Securing Modern Web Applications: Technologies, Threats, and Best Practices

Sandeep Phanireddy

phanireddysandeep@gmail.com

Abstract

Web Applications are now the foundation of many sectors today because they offer essential user services to the global population. However, as more organizations and individuals have started using these applications, they have become favorite points of call for hackers with security implications. This paper aims at describing the contemporary Web Application Security with emphasis on OAuth 2.0, WebAuthn, SSL/TLS, and Zero Trust Architecture. Many of the website threats are described, including the cross- site scripting (XSS) and SQL injection, and the issues of protection are considered, reflecting problematic aspects of the security system implementation within the organizations. This paper presents an analysis of defensive measures that have been adopted in the protection of web applications accompanied by lessons learned and the most important best practices to be followed by web application developers and managers. Web application security depicts a multidimensional problem whereby, with a strategic, multilayered approach, risks can be managed satisfactorily, users' information can be protected, as well as organizational compliance can be met successfully.

Keywords: Web application Security, Authentication and authorization standards, Security Protocols, Secure Communication, Token and Session management, OWASP Top10, Access Controls and API Security, Input Validations

INTRODUCTION

Web applications are the driving force that underpins many aspects of our lives, business, and interaction, including money management, healthcare, shopping, and much more [1], [2]. It is unfortunate that as these application forms become more complicated and versatile in their uses, they have also turned into victims of cybercrime. The last few years have witnessed a dramatic increase in various web application threats including identity theft, data leakage and insecure API misuse [3], [4]. This has put web application security not only as a technical advantage but as a business necessity. Web app security is a constantly growing field and has to deal with a wide spectrum of issues, including the protection of personal users' data, authentication, authorization, and protection against complicated threats, such as XSS or SQL injection [5]. Since organizations have shifted to the use of cloud based applications, and the adoption of third party integrations, security has become more challenging to uphold.

Here, the focus is discussed on the primary aspects of modern web application security, together with the current and advanced methods of web authentication and authorization, including OAuth 2.0 with client authentication flow, and advanced web authentication, including WebAuthn, aswell as on the web security frameworks like on Zero Trust Architecture [6], [7]. It also elaborates various threats, risks, and other problem areas, and solutions linked with web application security in the several contexts. Recent advancements of web applications across different sectors have revolutionized people's engagement, trade, and communication [8]. It starts from the banking systems to the e-commerce segment; Web applications are now become the norm in operations and usability. But as it has become apparent they have become more popular they have become more attractive to hackers as threats have evolved and become more dangerous. Examples of successful attacks include the leakage of millions of user credentials due to unsecured web applications and the use of ransomware to target insecure web APIs.

One of the most important reasons for the current research is the dynamic nature of threats. APTs, bot attacks, and usage of loophole in the integrated services reveal that traditional security mechanisms are not suitable for changing world scenarios [9]. The traditional security concepts that are generally top-down and best known as negative and ad hoc security methods are no longer adequate. Also the growing focus on users' personal data due to the increased legal requirements such as GDPR and CCPA increases the demand for safe and transparent web applications [10]. Organizations have to safeguard identifiable data of users as well as prove to be legal and ethical compliant as per modern trends. Both the requirement of high security and the demand for the privacy of data make the necessity of adopting trending frameworks and technologies, for instance, OAuth 2.0, Zero Trust, and WebAuthn even higher.

CORE TECHNOLOGIES

Web application security in contemporary society em- ploys a variety of progressive technologies and architectures that facilitate secure and seamless user application and data interaction. In this part, the author discusses the fundamental technologies which are the foundation of practical techniques of WASE, including authentication, authorization, secure communication, as well as innovations.

