A Survey on web application vulnerabilities, detection, and Testing

Anirudh Sunil
University of Massachusetts Lowell
Lowell, Massachusetts, USA

***Abstract*— Web-application security is a serious concern with almost all websites and applications on the internet requiring some personal information from the end user. With the increase in unauthorized use of personal data and privacy breach, it is important for website developers to understand the vulnerabilities of web applications and how to detect and test the application for potential threats. The paper will include a survey on web threats like SQL injections, cross-site scripting, broken authentication and how testing techniques can help in early detection of these vulnerabilities to ensure the security and confidentiality of data and information.**

I. Introduction

Today's web apps are quite functional and depend on two-way information exchange between the server and browser.Nobody wants to use a web application if they think their information may be leaked to unwanted parties, therefore security becomes a major concern.

The Vulnerabilities usually found in web applications are SQL injection, cross-site scripting, security misconfiguration, broken authentication, session management, and much more. Vulnerability assessment and penetration testing may be used to assess the vulnerabilities of web applications. A technique for assessing security by simulating an attack is called penetration testing.

A vulnerability assessment's goal is to identify a website or computer system's weaknesses and assist the system owner in addressing them.

Tunestore is used as an example online application in this survey's security testing. It provides an example of tools and manually assisted web application security testing. Testing on Tunestore is done using Paros, WebScarab, JBroFuzz, and Acunetix. According to the survey, manual testing is crucial because some vulnerability types can only be discovered through manual testing and tester observations. Additionally, to find most of the vulnerabilities in a web application, it's critical to use a variety of tools and to carefully conduct manual testing.

II. Web application Vulnerabilities:

Web vulnerabilities are weakness or shortcoming in a system that exposes the whole system to be attacked by hackers. This vulnerability can be either software code flaw/ bug, system misconfiguration, or some other weakness on the website/ web application or its components and processes. This reduces the overall security of the system and ultimately leads to the information being leaked and used for unethical purposes.

Having access to sensitive information , hackers can orchestrate attacks, takeover applications, engage in privilege escalation to exfiltrate data, cause large-scale service disruption.

With a clear understanding of what website vulnerabilities are and how they can be prevented, organizations can be better equipped to avert attacks and harden their security posture. The sections below define each vulnerability and provide a method to detect and test the systems .

A. *SQL injection attack :*

Structured Query Language, or SQL for short, is a computer language used to interact with databases. Many servers that house crucial data for websites and services handle the information in their databases using SQL.

This type of server is the subject of a SQL injection attack, which uses malicious code to persuade the server to provide information that it usually wouldn't. This is particularly troublesome if the server retains sensitive user data from the website, such as credit card numbers, usernames, passwords (credentials), or other personally identifiable data, which are profitable and alluring targets for hackers.

For example, if an application is vulnerable to an injection attack, it may be possible for an attacker to go to a website's search box and type in code that would instruct the site's SQL server to dump all its stored usernames and passwords for the site.

Unsanitized input is a common type of SQLi attack in which the attacker provides user input that isn't properly sanitized for characters that should be escaped, and/or the input isn't validated to be the type that is correct/expected.

A blind SQL injection attack doesn't reveal data directly from the database being targeted. Rather, the attacker closely examines indirect clues in behavior. Details within HTTP responses, blank web pages for certain user input, and how long it takes the database to respond to certain user input are all things that can be clues depending on the goal of the attacker. They could also point to another SQLi attack avenue for the attacker to try.

B. cross-site scripting (XSS) :

This type of attack is similar to the SQL injection which involves injecting malicious code into a website or web-based app , but if the hacker chooses to directly attack the users of the web application , they would opt for the XSS attack .

In the case of the XSS attack , the malicious code the hacker has injected only runs in the user's browser when they visit the attacked website and goes after the visitor directly.

One of the most common ways an attacker can deploy a cross-site scripting attack is by injecting malicious code into an input field that would be automatically run when other visitors view the infected page. For example, they could embed a link to malicious JavaScript in a comment on a blog.

