STM JOURNALS

Journal of
Web Engineering & Technology

ISSN: 2455-1880
Volume 11, Issue 1, 2024
January—April
DOI (Journal): 10.37591/JoWET

https://journals.stmjournals.com/jowet

Review JoWET

Web Security: Crafting Fortified Online Platforms

Riya Premarajan Vechiot*, Yogita Vijay Biradar

Abstract

With incidents of cyber-attacks on websites and breaches of sensitive data on the rise, adopting secure coding or development practices to build strong web applications is long overdue. These practices encompass a variety of strategies, including but not limited to, input validation to prevent SQL injection and XSS attacks, secure session management, encryption of sensitive data, and the implementation of robust authentication and authorization mechanisms. This approach not only fortifies applications against known web vulnerabilities but also ingrains a culture of security within the development process. Drawing from a wealth of knowledge shared by industry professionals, this research work underscores the importance of incorporating these secure development practices from the outset. Furthermore, adopting a security-first mindset encourages the ongoing evaluation and updating of security measures to combat evolving cyber threats. This research work compiles insights from various sources by industry experts. Applying these practices at the development stage would result in applications that are not susceptible to known web application vulnerabilities.

Keywords: Web-security, web applications, best practices, vulnerability prevention, security-first development

INTRODUCTION

In today's world, where our daily lives are significantly intertwined with digital space, ensuring the safety of web applications is of utmost importance. As we navigate through the digital landscape, the risks associated with cyber threats become more pronounced, highlighting the necessity for robust security measures [1].

Security as an Afterthought

For many firms, security is not a part of the development process. The penetration tester frequently conducts a security test prior to production. Typically, the penetration tester has a brief window of time to look for vulnerabilities, which is insufficient. This results in applications that lack security measures [2].

The Importance of Security for Web Developers

Users entrust developers with their sensitive data, so it becomes the developer's responsibility to handle the data responsibly and protect it from breaches. Developers also bear the responsibility of ensuring compliance with privacy regulations [3].

A breach in security not only puts user data at risk but also has the potential to permanently tarnish the reputation of developers and organizations.

Security incidents may also affect overall business continuity, leading to big financial losses for organizations [4].

Now that we know how important web application security is for developers and not just for users, developers are required to constantly be

*Author for Correspondence
Riya Premarajan Vechiot
E-mail: vechiotriya@gmail.com

Research Scholar, MCA, Thakur Institute of Management Studies, Career Development & Research (TIMSCDR), Mumbai, Maharashtra, India

Received Date: February 29, 2024
Accepted Date: March 30, 2024
Published Date: April 03, 2024

Citation: Riya Premarajan Vechiot, Yogita Vijay Biradar. Web Security: Crafting Fortified Online Platforms. Journal of Web Engineering & Technology. 2024; 11(1): 27–31p.

up to speed with new security vulnerabilities and threats and how to mitigate them. Organizations should have proper backup and testing, avoid unreliable third-party tools, and regularly revisit security measures before launching the application to production. This research is a dedicated exploration into the world of secure web application development, focusing on distilling practical and effective best practices. Rather than delving into intricate technicalities, the primary goal is to offer a user-friendly guide that developers and organizations can readily adopt [5].

LITERATURE REVIEW

Secret Management

These days, secrets are employed everywhere, particularly with the rise in popularity of the DevOps movement. SSH keys, certificates, IAM permissions, API keys, database credentials, and similar sensitive information are often dispersed across configuration files and management tools, or embedded directly in the source code in plain text, within numerous organizations. In order to regulate access to secrets, stop them from leaking, and stop the organization from being compromised, there is an increasing need for businesses to centralize the storage, provisioning, auditing, rotation, and administration of secrets. Example: An inadvertent disclosure by IT company Infosys led to the publication of a file on PyPi containing AWS keys to an S3 bucket, possibly holding patient data from Johns Hopkins University. This file remained publicly accessible for over a year. Software engineer Tom Forbes noticed the Infosys leak file when he received a pull request to eliminate data from a GitHub project of his, which catalogs metadata for each package on PyPi. The request came from a user attempting to remove a project named "ihip" [3]. To ensure effective secret management, standardization and centralization are crucial. This may involve using various solutions based on team preferences, such as cloud-native tools, third-party solutions, or password managers. Standardizing interactions across these solutions is essential for maintainability and incident response. Clear documentation is essential, especially when employing multiple secrets management solutions, to ensure immediate understanding of a secret's purpose and location. Having clear documentation is important, especially when using different secret-management tools. This ensures that anyone can quickly understand what a secret is, for and where to find it. Secrets should never be transported in the form of plaintext [6].

