International Journal of Advanced Engineering and Management Research

Vol. 10, No. 06; 2025

ISSN: 2456-3676

Developing a User Interface for the Security Vulnerability Detection in E-commerce Web Application

Seun Mayowa Sunday
ADEY Innovations Limited,
Stone House, GL10 3EZ, Gloucester, United Kingdom

doi.org/10.51505/ijaemr.2025.1521 URL: http://dx.doi.org/10.51505/ijaemr.2025.1521

Received: Nov 22, 2025 Accepted: Dec 02, 2025 Online Published: Dec 15, 2025

Abstract

The growing number of cyberattacks, especially SQLi vulnerabilities, emphasizes the need for easily available and efficient cybersecurity solutions, particularly for e-commerce systems that manage sensitive data. The aim of this project is to create an easy-to-use tool that enables non-technical users to identify and remediate SQLi vulnerabilities in web applications.

Using a structured methodology, the project integrated cutting-edge tools, including Django for system architecture, SQLMap for automated vulnerability scanning, and Celery for task management, ensuring seamless operation and usability. To validate the correctness and effectiveness of the solution, thorough testing was conducted on several web applications.

The findings revealed that the system reliably detected high-severity vulnerabilities, with actionable recommendations to mitigate risks. Key mitigation strategies include input validation, parameterised queries, and integration of Web Application Firewalls (WAFs), supported by regular security audits and adherence to the principle of least privilege. The project demonstrated how a multi-layered defence strategy not only guards against SQLi attacks but also promotes inclusivity by allowing non-technical individuals to preserve their digital assets. While effective, the study highlights areas for future research, such as expanding the scope to address other web vulnerabilities like cross-site scripting (XSS).

Keywords: SQL injection, cybersecurity, e-commerce platforms, vulnerability scanning, mitigation strategies, secure development, multi-layered defence.

Introduction

Web applications chronicle is a proof to human cleverness and adaptability, reflecting advancement in technology and change in society (John, 2024). As technology advances continually, so do web applications. Thus, the architecture surrounding web apps get more complicated day by day (Uptrends, 2020). This prompted various web applications maintenance teams to ensure growth beyond a handful of developers over the year to intelligent, skilled and highly specialized teams. Developers constantly make adjustments to the code and content, and their changes are published to the live site instantly in some cases (Uptrends, 2020).

According to Vijay (2024), web application development has changed the way humans use the internet from its inception to the present day. Most early websites were entirely immobile, only providing content to users but having no real interactive elements. Enter web applications, which are the cornerstone of our digital life, offering experiences that are dynamic and immersive, and compete with traditional desktop applications. This has been fueled by innovation in technology and development practices, leading to a new era -where developers design software differently and users engage with it differently. (Vijay, 2024).

These are some technologies used by internet apps: a business needs to transfer companies and supply products remotely, an organization has to communicate with customers confidentially and efficiently through Web Apps (Kashevnik et al., 2020). With the development of web technology, it has vastly contributed to huge growth in regular quantity and utilization of Web-based applications such as robots for communicating, intelligent assistants, and engines of suggestion within online shopping across different platforms, such as Instagram, Twitter, message boards, among many other services (Ding et al., 2020). Gadgets are a part of our daily and personal lives, so during this time, online web applications have become the target of hackers instead of computers.

In cases, web applications are made up of parts that work together (Intellect Soft, 2024). Web application structure is usually categorized into two groups. User interface and structural web components, with client-side and server side components falling under the group. Having components can make it challenging to grasp the system with just descriptions alone. A web application architecture diagram, as shown in Figure 1.1, is useful as it visually shows the components and how they interact with each other.