A. *AUTHENTICATION AND AUTHORIZATION STANDARDS*

1) OAuth 2.0 and OpenID Connect

OAuth 2.0 has been highly popular in implementing token- based Single sign-on in web applications [11]. It allows ap- plications to get user information without having to provide these unique sensitive details, by the use of features like access tokens, refresh tokens, and scopes. OpenID Connect is built upon OAuth 2.0 with the inclusion of identity that enables Singe Sign On (SSO) and Identity Verification. Key features of OAuth 2.0 are:

- Access Tokens: Allow secure access to user resources for a temporary time
- Refresh Tokens: Enable seamless authentication without requiring user input.
- Authorization Flow: Tailored mechanism to suit diverse application needs.

2) WebAuth

WebAuthn, a new passwordless authentication standard, drives a stake thru the heart of password-type

threats through public-key cryptography [7]. Users can log in using biometric information or HW security keys which make the service relatively secure from phishing and CSA threats.
Table 1 shows the comparison between authenticating mechanisms.

B. *SECURE COMMUNICATION*

1) SSL/TLS Encryption

Confidentiality means that transferred data between applica- tions' users and servers is not rendered readable and interven- able by third parties. Specifically, SSL/TLS is responsible for countering some security threats such as eavesdropping and man-in-the-middle attack (MITMA), as well as plagiarism of data [12]. Use of HTTPS across web applications is a sure way of protecting sensitive information like login credentials as well as financial information.

TABLE 1: Comparison of Authentication Mechanisms

Mechanism	**Description**	**Advantages**	**Challenges**
Password-based	Users authenticate with a password.	Simple to im- plement	Prone to phish- ing and reuse issues
Multi-Factor Authentication (MFA)	Combines multiple verification factors, such as passwords and biometrics.	Enhanced security	Can impact us- ability
OAuth 2.0	Token-based authentication for apps.	Secure, scalable, user-friendly	Requires proper configuration
WebAuthn (Passwordless)	Uses biometrics or security keys for login.	Highly secure, phishing- resistant	Requires device compatibility

Content Security Policy (CSP)

CSP is one type of web security standard that protects against threats like XSS for example by defining which resources (for example scripts, styles) a browser is permitted to load. CSP works by listing sources from which scripts can be downloaded, thereby preventing the use of code injected into web pages [13].

C. *WEB SECURITY PROTOCOLS*

1) Cross-Origin Resource Sharing (CORS)

CORS ensure cross origin request security by enabling a web application to maintain a list of the domains that are allowed to access its resources [14]. Correct CORS headers minimizes the vulnerability of cross-site forgery and data leakage while preserving all the conveniences that user has.

2) SameSite Cookies

SameSite cookies limit the ways in which cookies can be used in cross-site requests in an effort to combat cross-site request forgeries (CSRF) [15]. Authenticated cookies should not be sent to third parties, this is why by using cookies with SameSite attribute, the web application guarantees this safety feature.

D. *EMERGING TECHNOLOGIES*

1) Zero Trust Architecture (ZTA)

Zero Trust is a security concept that deviates from the traditional model of trusting everything inside a network and seeks to verify each user/device requesting access again and again [16]. ZTA greatly shrinks the attack area of web applications when applying principles and methods such as least privilege access, multi-factor authentication (MFA), and contextual risk analysis.

2) Runtime Application Self-Protection (RASP)

RASP technology, therefore, provides active real-time appli- cation monitoring and shelters the application from attack [17]. Unlike other security approaches, RASP works at the manner of the application and only pinpoints threats.

E. *TOKEN MANAGEMENT AND SECURITY*

To and for communication in modern applications happen through secure tokens (for instance, JSON Web Tokens or access tokens) and for session management [18]. Proper token handling includes:

- The best practice of limiting the validity period also reduces the identified risks as much as possible.
- Storing tokens in secure HTTP-only cookies.
- Granting of permissions on to tokens according to the principle of least privilege.

THREATS AND CHALLENGES

Though there have been improvements in security of web applications, there are still many risks that continue to pose high risks with a view of penetrating the application with an aim of capturing the user data, deasant the services or even vend the application. Meeting these threats requires knowledge not only of the typical and developing risks, but also of the practical obstacles in creating effective secure systems.

