ISSN:0975-9646

A Survey on Security and Vulnerabilities of Web Application

Gopal R. Chaudhari, Prof. Madhav V. Vaidya

Department of Information Technology,
SGGS IE & T, Nanded,
Maharashtra, India-431606

***Abstract*— Web applications are the distributed platform used for information sharing and services over Internet today. They are increasingly used for the financial, government, healthcare and many critical services. Modern web applications frequently implements the complex structure requires for user to perform actions in given order. The popularity adds value to these applications, which attracts attackers towards them. The attackers are well known about the valuable information accessible through the web application, which leads to serious security attacks on web applications.**

In this paper we survey the state of the art in web application security; first we explain working of web application and focus on the challenges for building secure web application. We organized the existing security vulnerabilities into the security properties that web application should preserved, discussed the root cause of these vulnerabilities and their corresponding preventive measures. Next we focus on the malware attacks on the web application, how the web applications are compromises for security and get infected by malware. Finally we summarize the lessons and discussed the future scope and opportunities in this area.

***Keywords*— Security vulnerability, Web Application, separated by comma.**

I. INTRODUCTION

Although traditional firewalls have effectively prevented network-level attacks, most future attacks will be at the application level, where current security mechanisms are woefully inadequate [1]. The application level security inherent in the web application's code, regardless of the technology used for web application development or the security of the web server or the database on which it is built. But the vulnerabilities are still present in the application as the firewall or the intrusion detection systems keep open the port 80 and 443 for online business purpose. The web application provides client access to the end user to the server functionality through the web pages. These web pages contain the HTML, images, script code, etc. as become more users friendly but also exploits the security vulnerabilities.

The web applications like financial applications, healthcare application, government websites, etc. are interact with the backend database many times for the client's request response. If such web applications are compromised for the security will result in financial, informational, ethical, legal consequences issues for the web application.

The security of the web application is most important, according to the report by Web Application Security Consortium, about 49% web application contains the high severity level vulnerabilities and 13% of them are automatically get compromised for the security vulnerabilities. The unsecure web application leads to the known security vulnerabilities such as Injection, Cross-side scripting, cookie theft, security misconfiguration, session hijacking, self-propagating worm's attacks, etc.

The substantial amount of research had been devoted into hardening and mitigating the different vulnerabilities of web application. Many of these solutions wares based on some assumption on the web technologies used in application development and mitigate the concern security flaws. The probability that these solutions can be applied to other similar type of issue may be very less due to narrow mind approach or practitioner may focus on providing an exact solution to particular technology related vulnerability only.

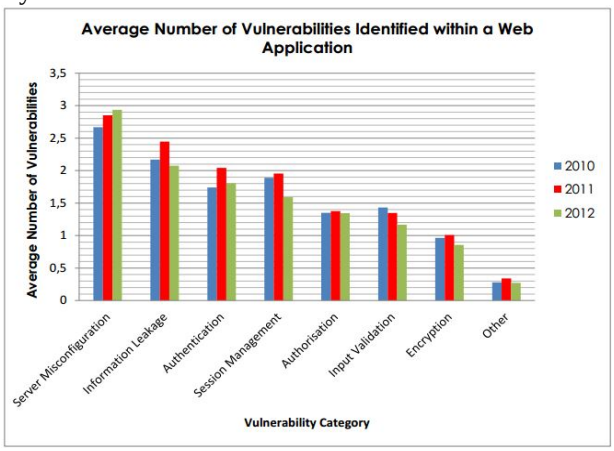

Fig. 1 Average number of vulnerabilities within web application

In this paper we survey the state of the art in web application security, with aim of having a systematic categorization of existing vulnerabilities under broad categories based upon security properties that web application should preserve, discuss the roots of vulnerability with preventive measures, and focused on malware web application threats. The figure 1 shows the percentage of the web applications affected by the different vulnerabilities; some of those also present in OWASP Top 10. The OWASP Top 10 focuses on the identifying the most serious vulnerabilities for the broad array of the organizations [2].

II. How A Web Application Works

Web application is a distributed application which enables the dynamic information and service delivery. As shown in figure 2, web application consists of client side and server side components. The client side component which includes static web pages with embedded scripting languages e.g. JavaScript executed within browser. Client make http request to the web server by specifying particular URL via internet.