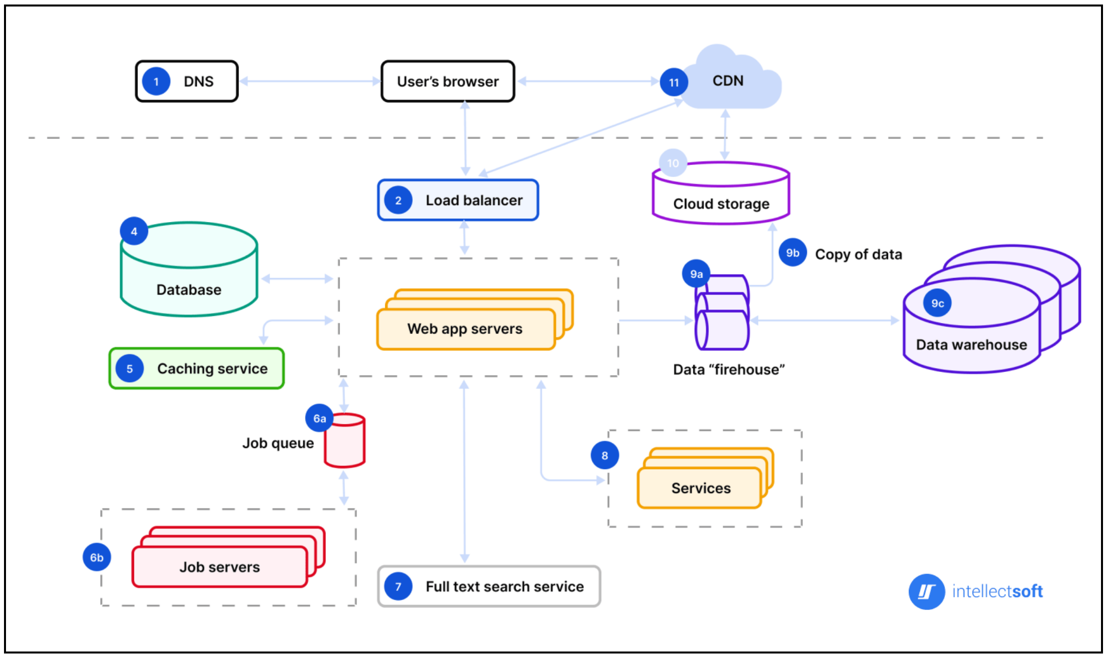


Figure 1.1: Web Applications Architecture (Intellect Soft, 2024).

The layout of web applications plays a role in how different applications communicate with each other and with the underlying software databases and systems to ensure operation when multiple applications are running simultaneously. When you type in a URL and click "Transition" the browser initiates a process to find and request the page from the web server facing the Internet. Once the server responds to this request and sends files back to the browser for processing, these files are then interpreted by the browser to show the requested page to the user effectively (Hammoudeh, 2021).

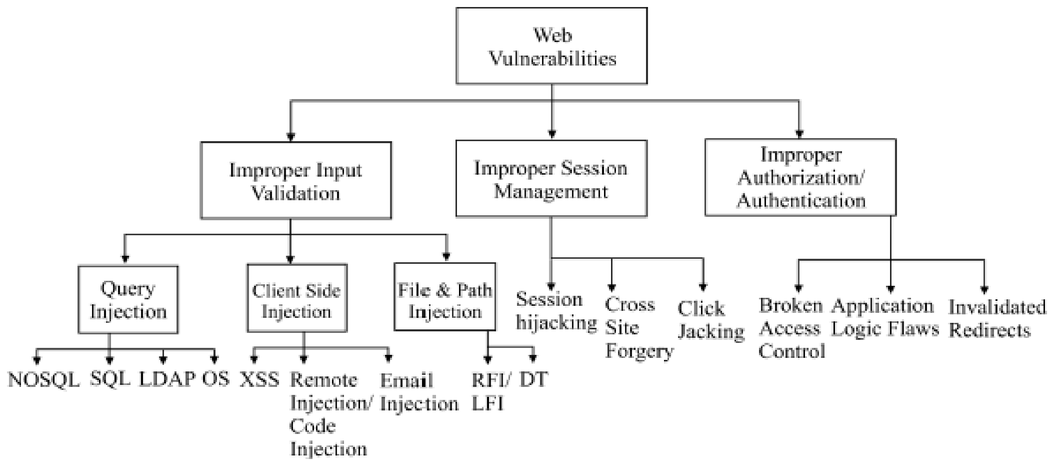
1.1 Background

Web Applications are no longer just an indispensable tool for businesses but also for day lives. The Software-as-a-Service (SaaS) market hit $167 billion in 2022 (Gartner, 2022). As a result, web applications have become a lucrative target for hackers, and this must be treated the same as other properties that change over time depending on how long it has been used. Attackers can infiltrate systems through the use of unpatched vulnerabilities and zero-day exploits, allowing them to steal payment details, passwords, personal information or a range of other sensitive information. Continuing, this means the need to discover vulnerabilities before finding them in production code that will be live on the internet has become increasingly crucial (Sakaoglu, 2023).

With the popularity of the internet and web applications, this has (and continues to be) required by pretty much every web application as security is crucial from a cyberattack aspect to ensuring that information is safe. Web applications are vulnerable to various logical and technical vulnerabilities (Altulaihan et al., 2023). Structured Query Language (SQL) injection, Cross-Site Scripting (XSS), and Remote and Local File Inclusion are some of the technical vulnerabilities. These vulnerabilities impair the security of web applications, and because of improper programming or outdated systems, these bugs appear (Altulaihan et al., 2023).

Vulnerabilities in web applications continue to persist, as developers remain committed to creating new web applications, thereby increasing the likelihood of web application vulnerabilities (Erik et al., 2023). Such web vulnerabilities, SQL injections in particular, can be disastrous to the developers of the web application(s), the company that runs these web applications, and end-users who log in or leave their data on each site-specific website application (Erik et al., 2023).