F. *COMMON THREATS*

1) Cross-Site Scripting (XSS)

XSS attacks happen when scripts are added into otherwise legitimate websites, and allow the attacker to gain access to data and assume the identity of other users or completely change the layout of the site. Lack of sufficient input validation and inadequate content sanitization are the primary attack enablers of XSS vulnerability [19].

2) SQL Injection

If an attacker enters SQL commands into input boxes, he can control what is displayed, updated, or destroyed in a database. This makes it one of the most severe web applica- tion vulnerabilities, even though there are ways to safeguard against it such as: prepared statements, and parameterized queries [20].

3) Cross-Site Request Forgery (CSRF)

CSRF is an attack on authenticated users where the attacker leads users to perform activities they did not intend to per- form on a trusted site. As it is shown without such verification mechanisms such as CSRF tokens, web applications are vulnerable to unauthorized state-changing requests [21].

4) Session Hijacking

This acts as a key to the accounts or resources of the actual session, and attackers can easily pilfer or forge session tokens to deceive other users. This is mainly common in the aspects of poor session management; transmitted messages are not encrypted or stored session IDs are not sufficiently secure [22].

5) Credential Stuffing

Hackers apply bots that entered the stolen credentials in various sites to check which of them work. Since users recycle their passwords, a breach in an application will frequently spread to other applications [23].

6) Insecure APIs

APIs are interfaces that provide other services with access to an application's functionality, and while they are an ac- ceptable means of providing access, they are also a potential point of vulnerability. API attacks can be because of weak authentication, disclosure of excessive information or poor rate control mechanisms [24].
Table 2 shows a comparison between the common web application threats and mitigations.

TABLE 2: Common Web Application Threats and Mitigations

Threat	**Description**	**Mitigation**
SQL Injection	Malicious	Use
	SQL queries	parameterized
	to manipulate	queries
	databases.	and ORM frameworks.
Cross-Site	Injecting	Sanitize user
Scripting	malicious	inputs and
(XSS)	scripts into web pages.	implement CSP.
CSRF	Exploiting	Use CSRF to-
	authenticated	kens and Same-
	sessions	Site cookies.
	to execute	

	unauthorized	
	actions.	
Session	Stealing or	Secure
Hijacking	manipulating	tokens with
	session tokens	HttpOnly
	to impersonate	and Secure
	users.	flags.

G. *EMERGING THREATS*

1) Automated Bot Attacks

Prominent types of attacks performed by bots are and include account takeovers, scraping, and denial-of-service (DoS). It is still quite difficult to distinguish a human user from a highly evolved artificial bot [25].

2) Supply Chain Attacks

Web applications are built on components like libraries or plugins which are developed by third parties; therefore, such applications are prone to threats. Unsanctioned dependencies have now emerged as one of the most frequently used attack vectors by attackers [26].

3) Cloud Security Risks

Web applications become distributed as they move to cloud platforms, they are then exposed to misconfigured storage, insecure APIs, and common infrastructure risks [27].

4) Zero-Day Exploits

Zero-day vulnerabilities are new and unidentified to the seller organization and hence they are unaddressed. These are areas that can be exploited by an attacker since the applications cannot defend themselves against such assaults [28].

A. *IMPLEMENTATION CHALLENGES*

5) Complexity of Modern Frameworks

Business web applications are complex due to a combination of frameworks, APIs, and third-party services that compose them. The challenge of maintaining system security when many components might be built by different teams or organizations can be overwhelming.

6) Evolving Threat Landscape

New threats and new approaches of attack appear all the time: this is why cybersecurity teams must maintain constant levels of alert. However, keeping itself abreast with the changing security measures are a challenge in terms of resource consumption.

7) Balancing Security and Usability

If security measures are rigid then the users get annoyed and even leave the website or the software. Finding this balance can be relatively easy, but at the same time, it is relatively challenging.