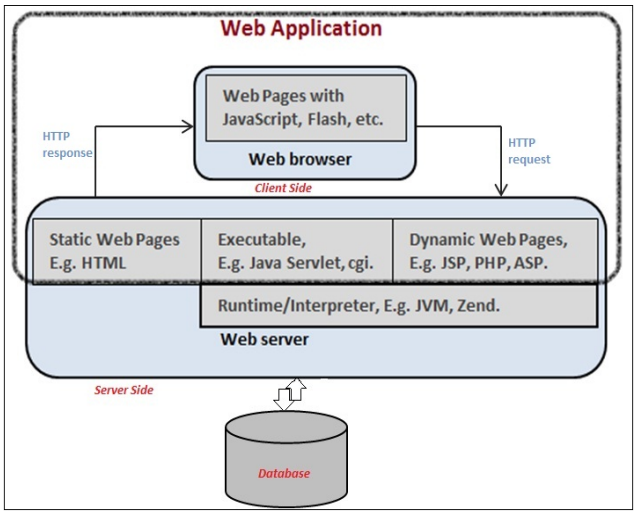

Fig. 2 Web application overview

At server side, client's request is processed within web server using dynamic HTML pages either through execution (CGI, or Java Servlets) or interpretation (PHP, JSP) and provides appropriate response to the client request. These servers actually contain the business logic of the application. Business logic refers to the algorithm implementation that to be performed to the data such as how to create, display, stored and change the data. Client can communicate with server side code using asynchronous call such as AJAX and dynamically updates the HTML pages. The web applications interact with the backend file system or database server for storing and retrieving data. Aspects of web application including programming language, state maintenance and logic implementation differentiate the web application from traditional applications.

III. Web Application Security Properties and Vulnerabilities

A secure web application should have the security properties as shown in following figure 3. The logic correctness means the application logic should be exactly corrected as intended by the developers; input validity refers to the user input validated before application use it; state integrity means the application state should be kept untampered and security misconfiguration refers to the configuration settings, using secure components, etc. [3].

A. Logic Correctness

The logic correctness ensures that web application functions correctly according to its business logic. Every web application has its own business application logic; it is difficult to covers all the aspects in single description. But at global level logic corrects property refer to "Users can access only the authorized information and operations and follows the intended workflow provided by the web application". The web application is implemented as number of independent modules which are accessed by user in any order. The control flow across these different modules performed through tight collaboration of two following approaches. First approach is interface hiding, where only the accessible resources and actions of application are presented to the user by web links. In second the explicit check on application state is done, before any sensitive information application can accessed. Vulnerabilities with logic correctness are known as logic attacks or state violation attacks.

1) Input validation and sanitization*:* These techniques ensure that the correct web application behaviour. Basically input validation means the user inputs to the web application should be validated before it can be utilized by web application. For any web application which accepts the untrusted data should incorporate the input validation procedures, to ensure that the computing values are legitimate and sensible.

In context of the web application the input validation should be applied on the client side inputs which are further processed on the server side. The web application accepts the inputs through different means such as HTTP request query strings, POST method bodies, database queries, HTML5 postMessages invocations.

Consider the POST request shown in figure 4. The request contains the several parameters including seesionID, CSRF token, multiple parameters such as mobile number, credit card number, and date. Each of these parameter requires the different input validation such as credit card number requires certain number of characters and Luhn check, mobile number also requires certain number of integer digits, date should be the current date in specified format e.g. dd/mm/yyyy, etc. The sanitization preferably output sanitization used throughout the web application lifecycle. The output sanitizer typically escapes the special characters from the untrusted input sources such as '<', '&' should not appear in un-escaped values to be incorporated.

Here we illustrate the most popular input validation attacks includes SQL injection, Cross Side Scripting (XSS).

i) SQL Injection: It is a code injection technique, allows attacker to retrieve crucial information from the web server's database [4]. The attacker is successful to launch this attack due to user inputs flow into the SQL queries without correct validation. Using SQL keywords along with user inputs attacker can get SQL query result manipulated as unintended execution. SQL injection can result in authentication bypass, data loss, denial of access, and also it may leads to destruction of whole database or the host takeover.