The term vulnerability is a flaw, bug or escape in web application development. Exploit: when a vulnerable person is exploited and an attack is successful. Based on the research work (Noman et al., 2020), there are three types of Web application vulnerabilities: Lack of input validation, meaning the failure to properly check and sanitize user-requested data. Examples of web attacks, such as SQL injection or Cross-Site Scripting (XSS), include the exploitation of improper input validation vulnerabilities. Improper session management: the web session is not adequately secured and, for this reason, the application often cannot recognize whether an incoming web request is harmful at all, except it could be tied to a genuine, valid session identifier. Improper session management examples include Cross-Site Request Forgery (CSRF) and Session hijacking, which are other common web attacks. Additionally, the vulnerability of improper authorization and authentication indicates a logic bug within the access control policies as well as authentication functions (Noman et al., 2020). Vulnerabilities can be classified into three main categories, namely: Improper Input Validation, Improper Session Management, and Improper Authorization/Authenticity (Noman et al., 2020). Figure 1.2 explains web vulnerabilities diagrammatically.



LDAP: Lightweight Directory Access Protocol
OS: Operating Systems
RFI: Remote File Inclusion
LFI: Local File Inclusion
DT: Directory Traversal

Figure 1.2: Types of web vulnerabilities (Noman et al., 2020).

According to SiteLock (2024), there are 22 attacks per day on average across websites. There are numerous attacks, exceeding 8,000 per year per website. In non-technical terms, website vulnerability is a weakness or misconfiguration in a web application code that allows an attacker to gain some level of control over the site -and potentially the hosting server. These kinds of vulnerabilities are often exploited by automated scripts, such as vulnerability scanners and botnets. Cybercriminals use customized tools to crawl the internet across all types of platforms, including WordPress, scouring for common and documented vulnerabilities (Aarushi et al., 2021).

Similarly, Relevant Software (2024) stated the top three security breaches, which include: SQL, Predictable resource location, and code injections. These make up 64% of all attacks on APIs and web apps. Meanwhile, there was a 33% reduction of DDoS attacks on web apps in 2023 compared to 2022. However, there was a 500% exponential increase in the frequency of hostile web app transactions. Therefore, there is now more focus on online apps and their infrastructure by the attackers, in which DDoS attacks are rapidly becoming the most complicated and complex attacks of web applications.

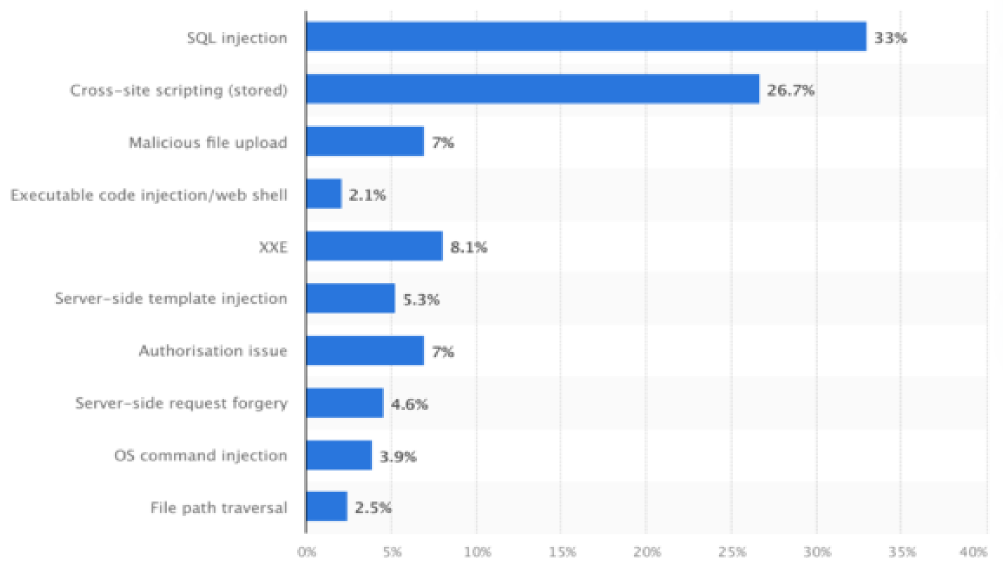


Figure 1.3: Distribution of critical web application vulnerabilities worldwide as of 2022 (Relevant Software, 2024).

To allay these shortcomings, it is of great importance to utilize web application security assessment. This security assessment process enhances vulnerabilities in web applications to be detected and mitigated in order to protect vital information and ensure balance in organizational security posture (Anjunaira, 2024). This systematic security assessment approach shown in Figure 1.4 provides a comprehensive overview to considerably areas in conducting web application security assessment effectively (Anjunaira, 2024).