Cross-site scripting attacks can significantly damage a web company's reputation by placing the users' information at risk without any indication that anything malicious even occurred. Any sensitive information a user sends to the site or the application—such as their credentials, credit card information, or other private data—can be hijacked via cross-site scripting without the owners realizing there was even a problem in the first place.

There are 3 types of XSS attacks :

1) Reflected XSS, in which dynamic web pages take text submitted by a client or user and simply renders this text back to user within its response.

2) Stored XSS , occurs where some malicious user-submitted data is stored in a database to be used in the creation of pages that will be served to other users later.
A key differentiator between reflected and stored XSS attacks is that stored XSS attacks consider all users of a vulnerable site/app as targets for attack.

3) Local XSS, exploits targets vulnerabilities that exist within the webpage code itself. Usually, this happens when document Object Model (DOM) is wrongly used in JavaScript. As a result, opening another web page with malicious JavaScript code in it at the same time might actually alter the code in the first page on the local system.

C. Broken authentication and session management :

It is a situation where the web application fails to appropriately perform its authentication and session management features, resulting in a vulnerability.

To mimic users, the attacker exploits vulnerabilities or defects in the authentication or session management mechanism, such as exposed accounts, session IDs, and passwords. When a hacker is successful, he or she can do whatever the victim can do.

For example : hackers can target password change, forgot password, remember my password, account update, and other related functions.

D. Insecure Direct Object References :

A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, database record, or key as a URL or form parameter. Unless an access control check is in place, an attacker can use direct object references to gain unauthorized access to other objects.

Cross-site request forgery usually occurs where HTTP cookies are primarily used to transmit session tokens. Where precaution is not taken against the misuse of the token, an attacker can trick an authenticated user to perform actions of his (attacker) choosing. Victims are forced to change data and perform functions they are allowed or authorized to do.

Having discussed the various web application vulnerabilities, we can now transition toward looking at the methods to detect these vulnerabilities.

III. VULNERABILITY ASSESSMENT :

A vulnerability assessment's goal is to identify a website or computer system's weaknesses and assist the system owner in addressing them.

This assessment can be performed manually or automatically. The manual assessment would include the web tester following a set protocol to identify the vulnerabilities. If the manual method is time-consuming, the tester can shift to an automatic assessment detection.

Vulnerability assessment can help reassure developers and users about the security situation of a particular web application. This in-turn helps to build a trustworthy relationship with the customers and retain a loyal user- base.

A. The step-by-step process of vulnerability assessment:

1) The first step is to determine the scope and extent of how the assessment can be used, which usually depends on the type of system that the tester chooses to scan for vulnerabilities.
The tester can select an application scanner, network scanner, or host-based scanner depending on the scope of the scan.

2) The vulnerability scan is the second stage. An automated scanner will now search the target system for common vulnerabilities using a vulnerability database.

3) The third step is the vulnerability assessment report. According to their severity and the danger they represent to the system and the business, vulnerabilities detected in a system are listed in a vulnerability assessment report.

4) the next step is vulnerability remediation, which involves assigning the vulnerabilities to the developers who can follow the suggestions given in the report and fix them.

B. Tools used in vulnerability Assessment

1) Paros:

This tool is a web application assessment tool written in Java. It can be used as a web spider or vulnerability scanner.
The web spider feature is used to identify the content of the web application by parsing a webpage for links and requesting these pages.

The scanner function is used to scan the web application to identify vulnerabilities such as cross-site scripting, SQL injection, forms with autocomplete enabled, and any old versions of files. Paros can also trap HTTP and HTTPS requests and responses so they can be modified manually.

2) WebScarab :

An open-source web application security testing tool written in Java can be used for the analysis of web applications that communicate with HTTP and HTTPS protocols.

This tool is like paros and can be used to discover content of a website. A feature called fuzzer of WebScarab allows a tester to test a web page by taking a list of test strings and trying them in the input fields automatically.

3) JBroFuzz :

JBroFuzz is a web application fuzzer written in Java , the role of the fuzzer here is for requests made over HTTP and HTTPs protocols.