Severity: Moderate to High

Preventive measures: Some preventive measures should be considered to reduce the probability that the injection attack successful such as avoid connecting to the database as a super user access, avoid the use of dynamic SQL queries,

validate/sanitize the input data before it can utilized by web application, use encrypted or hash format to store sensitive data, don't reveal much informative in error messages; instead of that use custom error messages to display the minimal information [5].

ii) Cross Side Scripting (XSS): XSS flows occur when an application processes the user supplied data from web pages without proper validation or escaping that content. XSS enables attacker to inject client-side script into web pages that exploit the interpreter in the browser. The consequence of XSS can be session hijacking, sensitive information disclosure, may bypass the perimeter defences and site trusting. There are three known types of XSS flows based upon the how malicious script are injected, including stored XSS (malicious script are injected to the persistent storage media), reflected XSS, and DOM –based XSS, etc.
Severity: Moderate to High

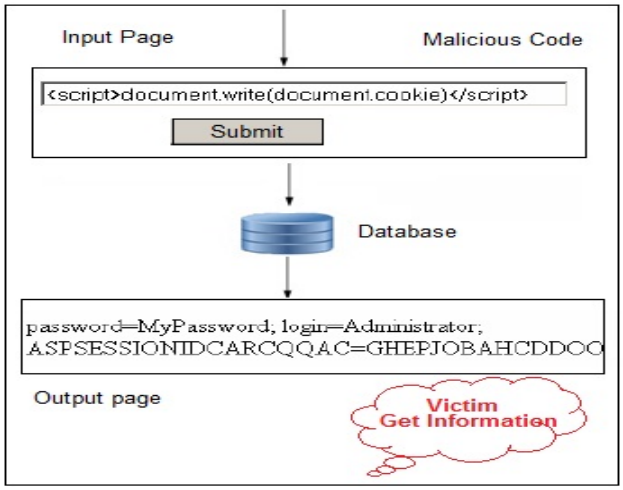

Fig. 4 XSS vulnerability

Figure 4 shows how the XSS attack is carried out in web application. The attacker can insert a message, containing script code through input fields. This script is stored into the database, when attacker opens the message page which results in information discloser. As there is no security checks performed while sending the script to the browser as a message, this exploits to run any arbitrary script in browser.
Preventive measures: Introduce the input validation function after every input statement in web pages. Use sanitization on untrusted data, identify the unsafe characters, escapes special characters or validate them against different parameters such as length, format, business rules, etc. Use a character encoding technique (e.g. UTF-8) and configure all components to use it.

2) State Integrity: In a typical web application, the user's web browser interacts with the remote application by sending HTTP requests. The HTTP is a stateless protocol without session mechanism [6]. This means the each request is independent to the next. Dynamic web applications have workflow that composed of multiple steps, which corresponds to the multiple HTTP requests to the application. Here the actions are defined by the URI [7] of the HTTP request, the request parameters, and the server side session record. State maintenance is the basic for building the state based web application. Attacker targets the vulnerabilities within session management and state maintenance including session fixation, session hijacking, cookie poisoning, and Cross-side request forgery. Number of techniques are proposed to preserve the state integrity [8] such as client-side information can be protected by Message Authentication Code (MAC), session identifier are generated with high randomness and transmitted over the SSL protocol.

i) Broken Authentication and Session Management: In application the authentication and session management functions are often not implemented correctly. There are key points to be considered such as logout functionality, password management, timeouts, secret question, account update. Attacker uses the flows in authentication and session management functions (e.g. exposed accounts, password, session IDs, etc.). Using the loopholes from authentication and session management functions the attacker can steals the sensitive information including user credentials, session IDs, etc.
Severity: High
Preventive measures: Implement strong functions for authentication and management, proper functions for request time out, user account log in-out policy, session expiration, allow secure and http cookies only, etc., sensitive information traversals in encrypted format only, avoiding session IDs in URLs, and follow security standards which specifies the requirement for secure authentication and session management.

ii) Cross-side Request Forgery: This is a type of exploitation where attacker tricks victim into sending crafted web requests via image tag, XSS, or by means of other techniques with the victim's valid session identifier, however, on the attacker's behalf. Since the browser sends the sensitive information such as session cookies automatically, the victim's session being tempered, result in loss of sensitive information, also attacker can make forge requests that are undistinguishable from legitimate ones.
Severity: Moderate
Preventive measures: CSRF can be prevented by use of unpredictable token in each HTTP request. These tokens are unique with per user and also unique with per request. Including these unique token as a hidden field, cause the value to be sent in the body HTTP request, and avoid its inclusion into the URL, which is subject to exposure. These unique tokens can also be included in the URL itself, however such placement runs the risk that it will expose to an attacker, thus it can compromise the secret token.