8) Human Error

Hackers can take advantage of misconfiguration defined by developers, administrators, or users, weak passwords, or in- secure coding resulting from a developer's negligence.

9) Resource Constraints

Small organizations do not have specialized personnel, sufficient funds, or resources that are necessary for investing in, and managing, effective security protocols.

SECURITY BEST PRACTICES

Appropriate measures should be taken for security because now a day's web applications have different types of threats and vulnerabilities. The next best practices describe a multi- layered approach that lays the groundwork for web application security that is also usable and compliant with industry standards.

H. *SECURE AUTHENTICATION AND AUTHORIZATION*

- **Implement Multi-Factor Authentication (MFA):** MFA work as an extra security measure whereby a user is needed to provide two or more factors of identification, passwords, fingerprints, one-time codes, among others [29].
- **Adopt Secure Protocols (OAuth 2.0, OpenID Connect):** Implement token-based authentication and authorization using reliable protocols such as OAuth 2. lightweight security protocol and identity layer on the web, OpenID Connect [11].

– Short-lived access tokens should be implemented along with refresh tokens to minimize risks of token misuse.

– Grant access only where it is necessary, and then set very rigid boundaries for these access rights This is known as the principle the of least privilege.

- **Enforce Strong Password Policies:** Users should be asked to come up with very strong passwords and constant password changes should be recommended. Passwords have to be also hashed and stored with such hashing algorithms as bcrypt.

I. *INPUT VALIDATION AND OUTPUT ENCODING*

- **Sanitize User Inputs:** Sanitize all data received from users to avoid contamination by attacks like the SQL injection attack and the cross-site scripting attack. to overcome the problem of client side validation, one should implement server side validation in parallel.
- **Use Parameterized Queries:** Don't let any direct String inputs to create SQL queries, use either prepared statements or use ORM libraries for building them safely.
- **Encode Output:** Encode the data before it's put onto the user interface so as to reduce the risk of XSS attacks. Apply content security policies that will help set restrictions towards other scripts which one cannot fully trust.

J. *SECURE COMMUNICATION*

- **Enforce HTTPS Everywhere:** Make all channels of communication between users and various servers to be HTTPS secured. Get SSL/TLS certificates frequently and expired certificates should be renewed.
- **Enable HSTS:** The HTTP Strict Transport Security (HSTS) policy ensures that web browsers cannot connect to the application using HTTP even if this is entered mistakenly.

K. *SESSION MANAGEMENT*

- **Secure Session Cookies:** Make cookies HttpOnly, Se- cure, and SameSite so that no other scripts have permis- sion to access the cookie, or to prevent someone from forging requests as the user (CSRF).
- **Limit Session Lifetimes:** Enforce session expiration to limit loss due to stolen session to. According to the multi-factor authentication, continue to check pass- words when undertaking sensitive processes.

L. *API SECURITY*

- **Authenticate and Authorize API Access:** API keys, OAuth tokens or JSON Web Tokens should be used for authenticating and authorizing APIs shown in Table 3.
- **Implement Rate Limiting:** Preserve APIs from us- age and denial-of-service (DoS) attacks by preventing individual clients from submitting a large number of requests.
- **Avoid Overexposure of Data:** Thus, don't reveal extra fields in API responses to avoid the probability of a data leak.

TABLE 3: Best Practices for API Security

Practice	Description
Use OAuth 2.0/JWTs	Securely authenticate API requests.
Rate Limiting	Limit the number of requests per user or IP to prevent abuse.
Input Validation	Validate inputs to prevent injection attacks.
Avoid Overexposure of Data	Return only necessary fields in API responses.

M. *ACCESS CONTROL*

- **Implement Role-Based Access Control (RBAC):** Maintain user management with particular roles where the users within a role can only perform assigned tasks. For more precise control you should use Attribute- Based Access Control (ABAC).
- **Follow the Principle of Least Privilege:** It is agreed that user and process privileges should be as

restricted as possible to carry out their activities.