Secrets can be stored in Jenkins, Gitlab, or Github; this is not the same as committing them to a code, but with a few caveats:

- Secrets are exposed to CI/CD jobs and can be viewed or configured by people with permission. The secrets should not be long-term or of high value.
- Reduce the number of people with admin rights to decrease exposure.
- Regularly rotate the secrets.
- Make sure that forking the repository should not copy the secrets.

Preventing SQL Injection

SQL injection attack also known as SQLi is a cyberattack wherein an attacker injects fragments of SQL queries into database queries to extract information. For example: http://workksite.com/products.php?category=1%20 union%20select%20database(),user(),version()

Whitelisting User Inputs

Data sanitization and standardization are critical components in protecting against SQL injection vulnerabilities. SQLi attackers use special characters to communicate SQL code to the database via a web interface, therefore data must be sanitized to avoid concatenation or recognizing user input as commands which could be done in code making use of regular expression validations. Consider a login attempt in which an attacker tries to log in with the password: password 'or 1=1'. This involves deviously adding a 'true' statement (1=1) to the database query, which is then interpreted by the database as a command to grant access [7].

Using Prepared Statements

Prepared statements make it easier to create SQL queries. In this method of querying databases, the developer must first create the query, and then insert the parameter values. While parameter values are passed dynamically, this fixes the query statement that would run and makes it immutable. For example, an SQLi attack attempt by passing jim' or '1'='1 would fail as now “jim' or '1'='1” would be taken as one single parameter for the already prepared query [8].

Language specific recommendations: (CheatSheetSeries.owasp.org) [5]

- Use PreparedStatement() with bind variables in Java EE
- .NET: Use bind variables in parameterized queries such as SqlCommand() and OleDbCommand().
- PHP: Use bindParam() to use PDO with strongly typed parameterized queries.
- Hibernate: Use bind variables in createQuery() (referred to as named parameters in Hibernate).
- SQLite: To generate a statement object, use sqlite3_prepare().

Stored Procedures

Stored procedures have the same effect as prepared statements when it does not include unsafe dynamic SQL generation.

Stored procedures are implemented the following way:

- Java: CallableStatement.
- VB.NET: SqlCommand and CommandType.StoredProcedure.
- Limiting Database privileges.

While providing privileges to the application users, the developers should be cautious about what data the user is able to access with the current access control.

Use of WAFs

A network firewall prevents or limits unauthorized access to private networks by enforcing policies that define the only traffic permitted on the network; any other traffic attempting to connect is blocked.

Firewall programming utilizes access control lists (ACLs), which contain lists of permissions dictating which network traffic is allowed or blocked. The requirements that a data packet must fulfill in order for the ACL action: allow, deny, or reject, to be carried out are specified in an access control list [9]. For the most part, this firewall is insufficient to combat online threats like SQL injection, XSS, and cookie poisoning. Here is where WAF can help: it guards the web application server by preventing any unauthorized data from leaving the app and filtering, monitoring, and blocking any malicious HTTP/S traffic heading to the web application. In many cases, WAFs are a reliable first line of security for applications, particularly when it comes to defending against the OWASP Top 10, which is the definitive list of the most common application vulnerabilities. Some popular WAFs include:

- Fortinet FortiWeb.
- Imperva Cloud WAF.
- Barracuda Web Application Firewall.
- MS Azure Web Application Firewall.
- F5 Essential App Protect.
- Cloudflare WAF.
- Akamai Kona Site Defender.