3) ***Security Configuration:*** The web application is a complex ecosystem composed of a large number of hardware/software components including web server, database server, request handling queues, and many more. This type of vulnerabilities focus on proper configuration of components, access control mechanism among them, stays component up to date, etc. The misconfiguration or unauthorized accesses leads to the vulnerability where web

application show false behaviour or anonymous user could get the access to the web application infrastructure.

i) Security Misconfiguration: Improper security configuration leads to this vulnerability. This vulnerability can happens at any level of the application stack, including components such as application server, web server, database server and platform. This flow allows an attacker to gain the unauthorized access to the system data or functionality. It may result in complete system compromised.
Severity: Moderate
Preventive measures: Good security for web application requires properly defined, implemented and maintained secure configuration for the all components including platform, framework, and server components, etc. Run scans and audits periodically to detect the future misconfigurations and missing patches. A repeatable hardening process also makes it fast and easy to deploy another environment that is properly locked down.

ii) Using Components with known Vulnerabilities: Web application composed of many vulnerable components such as external libraries, frameworks, and other software which always run with full privilege. Many times the application developer doesn't know about these components version and their dependencies which may cause things worse. Exploiting such vulnerability can cause serious data loss or may server takeover.
Severity: Moderate to High
Preventive measures: It is better to not use the components that you won't write. But also there are open source projects that do not create vulnerability patches for older vulnerable version; they simply mitigate the problem next versions. Software projects include some process to identify the legitimate components, their versions along with all dependencies, monitor the security of these components and keep them up-to-date.

The figure 4 shows the end-to-end working of the web application and the vulnerabilities as explained above possible at different stages of the web application.

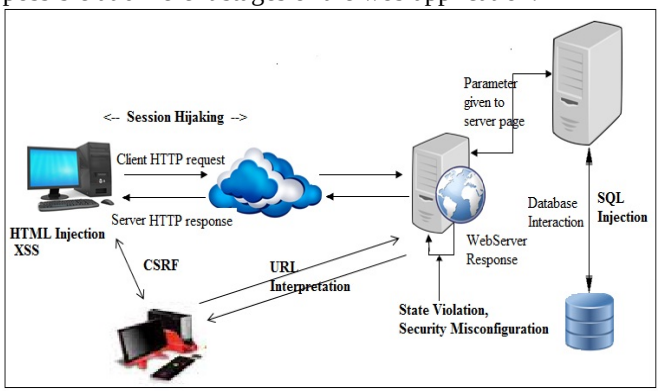

Fig. 4 Vulnerability present at different stages of web application

IV. WEB APPLICATION - MALWARE THREATS

Malware attacks are the most discussed area of the web application security. In recent years many attacks are happens on the banking websites, organization's websites including Google's blacklisting. Typically malware attacks compromises the unpatched browser, operating system's services, and popular applications such as ActiveX, Microsoft Office, outlook messenger, FTP, security related processes such as antivirus program, security tools, etc.
Websites are updated constantly and with each update the malware find a new means of infection. At initial stage the malware attacker targets the web server by targeting the vulnerabilities with it, later on malware attacks on the data and infrastructure related components of the web application, and latest malware attacks shows that malware attacks on the end user computer and tries to steals the user's sensitive information. Figure 5 shows the evolution of malware how they attacks on web application [9].