The systematic approaches include the following respectively: The first approach is to define objectives and scope in order to identify the specific web applications to be assessed, outlining the goals of the assessment. Followed by gathering information to identify the technologies used, understanding the application architecture, and compiling a list of assets such as third-party integrations, APIs, and URLs. The third step is Threat Modelling to identify potential threats and vulnerabilities specific to the web application. Planning and Resource Allocation is the next steps to ensure that the assessment is conducted systematically and efficiently. Followed by Reconnaissance in order to gather additional information about the web application, including identifying entry points, server details, and potential vulnerabilities, Vulnerability Scanning to identify common security issues, Manual Testing for uncovering complex vulnerabilities and validating the results of automated scans, Analysis and Reporting in a clear, actionable, and accessible term to both technical and non-technical stakeholders, Remediation to provide guidance and support and Validation and Retesting to confirm that the security measures are effective and that no new vulnerabilities have been introduced during the remediation process (Anjunaira, 2024).

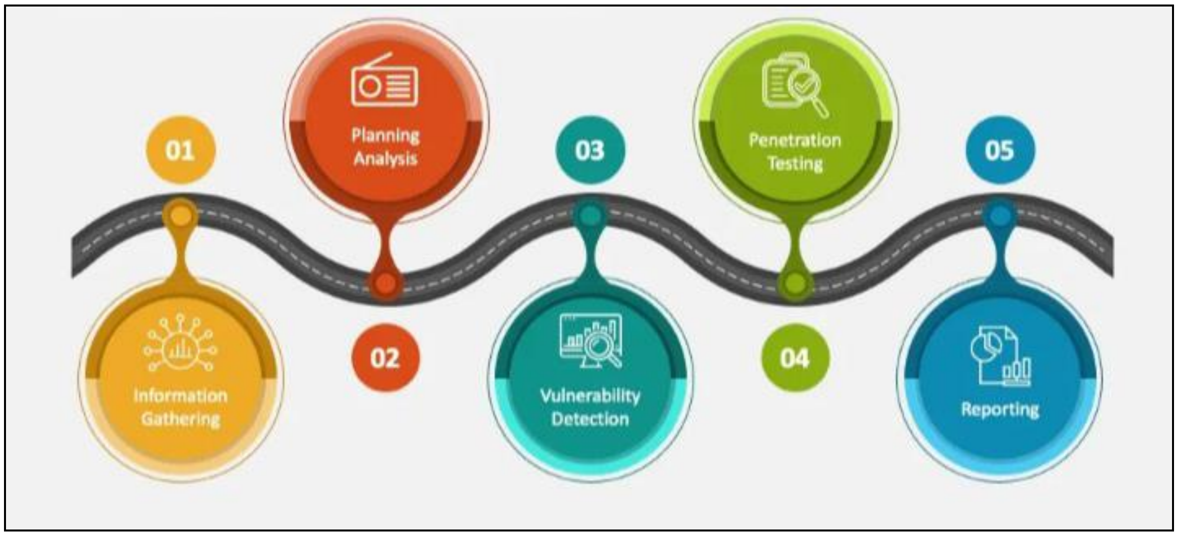


Figure 1.4: Web application security testing (Anjunaira, 2024).

1.2 Aims

The primary aim is to systematically develop a user interface which will help to identify and scan common security vulnerabilities in web applications.

1.3 Objectives

1. To create a user interface that enables non-technical users to easily perform vulnerability testing and identify potential security threats in e-commerce web applications.
2. To investigate common security vulnerabilities that are specific to e-commerce web applications, such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).
3. To conduct a detailed literature review for the purpose of understanding the potential consequences of security vulnerabilities in e-commerce web applications, including financial losses, reputational damage, and compromised customer data.
4. To propose effective mitigation strategies to improve the security of e-commerce web applications, including recommendations for vulnerability scanning and incident response planning.

1.4 Research Questions

1. How may a user interface be built to let non-technical users efficiently conduct vulnerability testing on e-commerce online applications?
2. What automated techniques and tools can be used to identify common security vulnerabilities in e-commerce web applications, and how can they be integrated into a vulnerability detection tool?
3. How can the user interface of a vulnerability detection tool be designed to effectively communicate complex security concepts to non-technical users, and what features can be included to facilitate broader participation in web application security?

2. Literature Review

According to Choon et al. (2024), e-commerce websites are essential in facilitating online transactions and offering a convenient shopping experience. However, these e-commerce websites need to be secured more than anything else. With the growing incidence of cyber threats and data breaches, the necessity for strong security measures on e-commerce websites is highlighted further. Effective security safeguards protect sensitive user information, including personal details and payment information, from being accessed or stolen by unauthorized persons. Strategies such as encryption, secure payment gateways, and authentication mechanisms for users contribute to making online transaction spaces safe and sound. (Choon et al., 2024). An e-commerce site that emphasizes security will be able to build trust with customers, reduce risks, and maintain the proper integrity of their platforms, thus creating a safe and reliable online shopping environment. Therefore, we aim to create an e-commerce site that will be less susceptible to cyber-attacks and easing for customers to have great shopping experiences.

