International Journal of Scientific Engineering and Technology Research

ISSN 2319-8885
Vol.03,Issue.08,
May-2014,
Pages:1487-1490

www.semargroup.org,
www.ijsetr.com

# Ethical Hacking For Security from Web Based Attacks

PANKAJ CHANDRE[1], BAPURAO LAVAND[2], RAJRATNA SHINDE[3], PRAVIN PATANE[4], VISHAL SONAWANE[5]
Dept of Computer Engineering, Sharadchandra Pawar College of Engineering, Otur, Pune, India.

**Abstract:** Web applications computer programs allowing website visitors to submit and retrieve data to/from a database over the Internet using their preferred web browser. Web application provides flexibility, interoperability, availability, because of these reasons they are more prone to web-based attacks. In today's online world Web applications are not safe. It is of no use of doing online transfer of money, if it is unsafe? Web application security plays a very important role for such an activity. Do you believe that web applications are as safe as they are? With the growing use of Internet, web application security plays a important role. To ensure the trust of the user, it helps in outsourcing, helps to reduce the risks, reduce maintenance cost, helps to control the economy. Internet banking is the backbone of any developed or developing country. In this paper we are introducing SQL Injection attacks, Cross Site Scripting Attacks, Http Banner Disclosure and Broken Authentication & Session Management. Some questions that arise when we talk about the security of web application are: Is the application having secure login? Is the firewall smart enough to handle the threats? Is there any scope of easy hacking. And the answer is only one: Web Application Security. In this paper we have define new vulnerabilities in web applications and help the developer to overcome these weaknesses.

**Keywords:** Web Application Security, Cross-site Scripting Attacks, SQL Injection, Broken Authentication.

.

## I. INTRODUCTION

Web applications do raise a number of security concerns serious vulnerabilities, allow hackers to gain direct and access to databases in order to access sensitive data. According to surveys, security issues in web applications are the most commonly reported vulnerabilities on the Internet. Web application are also vulnerable to new security threats such as SQL injection Attacks, Cross-site Scripting Attacks, Broken Authentication & Session management. To run the web application more securely, the above mentioned designing steps are very important. From client side to server side scripting and from web application to back-end (database). Secure web application is the solution to handle the threats. We are using the web crawler for scanning the web application pages. Crawling is the term used to describe the action taken by a program as it browses each hyperlink on a website. The crawler will fetch a starting page and parse the hyperlinks, crawling to those pages. Until the process is completed and there are no more links to crawl this process will continue repeatedly. Crawling makes scanning easier of a web application. It checks all the hyperlinks that the scanner is aware of all linked pages that exist on the website. Crawlers should be as configurable as possible, allowing the user to define criteria to ensure a thorough and efficient crawl.

## II. LITERATURE SURVEY

Web applications have increasingly become high-value targets for hackers. Since so many Web sites contain vulnerabilities, hackers can leverage a relatively simple exploit to gain access to a wealth of sensitive information, such as credit card data, social security numbers and health records. Therefore, it's more important than ever to examine your Web application security, assess your vulnerability and take action to protect your business. Table 1 lists just a few of the potential threats to Web applications, the effect they could have on your business, and the average percentage of Web sites that are vulnerable to this type of attack.

**Table1. List of attacks on Web applications.**

| Threat | What attackers can do? | Average % of vulnerable Web applications |
|---|---|---|
| Cross-site scripting | … impersonate a trusted user to gain access to your sensitive business data | 80% |
| SQL Injection | … access all the data in your database, resulting in a complete data compromise | 62% |
| Broken Authentication | … steal one or more of your customers' identities | 60% |

It involves overall facilities provided by the application to overcome the disadvantages of existing system. Web applications interface with databases that contain information

such as user names, credit card numbers, online orders, and so on. Web browsers build SQL queries to access these databases based, in part, on user-provided input. The intent is that Web applications will limit the kinds of queries that can be generated to a safe subset of all legitimate queries, against the input users provide. However, incomplete input checking can enable attackers to gain complete access to such databases. Attackers can submit input strings that contain specially encoded database commands. When the dynamic building of a query by using these strings and submits the query to its database, the inserted commands are executed by the database and the attack succeeds.

The results of these attacks are often disastrous and can range from leaking of sensitive data to the destruction of database contents. Since late 90's, attackers have managed to continue exploiting XSS attacks across Internet web applications although they were protected by traditional network security mechanisms, like firewalls and cryptography based techniques. The use of specific secure development techniques can help to solve the problem. But, not as expected. For instance, the use of secure coding practices and/or secure programming models are often limited to traditional applications, and not as required for the web Application. Many systems allow the use of weak passwords. Broken Authentication session management evolves automated process of trial and error, Guessing a person user ID and password, credit card numbers, system sends a value and waits for the response, then tries another value, and so on.