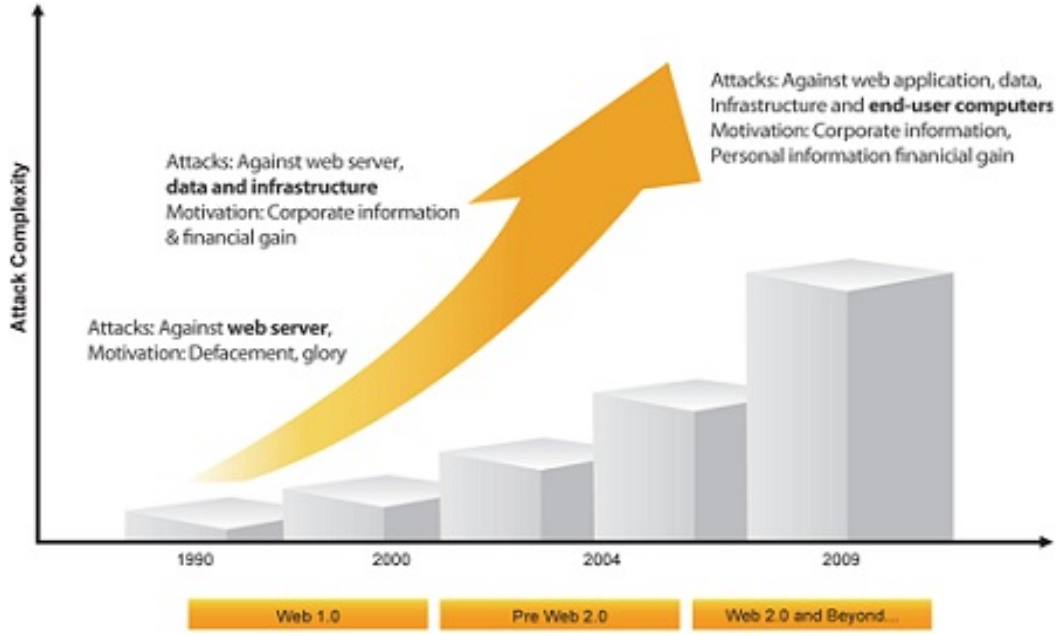

Fig. 5 Malware evolution

A. *Website vulnerabilities to Malware*

The website owner continuously looks for improving the customer experience, increasing the popularity of the website, supporting the mobile devices, social networking websites, user customization, etc. While making advances in website increases the risk of malware hosting.

Google had about 350,000 index of malware hosting websites [10]; malware distribution via website in 2010 was almost double with 286 million of malware variants identified in 2011. The interactivity on the websites including e.g. social networking site – user comment field, file upload, file download are also exploits vulnerability. Technically these vulnerabilities are with improper input validation, logging mechanism, and the fail-open error or fail to close a database connection. OS commands, LDAP, SQL, XPath queries are also vulnerable to the injection vulnerabilities.

Mobile devices are other targets for malware as mobile applications are wildly used by large amount of users [11]. The social networking sites are the malware transmission channels where user unknowingly share and spreads the links to the malware infected websites. Even with deep sense of security strategy, it is very difficult to website malware infection if the user interact malicious content or activity or utilize the cloud based services.

V. CONCLUSIONS

Web applications are becomes popular and have wide spread interaction medium in our daily lives. But at same point using vulnerabilities the user sensitive data also disclosed regularly. This paper surveys the different web application vulnerabilities based on the security properties that web application should preserved. However we enforce to have a pen test, vulnerability assessment of the web application for discussed vulnerabilities which reduces chances of the occurrence of the vulnerabilities. However vulnerability assessment tools are automated one which saves time and money and also defend the web applications from modern threats.

At the last the new advanced security attacks are always emerging, requires the security professional to have positive security solution without putting huge number of web applications at risk.

REFERENCES

[1] J. Pescatore, Web Services: Application-Level Firewalls Required, report no. SPA-15-5542, Gartner, Stamford, Conn, 7 Mar. 2002; available at www4.garter.com/DisplayDocument?id=353429.

[2] OWASP Top 10 Web Application Vulnerabilities, https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

[3] Xiaowei Li and Yuan Xue, "A Survey on Web Application Security", Technical report, Vanderbilt University, 2011.

[4] Z. Su and G. Wassermann. The essence of command injection attacks in Web applications. In Proc. POPL, 2006.

[5] Mohit Kumar, Abhishek Gupta, Azhar Shadab, Lokesh Kumar & Vikas Kumar Tiwari, Defending Against Modern Threats in Web Applications, International Journal of Computer Science and Informatics ISSN (PRINT): 2231 –5292, Vol-1, Iss-4, 2012.

[6] Fielding,R., Gettys, J., Migul, J., Freystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: Hypertext Transfer Protocol {HTTP/1.1.RFC 2616, http://www/w3/prg/Protocols/rfc2616/rfc2616.html (June 1999).

[7] Berners-Lee, T., Fielding, R., Irvine, U., Masinter, L.: Uniform Resource Identi_ers (URI): Generic Syntax. RFC 2396, http://www.ietf.org/rfc/rfc2396.txt (August 1998).

[8] A. Barth, J. Caballero, and D. Song, "Secure content sniffing for web browsers, or how to stop papers from reviewing themselves," in

[9] Oakland'09: Proceedings of the 30th IEEE Symposium on Security and Privacy, 2009, pp. 360–371.

[10] Malware Info Resource Center, http://www.malware-info.com/mal_faq_inject.html

[11] Google Security Blog, Http://googleonlinesecurity.blogspot.com/2009/08/malware-statistics-update.html

[12] http://www.forbes.com/sites/andygreenberg/2011/08/05/android-app-turns-smartphones-into-mobile-hacking-machines/