N. *LOGGING AND MONITORING*

- **Enable Comprehensive Logging:** Encode incidents connected with a secure system, for example, access to the system, activation of authorization, or change of privileges. Make sure that logs do not contain sensitive data on them.
- **Monitor for Anomalies:** Secure your applications and web frameworks against intrusion by using IDS and SIEM that will alert you of any undesirable events and take necessary action at once.

O. *REGULAR SECURITY TESTING*

- **Perform Vulnerability Assessments:** Ensure frequent scans of the application and underlining structures in a bid to isolate and eliminate any possible vulnerabilities.
- **Use Penetration Testing:** Pretend to attack to find out how vulnerable the application.
- **Adopt Secure Development Practices:** Validate security (e.g., static and dynamic analysis) as a set of DevSecOps practices to be inherited in the SDLC of the software.

P. *SECURE THIRD-PARTY DEPENDENCIES*

- **Review and Update Libraries Regularly:** Depend on the dependency management tools in order to monitor and deploy the required third-party libraries to eliminate potential weaknesses.
- **Verify Supply Chain Security:** The third-party components should be inspected the field of security before being incorporated in the application.

Q. *EDUCATE DEVELOPERS AND USERS*

- **Train Developers on Secure Coding:** Remind developers about these issues periodically; Offer training on the OWASP Top 10 and other security issues.

- **Raise User Awareness:** Inform users how to identify phishing, develop secure passwords and such things, which they have to avoid.

CASE STUDIES

This section looks at various real life cases in relation to success and failure in web application security with a view of seeking to explain why strong strategies in this area must be put in place. These case studies are also relevant examples about pratical application of todays technologies and best practices to reduce risks or on the other hand how lack of security can be extremely costly.

R. *OAUTH 2.0 IN GOOGLE APIS*

1) Overview

OAuth 2.0 implemented by Google has been taken as one of the benchmarks for safe identity verification and control in Web-based systems. It enables the user to authorise third party applications to access his/her information without pro- viding authentications [30].

2) Security Measures Implemented

- OAuth 2.0 scopes give users an ability to determine what data of theirs third-party application can use. For instance, a user will allow an app to use his/her calendar while denying them permission to his/her email.
- This has the added bonuses of the access tokens having short lifetimes, making it rather difficult should the tokens be intercepted for misuse. Access tokens are used to increase tokens' lifespan without requiring the user to reauthenticate themselves again.
- Google requires applications to sign up with it to gain access to user information and help clients only with verified redirect URLs.

Google has been adopting OAuth 2.0 and millions of users have benefitted by having a safe way of authenticating their access without compromising their information and credentials. An effective token based authentication and authorization mechanism is crucial for achieving reasonable security and convenience for practical and scalable web applications.

A. *DATA BREACH AT EQUIFAX (2017)*

3) Overview

Another big data mishap was the Equifax breach a credit reporting agency that saw the personal data of 147 million users including their SSN and financial records revealed to hackers [31].

4) Security Failures

- The breach targeted a well-documented exploitable weakness CVE-2017-5638 in Apache Struts, an open- source web application framework. Even with a patch released, Equifax did not install it as soon as they should have The Equifax Cybersecurity Incident and Its Impact on Consumers.

B. *IMPLEMENTATION OF WEBAUTHN BY MICROSOFT*

5) Overview

Microsoft also incorporated passwordless sign-in technology known as WebAuthn within its apps to boost not only user security but also comfort [32].

6) Security Features Implemented

- The users can log in using fingerprint or face recognition or through hardware security keys and thus mitigate risk posed by weak or reused passwords.
- WebAuthn operates based on public-key cryptography to make sure that no user data, for example passwords are stored in the server.
- Hardware based authentication does not allow the at- tackers to easily phish for users credentials.

