Web Care

Kavyashree K, A.Ananda Shankar

Abstract— Web applications are becoming more prevalent over the globe such as in corporate, public and also in government services today. Web applications provide convenience and efficiency but it also encounters number of new security threats frequently, which could potentially pose significant risks to any organization, if not handled properly. Web security vulnerabilities continually impact the risk of a web site. Performing the attack requires several application attack techniques. These techniques are commonly referred to as the class of attack.
Keywords: Security threats, Vulnerabilities, Web Security

1 Introduction

THE Open Web Application Security Project (OWASP) is a worldwide not-for-profit charitable organization focused on improving the security of software.
Our mission is to make software security visible, so that individuals and organizations worldwide can make informed decisions about true software security risks [1]. Vulnerability is a hole or a weakness in the application, which can be a design flaw or an implementation bug that allows an attacker to cause harm to the stakeholders of an application. Stakeholders include the application owner, application users, and other entities that rely on the application. The term "vulnerability" is often used very loosely. However, here we need to distinguish threats, attacks, and countermeasures.
Examples of vulnerabilities [3]

- Lack of input validation on user input
- Lack of sufficient logging mechanism
- Fail-open error handling
- Not closing the database connection properly

Attacks are the techniques that attackers use to exploit the vulnerabilities in applications. Attacks are often confused with vulnerabilities.

2 Procedure for Paper Submission

Kavyashree K is currently pursuing master's degree in Advanced Information Technology in Reva University, India, PH-+91 9739466588. E-mail: kavyashreek36@gmail.com
A.Ananda Shankar is the Associate Professor in School of Computing and Information Technology in Reva University, India

3 Literature Survey

Web application security vulnerabilities detection approaches [2]:

Number of security vulnerabilities in web application has grown with the tremendous growth of web application in last two decades. As the domain of Web Applications is maturing, large number of empirical studies has been reported in web applications to address the solution of vulnerable web application. However, before advancing towards finding new approaches of web applications security vulnerability detection, there is a need to analyze and synthesize existing evidence based studies in web applications area. To do this, we have planned to conduct a systematic mapping study to view and report the state-of-the-art of empirical work in existing.
The proposed solutions are mapped in my project
(1) The software development stages for which the solution has been proposed
(2) The web application vulnerabilities mapping according to OWASP Top 10 security vulnerabilities.

A Survey on Web Application Security [4]

As web applications are increasingly used to deliver security critical services, they become a valuable target for security attacks. Many web applications interact with back-end database systems, which may store sensitive information (e.g., financial, health), the compromise of web applications would result in breaching an enormous amount of information, leading to severe economic losses, ethical and legal consequences. The Web platform is a complex ecosystem composed of a large number of components and technologies, including HTTP protocol, webserver and server-side application development technologies (e.g., CGI, PHP, ASP), web browser and client-side technologies (e.g., JavaScript, Flash).

UNDERSTANDING WEB APPLICATION SECURITY PROPERTIES, VULNERABILITIES AND ATTACK VECTORS [5]

- SQL Injection
- Cross-Site Scripting
- Logic Correctness
- Input Validation
- Security by construction
- Security by verification

End-to-end Web Application Security

Web applications are important, ubiquitous distributed systems whose current security relies primarily on server-side mechanisms. This paper makes the end-to-end argument that the client and server must collaborate to achieve security goals, to eliminate common security exploits, and to secure the emerging class of rich, cross-domain Web applications.
In order to support end-to-end security, Web clients must be enhanced. We introduce Mutation-Event Trans-forms like:
Motivating Attacks
The Case for End-to-end Defenses
Server-side Defenses and their Limitations
New Client-side Security Mechanisms
Client-side Defenses and their Benefits
worms, etc.

New Client-side Security Mechanisms

Web Server Security and Survey on Web Application Security

A secure Web server provides a protected foundation for hosting Web applications, and Web server configuration plays a critical role in Web application's security. Badly configured virtual directories, a common mistake, can lead to unauthorized access.

A forgotten share can provide a convenient back door, while an overlooked port can be an attacker's front door.
Neglected user accounts can permit an attacker to slip by your defenses unnoticed.
The fact that an attacker can strike remotely makes a Web server an appealing target. Understanding threats to Web server and being able to identify appropriate countermeasures permits to anticipate many attacks and thwart the ever-growing numbers of attackers.