Here, the fuzzer creates a HTTP request to send to the web application. It can also be used as an attack payload which has attack strings of several vulnerability categories such as Injection, XSS, Integer Overflow, XPATH Injection, SQL Injection.

The HTTP responses are then displayed in an output window showing the payload and the HTTP response status code.
Hence, this feature can help in detecting the type of attack the website might be exposed to.

d) Acunetix :
This tool is a web vulnerability scanner created to replicate a hacker’s methodologies to find vulnerabilities in web applications.

This scanner has features like sophisticated and thorough testing for SQL injection and cross-site scripting, and penetration tools like HTTP Editor and HTTP Fuzzer.

Now that we have touched upon the tools for vulnerability assessment, we can look towards another way of ensuring applications maintain their security.

IV. Web Application testing

To identify and assess security flaws that an attacker may exploit, web app penetration testing involves simulating a hacker assault on your online application. The goal of the web application penetration test is to provide you with a better knowledge of the security posture of your web app, including its fortitude and resistance to online attacks.

Penetration testing, as opposed to vulnerability assessments, which essentially identifies and lists all current vulnerabilities in your website, focuses more on how each of these flaws may be exploited.

Pentesting uses the list of vulnerabilities found from the vulnerability assessment and then exploits them to check the extent of risk associated with them.

Pentensting is generally manual testing whereas vulnerability assessment is both automated and manual scan.

A. Methods involved in Penetration testing :

1) Gathering Information : The tester searches the website's backend for fingerprints while gathering information.
Example : Server OS, CMS version, etc.

2) Discovery: Here, a manual check by engineers is necessary to find business logic vulnerabilities because automated techniques frequently overlook these weaknesses.

3) Exploitation :The objective of this stage is to take advantage of any weaknesses found in discovery and ensure the web application is immune to them.

B. Tools in Penetration testing :

1. Astra’s Pentest :

The need to make web application security simpler for consumers is what inspired the creation of Astra Security.

There are several benefits to using this web application penetration testing tool. Astra's pentest suite, for instance, may be integrated with CI/CD (continuous Integration and continuous deployment) systems such that every time there is a code update, an automatic scan is launched.

2. NMAP :

NMAP also called Network mapper, is an open-source program that aids in network mapping by port-scanning, identifying operating systems, and compiling a list of hardware and the services it supports.

For various transport layer protocols, it transmits packets with various structures, and those packets return with IP addresses and other data.

3. Wireshark :

Another well-known open-source tool for protocol analysis is Wireshark. It enables microscopic monitoring of network activity.

Wireshark can perform several activities given below :

•Live capture and offline analysis
•Inspection of hundreds of different protocols
•Browse captured data via GUI
•Decrypt protocols
•Read live data from Ethernet and several other mediums
•Export output to XML, PostScript, CSV, or plain text

4. Metasploit :

Ethical hackers utilize this open-source Metasploit framework, which is built on Ruby, to look for widespread vulnerabilities on servers and networks. Additionally, the Metasploit architecture has evasion, anti-forensic, and fuzzing tools.

5. Burp Suite :

Burp Suite is a website pentesting framework built on java , features an integrated proxy that snoops on traffic going between your browser and the website you're pentesting.

Then, this proxy may be used to alter requests or for fuzzing, which is the process of identifying weaknesses in a website.

Example of tools included in the burp suite :

Spider : It can be utilized to map the intended application. It enables you to compile a list of every endpoint, keep an eye on how they perform, and search for vulnerabilities.

Proxy: Between the browser and the internet, a proxy is placed to track and edit requests and replies as they are being sent and received.

V. Implementation of these tools on an example application

The application used to conduct security testing is the Tunestore web application developed by Dr. Bei-Tseng Bill Chu’s project team from the University of North Carolina at Charlotte.

This application is installed on a server on an Ubuntu Virtual Machine to conduct security testing.