WebAuthn put a great dampener to the phishing and the credential stuffing that was majorly prevalent among the Microsoft users as it equally offered both the facilities, En- hanced Security amongst the users. Passwordless authentica- tion is a much stronger form of protection than the traditional password and can improve the customers' experience in the application substantially if the phishing threat is a factor.

C. *CAPITAL ONE DATA BREACH (2019)*

7) Overview

In 2019, Capital One left a misconfigured web application firewall (WAF) that resulted in the leaked of over 100 million customer records [33].

8) Security Failures

- A simple misconfiguration in the WAF let the attackers gain access to other buckets containing important information in Amazon S3.
- Those who are over-sharing data ended up with criminals receiving much more information than they needed to run their organizations.

It led to massive penalties, legal issues, and brands' loss of reputation. It pointed to the need to constantly harden cloud services and also to gain control over the rights of entrance. Governance in cloud environments needs a proper configuration management, audited and strict adherence to the principle of least privilege to reduce likelihood of occurrences of breaches.

D. *GITHUB'S BUG BOUNTY PROGRAM*

9) Overview

Similarly, GitHub has an open known vulnerability program or bug-bounty-program to encourage security research to expose holes in the platform [34].

10) Security Measures Implemented

- Security researchers perform hacker like activity on GitHub to determine its weak points and report those to the firm.
- This model of approach to the disclosure of vulnerabilities in GitHub helps the firm to cultivate trust both from the users and security experts.

GitHub has long been hosting bug bounty and has effectively managed to find and eliminate multiple types of flaws, many of which have given the web resource a reputation as a secure platform for developers. Actively involving the security sector through bug bounty programs is beneficial as it help improve an application's security as well as build trust among users.

Altogether these scenarios demonstrate the need for implementing best security solutions, including advanced authorization protocols like OAuth 2.0 and newcomers like WebAuthn, as well as ensuring trivial but crucial items like software patches and system configurations, described in Table 4. They also draw attention to the need to minimize risk, and experience subsequent successes and failures that would enable enhancement of web application security.

TABLE 4: Summary of Case Studies

Case Study	Success/Failure	Key Lesson
Google OAuth 2.0	Success	Token-based security is scalable and user-friendly.
Equifax Data Breach	Failure	Timely patching and proper segmentation are crucial.
Microsoft WebAuthn	Success	Passwordless authentication improves security and user experience.
Capital One Data Breach	Failure	Misconfiguration can compromise sensitive data.
GitHub Bug Bounty	Success	Proactive testing strengthens overall security.

CONCLUSION

As a sub-field of Web application security is still emergent and growing rapidly to address the growing threats and complex web application architectures. This paper has pointed out that strong security is derived from traditional standards like OAuth 2.0 and protocols like SSL/TLS, and new standards including WebAuthn and the Zero Trust Architecture. When applied and done properly, these measures are key components in the construction of safe and vigorously dependable web applications.

However, there are still issues that make a web application insecure, including but not limited to, SQL injection, Cross- Site Request Forgery (CSRF), etc., Supply chain attacks, and zero-day vulnerabilities. Exposure of these risks needs a dedicated security approach such as input validation, se- cure communication, testing, and monitoring. The examples described in the paper show that there is much to be learned from memorial success and failure experiences. They stress the importance of timely update, secure configuration, and mode of authentication in avoiding and reducing the effects of the breach.

In conclusion, the design of secure web applications invites the cooperation of the developers, organizations, and the security fraternity. With the efficient awareness of new and emerging threats and comprehensive multi-tiered approach, the stakeholders safeguard USD, uphold users' confidence, and guarantee the sustainability of the respective applications. While the threats and risks persist, establishing new trends toward security make the necessary steps for better understanding the threats and risks in digital processes.

REFERENCES

[1] A. Salamzadeh, P. Ebrahimi, M. Soleimani, and M. Fekete-Farkas, "Gro- cery apps and consumer purchase behavior: application of gaussian mix- ture model and multi-layer perceptron algorithm," Journal of Risk and Financial Management, vol. 15, no. 10, p. 424, 2022.