As Gupta et al. (2023) point out, e-commerce transactions are very sensitive; thus, the issue of security is paramount. Only by assuring effective cyber-security and trust will businesses be able to attract and retain customers. Technologies such as encryption protocols and secure payment gateways play an important role in protecting personal and financial information (Kumbhakar et al., 2023 and Nyangaresi, 2023). Collection and analysis of huge data through advanced analytics play a crucial role in customizing marketing approaches, enhancing customer experiences, and streamlining supply chain management. Though e-commerce is equipped with several advantages, it also faces challenges such as cybersecurity threats, logistical issues, and regulatory compliance (Andreianu, 2023). To effectively maintain their victory over the market, e-commerce companies must provide seamless supply chains while overcoming these obstacles. With mobile commerce, social commerce, and new technologies like artificial intelligence that will continue to influence the development of e-commerce so that more opportunities and challenges will be encountered by businesses and consumers alike in the future (Prasad, 2023). Supply chains need to be managed efficiently with some form of inventory control and order fulfilment so that customers can be satisfied. The laws governing digital transactions, data privacy, and consumer protection make the environment a bit more complicated for firms to operate in. Further development of technology will indeed create a wonderful future for e-commerce, wherein shopping experiences will be greatly enhanced by innovations like augmented reality and artificial intelligence. All in all, however, e-commerce is a revolutionary force that molds not only the present but also the future of global trade (Prasad, 2023).

The e-commerce website has a maximum challenge to security in this digitally evolving scenario. Cyber threats are increasing in frequency as well as sophistication; therefore, the current issue revolves around safeguarding user information and verifying the authenticity of online transactions (Choon et al., 2024). E-commerce sites hold huge volumes of confidential data, including personal information and payment credentials, so these sites become lucrative targets for cybercriminals. Compromise of website security paves the way for identity theft, financial fraud, and reputational damage to both businesses and customers. Similarly, constant vigilance and adaptation of security measures to stay one step ahead is required in the evolving nature of cyber threats (Shyaa, 2023).

According to Goutam and Tiwa (2019), information security has grown to be pertinent for all companies as there have been increased incidences of cyber hacking in the present digital era. Regardless of the magnitude of their establishments, every business has the unavoidable obligation of protecting their data from potential assailants. The role of web applications focuses on communication, resource sharing, social interactions, online transactions, e-commerce and many others. Cybercriminals exploit weaknesses in web applications for the purpose of obtaining sensitive information without proper authorization, thereby endangering the privacy and security of the users. (Goutam and Tiwa, 2019). A firm which specializes in creating web-based applications possesses a majority of these capabilities. So, these basically constitute minimum requirements for the attacker to launch his/her attack on that firm. It is for this reason that web applications are the main targets for the attackers. If the attacker was successful in hacking the web application, they would retrieve all the secrets or classified information that is present physically or logically in that web application. This takes the attack to the next level of the attack (Kothamasu, 2023). It can be assumed, for instance, just imagine what would happen if the credit or debit card number of an online shopping site's member is saved and the hackers attack the site, they would be able to get the whole number and illegally wire money out of the victim. The remaining section of this chapter examines detailed reviews by identifying vulnerabilities in web applications, privacy issues in e-commerce, security issues in e-commerce and reviews of authors supporting the paperwork.

2.1 Conceptual Review

E-commerce, also referred to as electronic commerce, is nothing less than the modernized form of commerce, which involves the buying and selling of commodities on the internet (Bara et. al., 2023). Over the years, however, this revolutionary idea has developed tremendously, changing the way businesses operate worldwide. The establishment of E-commerce platforms is one of the key features of E-commerce which allows business owners to promote and retail their products to millions of people. This development has crossed frontiers and made it easier for consumers to find numerous products from all over the globe (George, 2024).

According to Verbivska et al. (2023), E-commerce is a fundamental innovation in the digital world that has impacted the operation of businesses and the manner in which buyers interact with businesses. In its simplest terms, e-commerce is the process of purchasing, selling, or exchanging goods and services through the internet. The transition that has taken place from conventional trading of business locations to virtual trading has changed the landscape of international trade and consumer patterns in an absolute sense (Antonia, 2023). As shown in Figure 2.1, E-commerce consists of retail giants like Amazon offering a myriad of products to small businesses leveraging digital storefronts to reach a broader audience. The fundamental appeal of e-commerce lies in its unparalleled convenience. Consumers can browse, select, and purchase products from the comfort of their homes, breaking free from the constraints of physical store locations and operating hours (Grewal, 2021).