Tunestore is an online retail store of songs , where a user can create an account to gain access to the store and its features.
The Tunestore web application was tested using the four tools described in section III .

VI. Results

A. Testing Results with Paros :

37 possible vulnerabilities in the Tunestore online application were found using the Paros scanning tool.

Example :

Obsolete file: This suggests the existence of several objects such as files, backups, useless files, or out-of-date files.
Since certain file extensions might not be handled by the web server, information like user IDs and passwords may be exposed.

SQL Injection: This result indicates that SQL injection is possible. To handle the user parameters supplied to the database, a SQL query will be created. If the query is created by simply concatenating strings, the argument might be carefully chosen to alter the query's meaning.

B. Testing Results with WebScarab :
With the use of the fuzzer and the proxy features of WebScarab, potential SQL injection and cross-site scripting vulnerabilities were discovered.

Example :

SQL Injection: An input attack file was created which included SQL injection attack strings.

This attack file was then submitted to the WebScarab fuzzer to test the login page of the Tunestore web application. Some of the attacks were successful.

Reflected Cross-Site Scripting: Using WebScarab's fuzzer feature, an input attack file was created which included cross-site scripting attack strings to test the login page of the Tunestore web application. The scripts were executed showing that the login page has XSS vulnerabilities.

C. Testing Results with JBroFuzz and Acunetix :

JBroFuzz was first used to check for XSS and SQL injection vulnerabilities. But it was found that the tool had a flaw, JBroFuzz created incorrect requests to the web application for some viable SQL injection attack strings, making it unable to detect the SQL injection vulnerability.

Result of this bug, JBroFuzz was no longer used to test for other vulnerabilities in the Tunestore web application. However, the payloads embedded in JBroFuzz were used in manual testing to discover some of the vulnerabilities in Tunestore. Using Acunetix , cross-site scripting vulnerabilities were discovered.

VII. Discussion

Our survey shows that Vulnerability assessment and vulnerability testing can be used to demonstrate various web application vulnerabilities to developers and users.

The web spidering and proxy features of Paros and WebScarab are very useful for vulnerability assessment. The web spidering functions allow testers to understand the content and structure of the web application.

The proxy enables testers to carry out attacks by intercepting and changing requests and replies sent back and forth between the client and server.

The fuzzer function of WebScarab may be quite helpful since several attack strings can be automatically provided to test the application.

Further, Manual penetration testing can leverage the findings of vulnerability assessment to discover most of the vulnerabilities, especially the design flaws. Information collecting, discovery, and exploitation, the three phases of a web penetration test, will direct and organize the whole procedure.

Only manual testing based on tester observations can find the majority of authentication and access control issues. To detect the most vulnerabilities in a web application, it is crucial to employ a range of technologies and carry out comprehensive manual testing.

VIII. Conclusion

This survey examines how a web application can be assessed and tested for security flaws.

It offers a summary of web application security testing software, including Paros, WebScarab, JBroFuzz, and Acunetix. It lists the security flaws that various tools and manual testing have discovered in an application. Since some vulnerability types may only be discovered by manual testing and tester observations, our case study demonstrates the importance of manual testing as well.

To uncover the most vulnerabilities, it's critical to use a range of tools and to carefully do manual testing.

Since Web apps are becoming more and more popular as targets for security threats as they are utilized for various crucial services, making web applications more secure is crucial in this circumstance. A comprehensive evaluation of the security of a web application must include the effective use of these technologies.

References

[1] P. S. Aarya, A. Rajan, K. P. S. Sachin, R. Gopi and G. Sreenu, "Web Scanning: Existing Techniques and Future," 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), 2018, pp. 123-128, Doi: 10.1109/ICCONS.2018.8662934.

[2] L. Dukes, X. Yuan and F. Akowuah, "A case study on web application security testing with tools and manual testing," 2013 Proceedings of IEEE Southeastcon, 2013, pp. 1-6, doi: 10.1109/SECON.2013.6567420

[3] Vulnerability Assessment: https://www.getastra.com/blog/security-audit/vulnerability-assessment/