Preventing XSS

Cross-site scripting (XSS), a prevalent web security vulnerability, empowers attackers to manipulate user interactions within vulnerable applications. By circumventing the same origin policy, which aims

to segregate different websites, XSS flaws enable attackers to impersonate users, execute actions on their behalf, and retrieve their data. In cases where victims possess elevated access, attackers could potentially gain full control over the application's functionality and data.

Upon receipt, carefully filter incoming input. Tailor the filtering process to accurately match the expected or valid input at the specific juncture where user input is accepted.

When presenting output, ensure data encoding. Encode any user-controlled data in HTTP responses to prevent it from being interpreted as active content. Depending on the context of the output, this might involve employing a mix of HTML, URL, JavaScript, and CSS encoding techniques.

- Use the proper response headers. To prevent XSS in HTTP answers that are not meant to contain any HTML or JavaScript, employ the Content-Type and X-Content-Type-Options headers to ensure that browsers read the responses correctly.
- Policy on Content Security: You can utilize Content Security Policy (CSP) as a last line of defense.

Using Latest Version of Packages that the Application Depends on

Vulnerabilities and issues could rise up at any time anywhere, this makes installing any updates to packages crucial. The developers need to be up to speed about any upcoming updates to the packages that have been used for the application. Updating packages not only secures the application but also could bring in new features.

Hide Direct Error Messages

End users should never be given access to the inner workings of the system. This allows a potential attacker to streamline his attempts and boost the likelihood of success. The error message should be user-friendly and should not expose any sensitive information about the system. Figure 1 displays one of the instances wherein a part of code is exposed to the user.

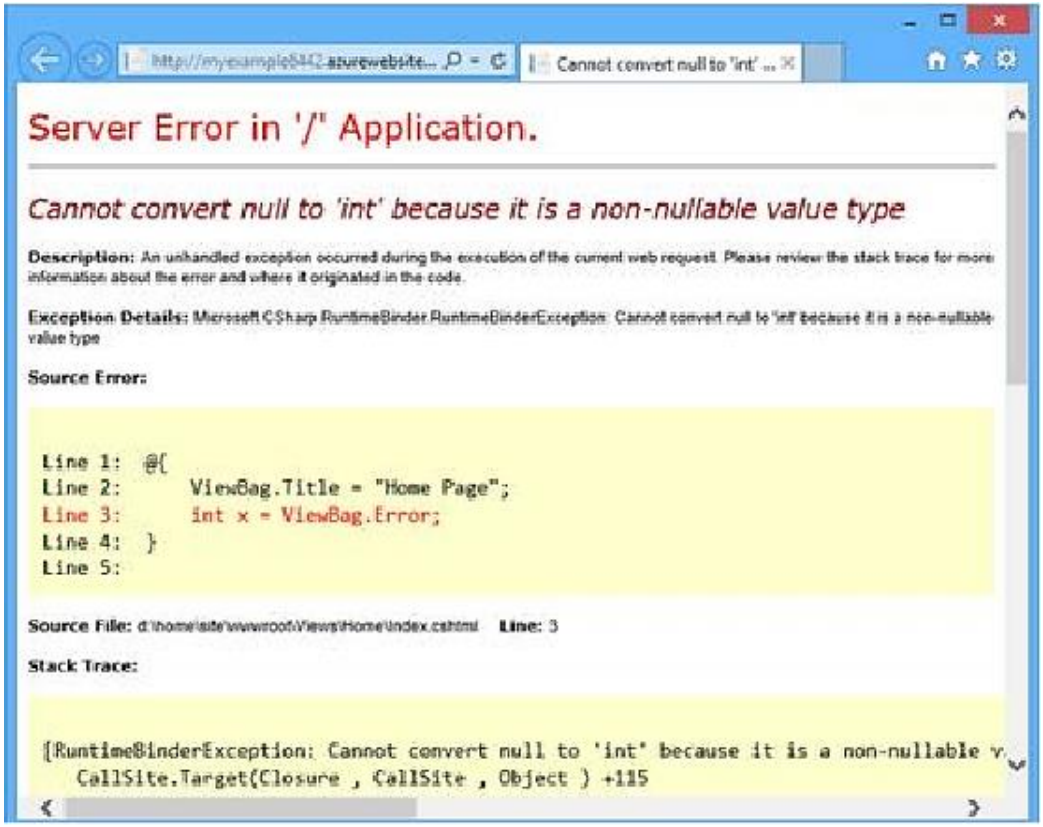

Figure 1. Represent error.

