A Study on Vulnerabilities and Their Impact in Web Applications

G. Nirmala, Padmaja Kanchi
Dept. of CSE, Sir C.R.R College of Engineering, Eluru, AP, India
Dept. of CSE, Dr. BRAGMR Polytechnic, Rajahmundry, AP, India

Abstract
Design flaws in web application security measures are constantly being researched, both by hackers and by security professionals. Most of these flaws or defects affect all dynamic web applications which may lead to different vulnerabilities. Several approaches are coming into market to prevent these applications vulnerabilities. Web applications can be secured by predicting and discovering flaws in designing process of a application. This paper focus on study and analysis of web application vulnerabilities and how they affect web applications.

Keywords
Webapplication, Vulnerabilty, Security, Dynamic Testing, WAF

I. Introduction
In this section we discussed about web Application, Structure and development process

A. Web Application
In computing Environment, a web-based application [1] is any application that uses a web browser as a client. Web Application or Web app is defined as an application program that is stored on a remote server and delivered over the Internet through a browser interface. From the last two decades web applications are growing day by day form static to dynamic applications. According to a definition given by the Open Web Application Security Project (OWASP)[2], a web application is a software application that interacts with customers or users or other systems using the networking protocol Hyper Text Transfer Protocol (HTTP) which is used for data communication for World Wide Web.

Fig. 1: Web Application Structure

II. Web Applications Lifecycle Model
Web Application Lifecycle[3] is the process of developing a web application and involvement of the multiple teams that are engaged in the development process. Each organization may set forth its own unique style of operating. Some companies follow a certain standard model such as SDLC (System Development Life Cycle) or Agile Software Development Model [12, 14]

As stated in [3], SDLC is the traditional process of developing software or web applications by including research to identify and define the application requirements, information analysis, architectural design and specifications blueprint, team involvement, programming, testing and bug fixing, system testing, implementation and maintenance.

Agile Software / Web Application Development is the iterative development process and development process practices that focus on collaboration of people involved and provide a better procedure to allow revisions and evolution of web application requirements. Agile methodology includes research, analysis, project management, design, programming, implementation, frequent testing, adaptation and maintenance.

A. Web Application Development Process
Web Application Development Process [4] organizes a practical procedure and approach in application development. Testing is an important part of the Web Application Development process. On occasion, testing would consume more manpower and time than development itself. Below are some of the most common testing [15] needed for any web application development process:
- Quality Assurance and Bug Testing
- Multiple Browser Compatibility
- Application Security
- Performance - Load and Stress Testing
- Usability

Open Web Application Security Project (OWASP) has released top ten web application vulnerabilities in 2011. They are SQL Injection, Cross-Site Scripting, broken authentication and session management, invalidated redirects and forwards, failure to restrict URL’s, security misconfiguration, insufficient transport layer protection, insecure cryptographic storage, insecure direct object references.

Web application vulnerabilities are affecting enterprises in horrific way. Every day a bad news is scrolling in media that a major retailer’s website is hacked, and thousands of customer records, including credit card numbers, are stolen. A small flaw in design of web application leads to big trouble. Not only is application attacks growing more prevalent, they are also expensive. The research firm Gartner, states that 80% enterprises will suffer from these vulnerabilities.

III. Web Application Vulnerabilities

A. Vulnerabilities and Their Effects
In order to emphasize the meaning of the results we classify the vulnerabilities by effect on the system. We use four categories which can be stated in simple terms that apply not only to information security experts, but to end users as well.
- Execution of arbitrary code on the server
- Unauthorized arbitrary information retrieval (includes private information theft)
- Direct financial damage
- Denial of Service
Each class may be the consequence of a wide variety of attacks[9]

1. Cross Site-Scripting
An attack aimed at pushing a script tag into a server that would be sent from the server to an innocent user browsing the Web server thus causing the script to be activated in the innocent user’s browser.

(i). SQL Injection
An attack that manipulates input data sent to the server, causing it to run a SQL-generated input that would pull data or change the contents of its internal data.

(ii). Parameter Tampering
Changing the data within a parameter sent from one Web page to another in a way that would alter the behavior of the latter page.

(iii). Known Vulnerabilities
Using known vulnerabilities and exploits on commercial software platforms. This class holds dozens of attacks that are widely known and published.

(iv). Cookie Poisoning
Changing the contents of cookie saved in the client’s computer in such a way that it would change the normal flow of the application.

(v). Access to Administration Area and Internal Modules
Allowing unauthorized access to administrative areas or other internal modules of an application.

(vi). Directory Traversal
Allowing access to unauthorized server directories.

(vii). Improper Management of Permissions
Improper management of the server’s permissions allowing a non-privileged user to access some modules that weren’t originally intended to be seen by that user.

2. Buffer Overflow
Data sent as input to the server that overflows the boundaries of the input area, thus causing the server to misbehave. Buffer overflows can be used to make the server run a code sent into the overflowed buffer.