Threats to Web Server and Countermeasures

The main threats to a Web server are:

- Profiling
- Denial of service
- Unauthorized access
- Arbitrary code execution
- Elevation of privileges
- Viruses, worms, and Trojan horses
- Attacks
- Common attacks used for profiling include:
- Port scans
- Ping sweeps
- NetBIOS and server message block (SMB) enumeration

4 VULNERABILITIES

4.1 CROSS SITE SCRIPTING

Cross site scripting is a type of computer security vulnerability typically found in web applications. XSS enables attackers to inject client-side scripts into web pages viewed by other users. A cross-site scripting vulnerability may be used by attackers to bypass access controls such as the same-origin policy.
Request validation is a feature that contains potentially dangerous content. This exploit is typically referred to as a cross-site scripting (XSS) attack
Categories of XSS attacks:

- Stored: The injected code is permanently stored (in a database, message forum, visitor log, etc.)
- Reflected - Attacks that are reflected take some other route to the victim (through an e-mail message, or bounced off from some other server)

4.1.1 Countermeasure for CSS attack

We can encode all the values coming as input from page or from web service or from API. XSS flaws occur whenever an application takes user supplied data and sends it to a web browser without first validating or encoding that content. XSS allows attackers to execute script in the victim's browser which can hijack user sessions, deface web sites, possibly introduce other than names and passwords. Unless such precautions are taken, an attacker can use the input boxes to send their own

4.2 Cross Site Request Forgery (CSRF)

A CSRF attack forces a logged-on victim's browser to send a pre-authenticated request to a vulnerable web application, which then forces the victim's browser to perform a hostile action to the benefit of the attacker. CSRF can be as powerful as the web application that it attacks. Malicious website can perform action using your authentication. Server authenticates user and response from the server that includes authentication cookie.
Without logging out, user visited a malicious website. That malicious website may contain malicious image link, post action to site user is logged in, malicious script or Ajax call which will be totally hidden from visitor. This is the "cross site" part of CSRF attack.
Applications are vulnerable if any of following:

- Does not re-verify authorization of action
- Default login/password will authorize action

Action will be authorized based only on credentials which are automatically submitted by the browser such as session cookie, Kerberos token, basic authentication, or SSL certificate etc.

4.2.1 Countermeasure for CSRF attack

XSS is a major channel for delivery of CSRF attacks. Generate unique random tokens for each form or URL, which are not automatically transmitted by the browser. Do not allow GET requests for sensitive actions. For sensitive actions, re-authenticate or digitally sign the transaction. User will click the image or any button (filling forms and clicking submit), then browser will happily send the authentication cookie for the request because post request is like (www.examplewebsite.com/account/delete)

Figure 4.2.1: CSRF attack

4.3 SQL injection

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 a request for some action to be performed on a database. Typically, on a Web form for user authentication, when a user enters their name and password into the text boxes provided for them, those values are inserted into a SELECT query. If the values entered are found as expected, the user is allowed access; if they aren't found, access is denied. However, most Web forms have no mechanisms in place to block input

request to the database, which could allow them to download the entire database or interact with it in other illicit ways. SQL Injection. There are various approach in Entity Framework like Schema First Approach, Model First Approach and Code First Approach

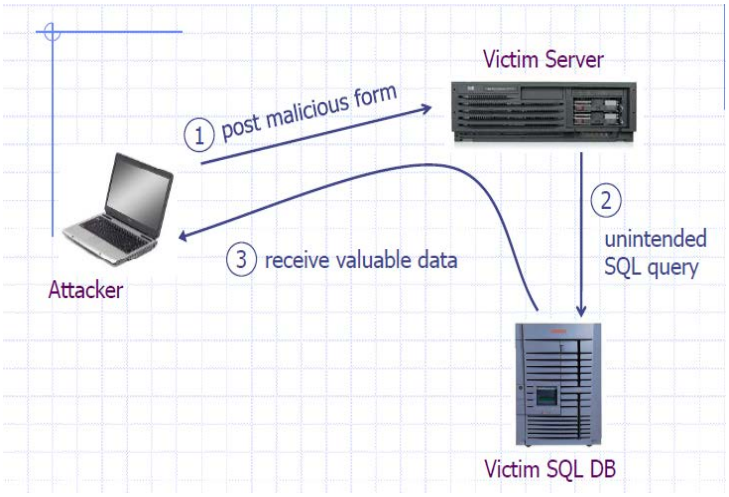

Figure 4.3.1: Basic SQL injection

4.3.1 Countermeasure for SQL injection

- Use language specific libraries to perform the same functions as shell commands and system calls
- Check for existing reusable libraries to validate input, and safely perform system functions, or develop your own.
- Perform design and code reviews on the reusable libraries to ensure security.

Other common methods of protection include:

- Use stored Procedures
- Data validation (to ensure input isn't malicious code),
- Run commands with very minimal privileges
- If the application is compromised, the damage will be minimized.

Using Inline Queries is not the best way as we are writing business logic inline. Every time we have to write again this query. The best way to prevent from SQL injection is use stored procedure. As business logic are hidden, it provides better performance, reusability.