[2] M. Attaran, "Blockchain technology in healthcare: Challenges and oppor- tunities," International Journal of Healthcare Management, vol. 15, no. 1,
pp. 70–83, 2022.

[3] Z. Mousavi, C. Islam, M. A. Babar, A. Abuadbba, and K. Moore, "De- tecting misuses of security

apis: A systematic review," arXiv preprint arXiv:2306.08869, 2023.
[4] A. Hoffman, Web application security. " O'Reilly Media, Inc.", 2024.
[5] S. S. Nair, "Securing against advanced cyber threats: A comprehensive guide to phishing, xss, and sql injection defense," Journal of Computer Science and Technology Studies, vol. 6, no. 1, pp. 76–93, 2024.
[6] J. Singh and N. K. Chaudhary, "Oauth 2.0: Architectural design aug- mentation for mitigation of common security vulnerabilities," Journal of Information Security and Applications, vol. 65, p. 103091, 2022.
[7] N. Bindel, C. Cremers, and M. Zhao, "Fido2, ctap 2.1, and webauthn 2: Provable security and post-quantum instantiation," in 2023 IEEE Sympo- sium on Security and Privacy (SP), pp. 1471–1490, IEEE, 2023.
[8] S. Wan, H. Lin, W. Gan, J. Chen, and S. Y. Philip, "Web3: The next internet revolution," IEEE Internet of Things Journal, vol. 11, no. 21, pp. 34811– 34825, 2024.
[9] J.-P. A. Yaacoub, H. N. Noura, O. Salman, and A. Chehab, "Robotics cyber security: Vulnerabilities, attacks, countermeasures, and recommen- dations," International Journal of Information Security, vol. 21, no. 1,
pp. 115–158, 2022.
[10] R. Y. Wong, A. Chong, and R. C. Aspegren, "Privacy legislation as business risks: How gdpr and ccpa are represented in technology com- panies' investment risk disclosures," Proceedings of the ACM on Human- Computer Interaction, vol. 7, no. CSCW1, pp. 1–26, 2023.
[11] S. Dashti, A. Sharif, R. Carbone, and S. Ranise, "Automated risk assess- ment and what-if analysis of openid connect and oauth 2.0 deployments," in IFIP Annual Conference on Data and Applications Security and Privacy,
pp. 325–337, Springer, 2021.
[12] D. D. Kumar, J. D. Mukharzee, C. V. D. Reddy, and S. M. Rajagopal, "Safe and secure communication using ssl/tls," in 2024 International Conference on Emerging Smart Computing and Informatics (ESCI), pp. 1–6, IEEE, 2024.
[13] A. Koohang, J. H. Nord, Z. V. Sandoval, and J. Paliszkiewicz, "Reliability, validity, and strength of a unified model for information security policy compliance," Journal of Computer Information Systems, vol. 61, no. 2,
pp. 99–107, 2021.
[14] M. Golinelli, E. Arshad, D. Kashchuk, and B. Crispo, "Mind the cors," in 2023 5th IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA), pp. 213–221, IEEE, 2023.
[15] M. Squarcina, P. Adão, L. Veronese, and M. Maffei, "Cookie crumbles: breaking and fixing web session integrity," in 32nd USENIX Security Symposium (USENIX Security 23), pp. 5539–5556, 2023.
[16] E. B. Fernandez and A. Brazhuk, "A critical analysis of zero trust archi- tecture (zta)," Computer Standards & Interfaces, vol. 89, p. 103832, 2024.
[17] P. Le-Thanh, T. Le-Anh, and Q. Le-Trung, "Research and development of a smart solution for runtime web application self-protection," in Proceedings of the 12th International Symposium on Information and Communication Technology, pp. 304–311, 2023.