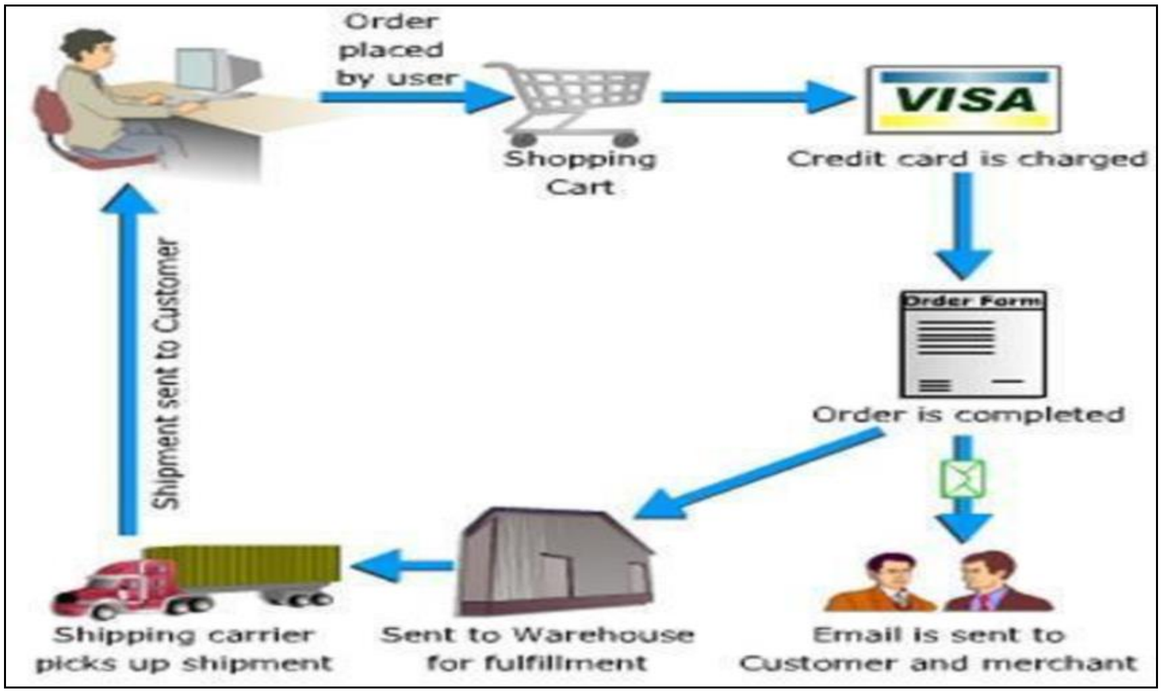

Order placed by user
Shopping Cart
VISA
Credit card is charged
Order Form
Order is completed
Email is sent to Customer and merchant
Sent to Warehouse for fulfillment
Shipping carrier picks up shipment
Shipment sent to Customer

Figure 2.1: E-commerce operations (Antonio, 2023)

The rise of e-commerce has not only revolutionized the way businesses are conducted but has also altered the way consumers behave. Thanks to technology, the consumer no longer needs to be confined to certain physical store hours as online shopping allows them to browse and make purchases at any given time (Kedah, 2023 and Lukah et al, 2023). This convenience factor, along with the ability to find many products and services, has led to the growing popularity and fast expansion of e-commerce across various sectors. Security remains an important factor in e-commerce transactions due to the significant amount of sensitive information (Alsamhi et al, 2022) that is involved in online transactions. Companies seek to use SSL as well as Two two-factor authentication methods to protect the clients' information and enhance their reliability.

2.1.1 Web Application and Vulnerabilities

Gausiya et al. (2023) pointed out that there are examples of web applications, such as forms and shopping carts, to word processors and email clients like Gmail and Hotmail amongst others. With Google Apps and Microsoft 365 being popular choices, in this category of software programs available online for users to access through their web browsers. Web services can be described as applications found across websites that are utilized to protect web servers and other online services from cyber threats originating from the Internet. This concept of safeguarding web applications is known as web application security. Involves employing methods and technologies to mitigate risks (Gausiya et al. 2023).

The term vulnerability is a flaw, bug or escape in web application development. Exploit: when a vulnerable person is exploited, and an attack is successful. Based on the research work (Noman et al., 2020), there are three types of Web application vulnerabilities: Lack of input validation Meaning, the failing to properly check and sanitize user-requested data. Examples of web attacks, such as SQL injection or Cross-Site Scripting (XSS), include exploitation of the improper input validation vulnerability. Improper session management: the web session is not adequately secured, and, for this reason, the application often cannot recognize whether an incoming web request is harmful at all, except it could be tied to a genuine, valid session identifier. Improper session management examples: Cross-Site Request Forgery (CSRF), Session hijacking being another web attack. Additionally, the vulnerability of improper authorization and authentication means there is a logic bug within the access control policies as well as authenticating functions (Noman et al., 2020). Vulnerabilities can be classified into three main categories, namely: Improper Input Validation, Improper Session Management and Improper Authorization/Authenticity (Noman et al., 2020).

2.1.2 Privacy issues in E-commerce

Privacy issues in e-commerce are of various facets, evolving primarily around the extensive collection and handling of personal information by online retailers (Youssef and Hossam, 2023). Online shopping websites often collect information such as names and addresses along with payment information which raises worries about how responsibly this data is handled. The risk of data breaches is a concern as hackers often target these sites to steal financial details which could result in identity theft and financial harm for customers (Nyangaresi, 2023). Figure 2.2 presents some of the most common threats in the e-commerce environment.