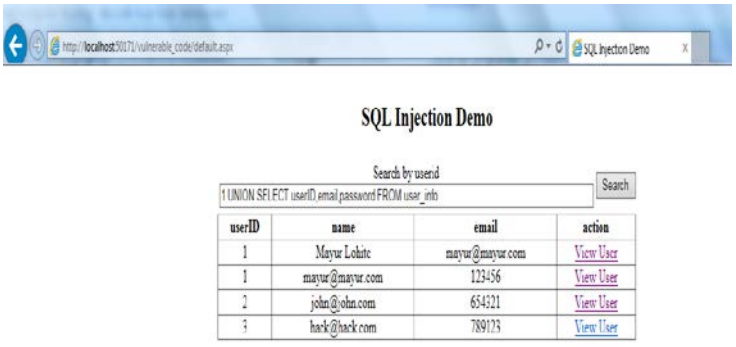

Figure 4.3.2: Vulnerable Page

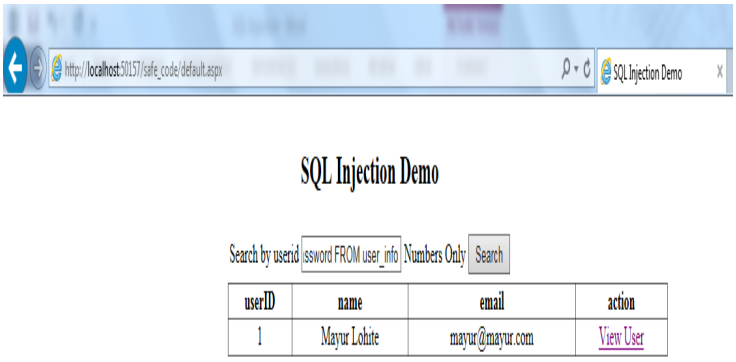

Figure 4.3.3: Countermeasure for SQL Injection

4.4 URL Redirection and URL Routing

URL parameters must be encrypted and different name should be given as parameter.
Server side validations should be performed to check whether they are white list url's.

Page extensions like aspx, php, jsp, nl are hidden from public users. Routing is implemented as a countermeasure. Page name cannot be seen even in page properties or in any of the tools.
Ex: http://localhost:64690/Category/ProductList.aspx
4.4.1 Countermeasure:
http://localhost:64690/Category/Cars
http://localhost:64690/Category/Planes

4.5 Insecure Direct Object Reference

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. Attackers can manipulate those references to access other objects without authorization.

- Applications often expose internal objects, making them accessible via parameters.
- When those objects are exposed, the attacker may manipulate unauthorized objects, if proper access controls are not in place.
- Internal Objects might include
  - Files or Directories
  - URLs

Database key, such as acct_no, group_id etc.

4.5.1 Countermeasure for Insecure Direct Object Reference

- Do not expose direct objects via parameters
- Use an indirect mapping which is simple to validate.
- Re-verify authorization at every reference.
  For example: Application provided an initial lists of only the authorized options. When user's option is "submitted" as a parameter, authorization must be checked again.

6 References

[1] "About Critical Web Application Security Risks"

Figure 4.5.1: Account Information for 1344573490

Figure 4.5.2: Changing account number parameter from 1344573490 to 1344573491

4.6 Cryptographic algorithms

The use of a non-standard algorithm is dangerous because a determined attacker may be able to break the algorithm and compromise whatever data has been protected. Well-known techniques may exist to break the algorithm [6].

4.6.1 Countermeasure for Cryptographic algorithm

Check algorithms whether they are broken or safe in CWE (Common Weakness Enumeration) site.

5 Conclusion

The security standards set by OWASP for the websites should be followed. The applications must take care of all the major vulnerabilities that an attacker targets.

Web applications have been evolving extraordinarily fast with new programming models and new emerging technologies, so there are lot of new challenges in web application security, which requires a lot of effort from security researchers.

Web applications reach out to a larger number of users, and yet they are more vulnerable to attacks. Many companies have starting Security Code reviews, extensive penetration testing to secure their websites and also there client sites.

In this paper, we have demonstrated several common web application vulnerabilities, their countermeasures and their criticality.

https://www.owasp.org/index.php/Main_Page

[2] "Web application security vulnerabilities detection approaches: A systematic mapping study" https://www.computer.org/csdl/proceedings/snpd/2015/8676/00/07176244.pdf

[3] "Vulnerability (computing)" https://en.wikipedia.org/wiki/Vulnerability_(computing)

[4] "A Survey on Web Application Security" https://pdfs.semanticscholar.org/4090/860cf6eb3c574bc41612585fb9452251b37.pdf

[5] "Understanding Web Application Security Defending the Enterprise's New Porous Perimeter by Extending Security to the Edge" https://www.motiv.nl/documenten/whitepapers/akamai-web-application-security-whitepaper.pdf

[6] "Use of a Broken or Risky Cryptographic Algorithm" https://cwe.mitre.org/data/definitions/327.html