[18] Y.-S. Yang, S.-H. Lee, W.-C. Chen, C.-S. Yang, Y.-M. Huang, and T.-
W. Hou, "Securing scada energy management system under ddos attacks using token verification approach," Applied Sciences, vol. 12, no. 1, p. 530, 2022.

[19] M. Alsaffar, S. Aljaloud, B. A. Mohammed, Z. G. Al-Mekhlafi, T. S. Almurayziq, G. Alshammari, and A. Alshammari, "Detection of web cross-site scripting (xss) attacks," Electronics, vol. 11, no. 14, p. 2212, 2022.

[20] M. Nasereddin, A. ALKhamaiseh, M. Qasaimeh, and R. Al-Qassas, "A systematic review of detection and prevention techniques of sql injection attacks," Information Security Journal: A Global Perspective, vol. 32, no. 4, pp. 252–265, 2023.

[21] A. Khade, J. Iyer, M. Inbarajan, and V. Yadav, "Mitigating cross-site request forgery threats in the web," in 2023 7th International Conference on Trends in Electronics and Informatics (ICOEI), pp. 695–698, IEEE, 2023.

[22] W. S. Hwang, J. G. Shon, and J. S. Park, "Web session hijacking defense technique using user information," Human-centric Computing and Infor- mation Sciences, vol. 12, p. 16, 2022.

[23] S. Vinberg, J. Overson, A. C. B. D. Woods, S. Ghosemajumder, S. Boddy,
R. Pompon, and A. Koritz, "2021 credential stuffing report," 2021.

[24] F. A. Qazi, Insecure Application Programming Interfaces (APIs) in Zero- Trust Networks. Capitol Technology University, 2022.

[25] S. Shaikh, C. Rupa, G. Srivastava, and T. R. Gadekallu, "Botnet attack intrusion detection in iot enabled automated guided vehicles," in 2022 IEEE International Conference on Big Data (Big Data), pp. 6332–6336, IEEE, 2022.

[26] A. Coufalíková, I. Klaban, and T. Šlajs, "Complex strategy against supply chain attacks," in 2021 International Conference on Military Technologies (ICMT), pp. 1–5, IEEE, 2021.

[27] U. A. Butt, R. Amin, M. Mehmood, H. Aldabbas, M. T. Alharbi, and
N. Albaqami, "Cloud security threats and solutions: A survey," Wireless Personal Communications, vol. 128, no. 1, pp. 387–413, 2023.

[28] A. Waheed, B. Seegolam, M. F. Jowaheer, C. L. X. Sze, E. T. F. Hua, and S. R. Sindiramutty, "Zero-day exploits in cybersecurity: Case studies and countermeasure," Preprints [Preprint]. https://doi. org/10.20944/preprints202407, vol. 2338, p. v1, 2024.

[29] B. O. ALSaleem and A. I. Alshoshan, "Multi-factor authentication to sys- tems login," in 2021 National Computing Colleges Conference (NCCC),
pp. 1–4, IEEE, 2021.

[30] E. Ylli, I. Tafa, and E. Gjergji, "Oauth 2.0 in securing apis," International Jou rnal of Research In Commerce and Management Studies (ISSN: 2582- 2292), vol. 3, no. 1, pp. 10–19, 2021.

[31] J. Craig and J. Kovacic, "Equifax data breach 2017," 2024.

[32] I.-E. Wu, "Authentication in web applications," 2023.

[33] O. O. Olaniyi, O. O. Olaoye, and O. J. Okunleye, "Effects of information governance (ig) on profitability in the nigerian banking sector," Asian Journal of Economics, Business and Accounting, vol. 23, no. 18, pp. 22– 35, 2023.

[34] J. Ayala, Y.-J. Tung, and J. Garcia, "Poster: A glimpse of vulnerability disclosure behaviors and practices using github projects," in Proceedings of the 45th IEEE Symposium on Security and Privacy (S&P), 2024.