applications Conference, 2008. ACSAC 2008. Annual* , vol., no., pp.335-344, 8-12 Dec. 2008.

[13]Nenad Jovanovic,; Engin Kirda,; Christopher Kruegel,;"Preventing Cross Site Request Forgery Attacks," *Securecomm and Workshops, 2006* , vol., no., pp.1-10, Aug. 28 2006-Sept. 2006.

[14]A.Kieyzun, P.J. Guo,K. Jayaraman, M.D. Ernst, "Automatic Creation of SQL Injection And Cross-Site Scripting Attacks," *Software Engineering, 2009. ICSE 2009. IEEE 31st International Conference on* , vol., no., pp.199-209, 16-24 May 2009.

[15]M.Ter Louw, V.N Venkatakrishnan, "Blueprint: Robust Prevention of Cross-Site Scriping Attacks for Existing Browsers," *Security and Privacy, 2009 30th IEEE Symposium on* , vol., no., pp.331-346, 17-20 May 2009

[16] Meier, J.D, "Web Application Security Engineering," Security & Privacy, IEEE, vol.4, no.4, pp.16-24, July-Aug. 2006

[17] Nichols, E.A.; Peterson, G.; , "A Metrics Framework to Drive Application Security Improvement," Security & Privacy, IEEE , vol.5, no.2, pp.88-91, March-April 2007.

[18] Schneier, B, "The Death of the Security Industry," Security & Privacy, IEEE , vol.5, no.6, pp.88, Nov.-Dec. 2007

[19] D. Scott, R. Sharp, "Specifying and enforcing application-level Web security policies," Knowledge and Data Engineering, IEEE Transactions on , vol.15, no.4, pp. 771- 783, July-Aug. 2003

[20] J .Shanmugam, M.Ponnavaikko, "XSS Application Worms: New Internet Infestation and Optimized Protective Measures," *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007. Eighth ACIS International Conference on* , vol.3, no., pp.1164-1169, July 30 2007-Aug. 1 2007.

[21] J.Shanmugam, M.Ponnavaikko, "Risk mitigation for cross site scripting attacks using signature based model on the server side," *Computer and Computational Sciences, 2007. IMSCCS 2007. Second International Multi-Symposiums on* , vol., no., pp.398-405, 13-15 Aug2007.

[22]J.Shanmugam, M.Ponnavaikko, "Behavior-Based Anomaly Detection on the Server Side to Reduce the Effectiveness of Cross Site Scripting Vulnerabilities," *Semantics, Knowledge and Grid, Third International Conference on* , vol., no., pp.350-353, 29-31 Oct. 2007

[23] S.Tiwari, R.Bansal, D.Bansal, "Optimized client side solution for cross site scripting," *Networks, 2008. ICON 2008. 16th IEEE International Conference on* , vol., no., pp.1-4, 12-14 Dec. 2008

[24] Wei Yu; Nan Zhang; Xinwen Fu; Bettati, R.; Wei Zhao, "Localization Attacks to Internet Threat Monitors: Modeling and Countermeasures," Computers, IEEE Transactions on , vol.59, no.12, pp.1655-1668, Dec. 2010

[25] P.Wurzinger, C.Platzer, C. Ludl, E. Kirda, C Kruegel, "SWAP: Mitigating XSS attacks using a reverse proxy," *Software Engineering for Secure Systems, 2009. SESS '09. ICSE Workshop on* , vol., no., pp.33-39, 19-19 May 2009.

[26] Yi Xie; Shun-Zheng Yu; , "Monitoring the Application-Layer DDoS Attacks for Popular Websites," Networking, IEEE/ACM Transactions on , vol.17, no.1, pp.15-25, Feb. 2009

[27] Qianjie Zhang, Hao Chen, Jianhua Sun, "An execution-flow based method for detecting Cross-site Scripting attacks," *Software Engineering and Data Mining (SEDM), 2010 2nd International Conference on* , vol., no., pp.160-165, 23-25 June 2010

[28] Qi Zhenyu, Xu Jing, Li Baoguo, Tan Fang, "MBDS: Model-based detection system for Cross Site Scripting," *Wireless, Mobile and Sensor Networks, 2007. (CCWMSN07). IET Conference on* , vol., no., pp.849-852, 12-14 Dec. 2007