OWASP Top 10: "Globally Recognized by Developers as the First Step Towards More Secure Coding"

A multinational non-profit group devoted to online application security is called the Open online Application Security Project, or OWASP. One of the main tenets of OWASP is that anyone may enhance their own web application security by using the publicly available and user-friendly resources on their website. They include forums, movies, tools, and documentation among other stuff. Their most well-known endeavor is presumably the OWASP Top 10.

The OWASP Top 10, a periodically updated report, identifies the most significant security concerns pertaining to web application security. This report is compiled by a team of global security experts. OWASP calls the Top 10 a "document of awareness", and they advise all businesses to adopt the report into their process to mitigate security risks.

CONCLUSION

In conclusion, this research work emphasizes the critical need for adopting secure coding practices in web application development. With the increasing occurrence of cyber-attacks and data breaches, it is clear that security must be considered as an essential aspect of the development process, rather than an afterthought. The document underscores the duty of developers to protect sensitive user information and adhere to privacy laws.

By distilling insights from industry experts and focusing on user-friendly guidelines, this research work aims to empower developers and organizations to create robust, secure web applications. Recognizing security as a shared responsibility and integrating it into the development life cycle is crucial for mitigating risks, preserving user trust, and safeguarding the reputation of developers and organizations. As we navigate our interconnected world, the proactive adoption of these secure coding practices becomes not only a necessity but a fundamental commitment to the resilience and integrity of our digital landscape.

REFERENCES

1. Alabdulrazzaq Haneen. Securing Web Applications: Web Application Flow Whitelisting to Improve Security. PhD Dissertation. Auburn, AL: Auburn University; 2017.
2. Baars Nanne. (2018 May 22). Web Application Security: 10 Things Developers Need to Know. [Online]. www.youtube.com. youtu.be/qjrkV4RjgIU?si=ZIPfOCqWmNOqIj pr. Accessed 7 Oct. 2023.
3. Beer E. (2022). Infosys leak: IT firm left AWS key exposed on PyPi since Feb 2021. [online] The Stack. Available from: https://www.thestack.technology/infosys-leak-aws-key-exposed-on-pypi/
4. Marin Bratanov. (2019 Nov 06). First 5 Tips for Building Secure (Web) Apps. [Online]. Telerik Blogs. www.telerik. com/blogs/first-5-tips-for-building-secure-web-apps. Accessed 30 Oct. 2023.
5. OWASP Cheat Sheet Series. Secrets Management Cheat Sheets. [Online]. cheatsheet series.owasp.org/cheatsheets/Secrets_Manage ment_Cheat_Sheet.html. Accessed 22 Oct. 2023.
6. F5. What Is a Web Application Firewall (WAF)? [Online]. www.f5.com/glossary/web-application-firewall-waf. Accessed 27 Oct. 2023.
7. Haider A. (2023 Nov 16). Top 21 .NET Security Best Practices for Web Applications. [Online]. Clickysoft. Clickysoft. Available from: https://clickysoft.com/dot-net-security-best-practices/
8. zac1987. (2011). Can I fully prevent SQL injection by PDO Prepared statement without bind_param? [Online]. Stack Overflow. Available from: https://stackoverflow.com/questions/7915952/can-i-fully-prevent-sql-injection-by-pdo-prepared-statement-without-bind-param
9. Gayatri R. (2020 May 23). Stored Cross-Site Scripting(Non-Privileged User to Anyone). [Online]. Medium. Available from: https://gaya3-r.medium.com/stored-cross-site-scripting-non-privileged-user-to-anyone-1754e0a053d6