## III. EXISTING SYSTEM

AMNESIA(Analysis and Monitoring for Neutralizing SQL-Injection Attacks) previously developed by combining static analysis and runtime monitoring to detect SQL Injection Attacks. A static analysis approach is used to build models of the different types of queries that an application can generate and dynamic analysis to intercept and check the query strings generated at runtime. Queries that are different than the generated one in the model are identified as SQLIAs. Dependency on the precision and efficiency of its underlying static analysis is the real problem of this approach, which may not scale to large applications. We apply purely dynamic approach to preventing SQLIAs, thereby eliminating scalability and precision problems.

**Pixy:** A Static Analysis Tool for Detecting Web Application Vulnerabilities. This approach scans the application and leverage information flow analysis or heuristics to detect code that could be vulnerable to SQLIAs. Because of the inherently imprecise nature of the static analysis that they use, these techniques can generate false positives. Moreover, since they rely on declassification rules to transform untrusted input into safe input, they can also generate false negatives.

The problem with the current JavaScript security mechanisms is that scripts may be confined by the sand-boxing mechanisms, but still breaks the security of a system. This can be achieved when a user is lured into downloading malicious JavaScript code (previously created by an attacker) from a trusted web site. Authentication and session management includes all aspects of handling user authentication and managing sessions. Authentication has main emphasis in this process, but even authentication techniques can be undermined by flawed password management with account update, and similar functionalities should require re-authentication even if the user has a valid session id.

## IV. PROPOSED SYSTEM

In our proposed System first we scan the web application completely for weaknesses existed in that. We are using web Crawler in our system to fetch all the links in respected application.

### A. SQL Injection Attack:

When the Web application builds a query by using strings and submits the query to its underlying database, the attacker's inserts his own commands that are executed by the database and the attack takes place. The results of these attacks are often dangerous and there are more chances of leaking of sensitive data. SQL injection is a type of security exploit in which the attacker adds Structured Query Language (SQL) code to a Web form input box to gain access to resources or make changes to data. An SQL query is used to perform some action on a database. Typically, on a Web Login forms for user authentication, when a user enters their Username and password into the text boxes over there, those values are then inserted into a SELECT query. If the values entered are found as expected, the user gets authorized access; if they didn't match, then access is denied.

In our approach we are going to propose a system for detection of SQLIA, it works on identification of "trusted" strings and allowing only trusted strings to be used to create the semantically relevant parts of a SQL Query. For building SQL queries, strings are constantly broken into substrings, manipulated, and combined. We check the queries for special characters in given user input. Also we check the syntax of the generated query. Ex. if a database contains Usernames and passwords, the application login form may contain code such as the following: Query="SELECT * FROM accounts WHERE name = ‟ " + request.getParameter ("name")+ " " ‟ AND password=‟ +request.getParameter("pass") + " ‟ ";
Previous code generates a query to authenticate user who tries to login to a web application. On the other hand malicious users enter "doe" into the name field and" „OR „a‟‟=‟‟a‟‟ into the password field, the query string becomes: SELECT * FROM accounts WHERE name=‟‟doe‟‟ AND password=‟‟ ‟‟ OR „a‟‟=‟‟a‟‟ which condition is always true, and the attacker may gain access it will bypass the authentication logic.

### B. Cross-site Scripting Attacks:

Cross-Site Scripting attacks (XSS attacks for short) are those attacks against web applications in which an attacker gets control of a user's browser in order to execute a

malicious script (usually an HTML/JavaScript code) within the context of trust of that website. As a result, and if the inserted code is successfully executed, then attacker might be able to access, to any sensitive browser resource associated to the web application (e.g., cookies, session IDs, etc.). The "cross-site scripting" originally referred to the act of loading the attack from third-party on web application from an unrelated attack site. JavaScript prepared by the attacker in the security context of the targeted domain.

The definition also consists other modes of code injection, which includes persistent and non-JavaScript vectors (including ActiveX, VBScript, or even HTML scripts), causing some confusion to newcomers. XSS vulnerabilities have been reported and exploited since the 1990s. Most famous social-networking sites like as Twitter, Facebook, MySpace, YouTube and Orkut. In recent years, cross-site scripting attacks become the most common publicly reported security vulnerability, with some researchers in 2009 viewing as many as 72% of websites as likely open to XSS attacks. The attacker has now stolen the session and can login as the dummy user. We check web application for XSS to find several vulnerabilities. The most website security issues, the main area to exploiting these vulnerabilities is in user supplied data. Beginning with first mapping the application, and making note of every place a user might control information and related data. GET and POST parameters are often good to test, though developers implement filtering defenses in these areas.

**1. Persistent Cross-Site Scripting**

Persistent XSS is more dangerous than reflective XSS. In this attack the malicious script is embedded permanently into the web application. The scripts will executed when the victim people access the page it is located on.