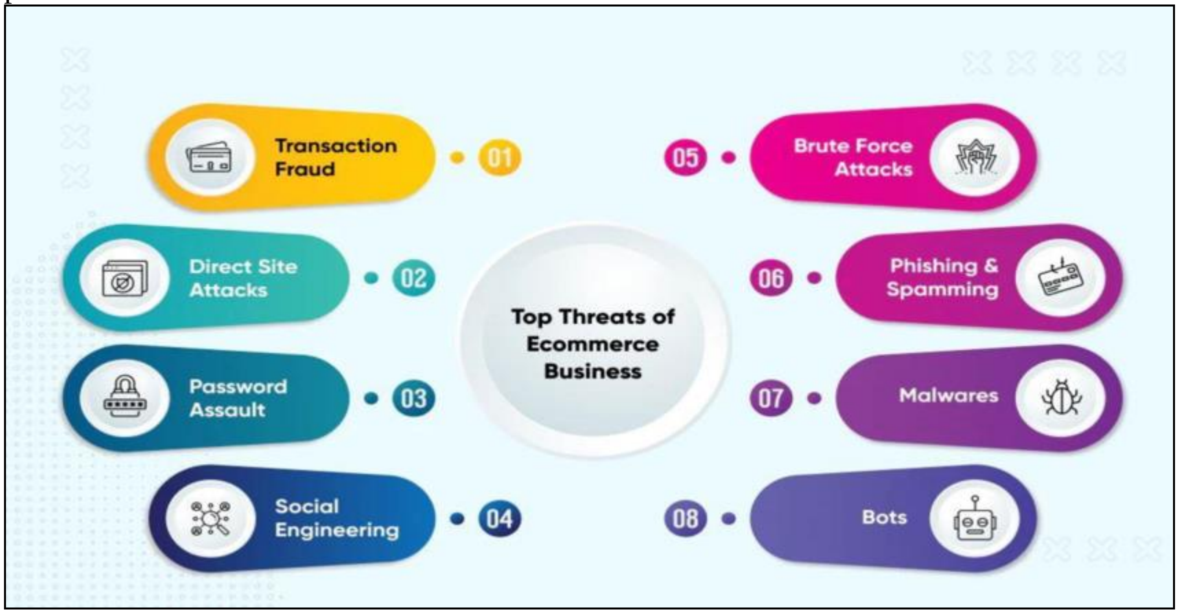


Figure 2.2: Common threats in e-commerce (Arben, 2024)

The widespread utilization of cookies and tracking technologies to analyze user behavior prompts concerns regarding the transparency of these methods since users might not have an understanding of how their actions are being observed and the purposes for which this data is employed by both the online shopping platform and external entities (Srivastava, 2023). As highlighted by Tao et al. (2023) privacy matters within the realm of e-commerce have gained importance with the surge of online shopping activities. When talking about privacy concerns, within e-commerce settings, there are points to keep in mind: the gathering and safeguard of data participation of external parties, user consent and authority, the rise of new technologies and adherence to global regulations and standards.

2.1.3 Security issues in E-commerce

The security issues of e-commerce are quite significant because they involve very sensitive data matters when dealing with online transactions. One major issue is that of data breaches, which happen frequently where the cybercriminals exploit some weaknesses in e-commerce systems to access customer information illegally (Santos et al., 2023). Such breaches expose personal information, payment details, and login credentials which further leads to terrible consequences like identity theft and financial fraud to the affected individuals. In e-commerce, a holistic approach is not only required to build but also to maintain the trust of online consumers towards constant monitoring, timely updates of security infrastructure, and education of users regarding best practices in security (Sankar et al., 2023). Figure 2.3 demonstrates how secure communication can be achieved in an e-commerce environment. According to Abu-ulbeh et al. (2023), security is a critical aspect of e-commerce, as it involves the online transaction of sensitive information such as personal and financial data.