(i). Forceful Browsing
The ability of an attacker to directly access unauthorized web pages by bypassing the logical flow of the application, possibly avoiding authentication requirements and credentials checking.

(ii). Denial of Service
Causong the site to malfunction due to some sort of denial of the service it is offering by means of bandwidth consumption, site defacement.

(iii). Session Hijacking
Capturing the session of another user, which in effect means being able to impersonate the user in the eyes of the application.

(iv). Brute Force
Attacks designed to steal of passwords or session ids, by means of enumerating a large number of password/session ID options.

(v). Information Gathering
Attacks whose purpose is not to actually perform an attack, but rather to reveal information on the system, which can further assist in other attacks.

IV. Discovering Web Application Vulnerabilities
Protecting web applications is an around-the-clock job. To defend against targeted attacks, organizations can deploy a scanner to check web applications for vulnerabilities such as SQL injection, cross site scripting (XSS), and forceful browsing; or they can use a web application firewall (WAF) to protect against these vulnerabilities.

First, security is rarely considered during the functional requirements phase [5]. Second [6], even when developers do consider security, they are covering only the basics: authentication, authorization, access control and encryption. They often do not provide comprehensive input validation to prevent SQL-injection and cross-site scripting defects. As a result security defects arise in source code.

Remedying security problem during the development process is not something that can be achieved immediately. It takes time to integrate security into the various stages of software development.

Web Application Assessment Tools
- Web Browser
- Man-in-the-middle HTTP / HTTPS proxy
- Enumeration and fuzzer software
- Encoders / Decoders and
- Profiling the Platform
- Profiling the Application

There are many ways to exploit web applications. Some of the most common methods used by penetration testers:
1. Session Hijacking
2. URL Manipulation
3. Invalidated Iput
4. Improper Error Handling
5. Unencrypted Network Traffic
6. Denial of Service, DDoS
7. Permissions Misconfiguration
8. Hidden Values in Source Codes

For session hijacking, mismanagement and exploitation of session (variable protocol), token or cookies is used. There is a technique called “Vertical Privilege Escalation” where intruders altering the hidden directory parameter to access admin rights, once they had logged into system with normal user account. Care should be taken for invalidated input and improper error Random feeding server-side script (ie. CGI)

B. Discovering vulnerabilities in Web Applications

1. Scan the Applications
Scanning web application is one solution to discover vulnerabilities[10] Hardware scanners are widely used to scan apps .These scanners will scan apps and identify the apps which have insecure code, then fix the vulnerability [7]. NTOSpider, featuring new Universal Translator technology, is the only Dynamic Application Security Testing (DAST) solution available that is
capable of effectively testing modern mobile and web applications that leverage new technologies like REST, AJAX, JSON and DYNAMIC APPLICATION SECURITY TESTING (DAST). Scanner is the fastest way to locate vulnerabilities like cross site request forgery (XSRF)

2. Using Web Application Firewall (WAF)

WAF is another method or tool to discover and prevent vulnerabilities. WAFs are defined by the consortium as an intermediary device, sitting between a Web Client and a Web Server, analyzing OSI Layer 7 messages for violations in the programmed security policy. WAFs address different issues than network firewalls, which defend the perimeter of a network. A WAF is used as a security device protecting the Web server from an attack as shown in the figure. A WAF does not require modification of source code. A WAF can use a proxy-based architecture, a deep packet inspection-based architecture or both.

Using automation to discover and catalog web application services.
- Scanner should automate discovery and cataloging, which otherwise overwhelm manual review.

4.2.4. Establishing an security program

a) Address everything it takes to develop, deploy and maintain secure web applications.
b) Focus on obtaining enterprise-wide visibility, scale and results.

4.3. Following are necessary steps for effective Web Application Security

- Generic Input Validation
- Source Disclosure
- Character Encoding
- Identify y a vulnerable parameter.
- Examine errors for indicators of a SQL injection.
- Examine errors for information on database, table, and column names.
- Query standard variables for the type of database.
- Determine system-specific users.
- Determine database-specific users.
- Determine application-specific users.
- Query standard database objects
- Record available databases, tables, columns, and known row values.
- Query arbitrary data from application tables.
- Use OR TRUE=TRUE commands to bypass authentication.
- Insert arbitrary data into standard database tables.
- Insert arbitrary data into application tables.
- Attempt to read and write files on the operating system.
- Execute arbitrary commands on the database’s host operating system
- Send files to an FTP, HTTP, TFTP server or netcat listener.

- Write files to the web document root.
- Overwrite important configuration files.
- Denial of service

V. Conclusion

Protecting web applications is an clock –around task. Several approaches are coming into market to provide security for these web applications. In this paper, we focus on study and analysis of web application vulnerabilities and how they affect web applications. Also mentioned some preventive measures to discover vulnerabilities in web applications.

References

[8] NTOSpider, Application Security Scanner