Here is an attack using Stored Cross-Site Scripting:

1. The victim visits a website they trust, flipkart.com.
2. A script has been inserted by an attacker on a page they happen to visit while on flipkart.com.
3. The script executes in the context of flipkart.com.
4. The attack takes place and victim is then compromised.

**2. Reflective Cross-Site Scripting**

In this attack, we take help of social engineering to make the attack successful. Here is an attack, using Reflective Cross-Site Scripting:

1. The victim gets an email/Instant Message that contains a link.
2. The victim clicks the link. (Requires User Intervention)
3. A script has been inserted by an attacker on the page they then visit.
4. The script executes in the context of that site.
5. The victim is then compromised.

**3. Broken Authentication and Session Management:**

Broken authentication and session management attacks are those attacks against web applications in which an attacker gets control of a user's browser in order to execute malicious script (usually an HTML) within the context of trust of the web application's site. As a result, and if the inserted code is executed successfully on victim's side then the attacker might able to access, passively or actively, any sensitive browser resource associated to the web application (e.g. cookies, session IDs etc.). Examples include forgotten password, depending on IP address for session, emailing user credentials, not authenticating a user before password is changed, and not having limited timeouts for inactive sessions. Forgotten Password functionality is probably the easiest to understand. Web applications often have a forgotten password functionality that allows a user to submit their user name to the application and are taken to a page with secret questions or a temporary password reset function. Attackers can exploit this functionality to enumerate valid user name for the application.

Types of broken authentication and session management attacks:

1. Brute Force Attack
2. Session Spotting
3. Replay Attack
4. Session Fixation Attack
5. Session Hijacking
6. Insufficient Session Expiration.

## V. CONCLUSION

This paper has presented an Ethical Hacking for Security from Web Based Attacks. In this paper important threats are described which intrudes the web application security. We should keep different type of the threats and their solution in mind before developing the web application. From this paper we research on several web threats detection and prevention techniques for them. Our research is helpful for development of better and secures web applications. We examine client's existing security practices and make recommendations that can help them protect there enterprise from the consequences of a security breach.

## VI. REFERENCES

[1] Ammar Alazab, Moutaz Alazab, Jemal Abawajy, Michael Hobbs, "Web Application Protection against SQL Injection Attack" The 7th International Conference On Information Technology and Applications(ICITA 2011). ISBN: 978-0-9803267-4-1.

[2] N. Jovanovic, C. Kruegel, and E. Krida, "Pixy: A Static Analysis Tool for Detecting Web Application Vulnerabilities," Proc. IEEE Symp. Security and Privacy May 2006.

[3] W.G. Halfond and A.Orso, "AMNESIA: Analysis and Monitoring for Neutralizing SQLIA," Proc.20th IEEE and ACM Int"l Conf. Nov 2005.

[4] Zhendong Su, Gary Wasserman , "The Essence of Command Injection Atacks in Web Applications" ACM 1-59593-02702/06/0001.

[5] D. Scott and R. Sharp, "Abstracting Application Level Web Security" In Proc. Of the 11th International Conf. on WWW 2009.

[6] Y. Huang, S. Huang, T. Lin and C. Tsai, "Web Application Security Assessment by Fault Injection and Behavior Monitoring. In Proceedings of the 11th Int"l WWW 2003.

[7] E. Krida, C. Kruegel, G. Vigna and N. Jovanovic, Noxes : A Client-Side Solution For Mitigating Cross-site Scripting Attacks. In the 21st ACM Symp. On Applied Computing. (SAC 2006).

[8] D. Parsons, "Dynamic Web Application Development Using XML and Java : Cengage Learning, EMEA,2008.

[9] C.Torrano-Gimenez, A.Perez-Villegas and G. Alvarez "WASAT-A New Web Authorization Security Analysis Tool," Web Application Security, May 2010.

[10] David Endler,"The Evolution of Cross Site Scripting Attacks," Technical Report, iDEFENSE Labs, 2011.

[11] Huluka, D. and Popov, O. 2012. Root Cause Analysis of Session Management and Broken Authentication Vulnerabilities. In proc. Of IEEE Conf. on World Congress On Internet Security.

[12] Lieven Desmet, Pierre Verbaeten, Wouter Joosen, and Frank Piessens.2008.Provable Protection against Web Application Vulnerabilities Related to Session Data Dependencies. IEEE Transactions on Software Engineering.

[13] Nish V. K. and Layamon Aliyar. 2010.An Overview of Cryptographic Solutions to Web Security. Anna University of Technology Coimbatore, India. In Proc. Of IEEE Conference.

[14] Matbouli, H. and Gao, Q. 2012.An Overview on Web Security threats and impact to E-Commerce Success. In Proc. of IEEE Conf. on Information Technology and e-Services.

[15] Atashzar, H.Torkaman, A.Bahrololum, M. and Tadayon, M.H. 2011. A Survey on Web Application Vulnerabilities and Countermeasures. In Proceedings of IEEE conference on Computer Sciences and Convergece, Information Technology.