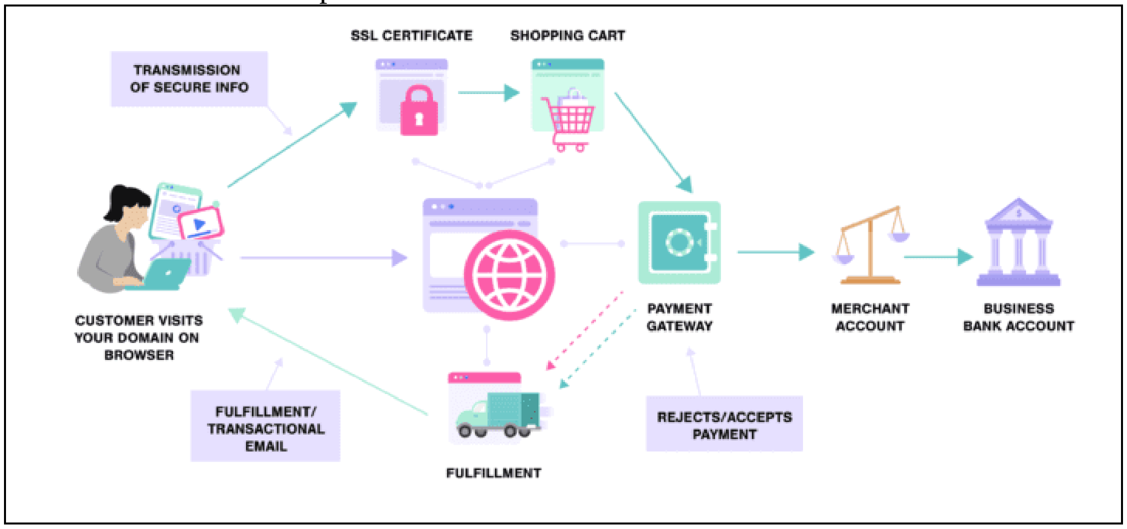


Figure 2.3: Processes of secured communication (Darren, 2024).

Various security issues can arise in the e-commerce environment, and addressing these concerns is essential to building trust among consumers. Here are some key security issues in e-commerce:

Data breaches

According to Senapati et al. (2023), data breaches in the e-commerce sector are a problem that affects businesses and consumers in equal measure. Most of these breaches involve the theft of customers' private information, such as their homes and financial records or details.

Payment Card Fraud

Beju and Fat (2023) indicate that the most common methods used in breaching credit cards' security involve taking advantage of the loopholes in online payment systems that allow them access to credit card details and later impersonate the true owners with the details obtained to commit fraud.

Phishing Attacks

To fish for sensitive information, impersonation of a reliable source is used to steal the individuals' information. As a result, some consumers and employees may be careless and enter their usernames, passwords, or any other private information, which leads to intrusion without any authorization (Carroll et al., 2022).

Denial of Service and Distributed Denial of Service attacks

These attacks enable one to use heavy traffic to a system or networks so that the system or network will become slow, or the system will be completely offline. Of concern to these two types of attacks are the availability of the e-commerce sites and the uninterrupted maintenance of the e-commerce functionalities as a result of the heavy attacks. DDoS attacks are the most critical security threat to the availability of e-commerce functionalities (Shah et al., 2022).

Insecure Payment Gateways

When these gateways are not properly protected, people with ill intentions can take advantage of the lack of security and intercept and change transaction information and gain unauthorized access to clients and their compromised financial data (Ennafiri et al., 2023).

Inadequate Authentication and Authorization

E-commerce authentication and authorization weaknesses greatly threaten e-commerce and all stakeholders, including the company and customers. Where authentication checks who the e-commerce user is, authorization checks what the user is allowed to do on the platform (Khan, Ashgar et al., 2022).

Insufficient Encryption
Encryption is one of the most important information security methods and it ensures for example, that the text a message is sent in is changed to a non-readable format while in transit to shield it from interception by any unauthorized individual (Pourrahmani, 2023).

2.1.4 User interface for vulnerability detections

The digital world is constantly changing which calls for constant change in the way websites are designed to be secured. Adopting mechanisms that secure sensitive information and prevent unwarranted access are fundamental to the provision of trusted online services. The best way to avoid web application flaws is to proactively scan the applications for flaws and fix them. Schemes in order to do that include:

Penetration Testing
Penetration testing is described as a systematic, organized, and thorough approach to conducting cybersecurity risk assessments. This involves gaining access to sensitive data, compromising IT assets, and trying out the security measures put in place. This method of testing web application security is a blend of human knowledge and intuition with the efficient use of dynamic scanning tools (Amanda, 2024).

Static Application Security Testing
SAST methods involve scrutinizing the application in much the same way that a developer goes through the code, only that they do it line by line through the use of technologies that analyze the code, the bytecode, and the binaries. Because the examined code is lucid and accessible to the tool, SAST is regarded as a white-box testing method (Nickolay, 2024).

Dynamic Application Security Testing (DAST)
Sequentially, Dynamic Application Security Testing (DAST) refers to an approach in Application Security Testing where the application is focused on in an active state to identify security flaws. And since DAST tools are blind to the application and API’s source code, they exploit the potential vulnerabilities through actual attacks, much like a real hacker would do. To a degree, DAST tools are doing penetration testing over your web applications and automating the process (Gadi, 2024).

2.2 Theoretical Review

This theoretical review section of this chapter discusses the underlying theories and several models that support the development of security vulnerability detections in web applications. There are theories and models supporting vulnerability detection on the web applications: The secure development lifecycle (SDL) defines well the application and secure coding, testing, certifying, deploying, and sustainment stages of software development (Uçak and Tuna, 2023 and Gurung et al., 2020). A process which helps keep the necessary level of system security during development and throughout its lifetime. The secure development lifecycle ensures a structured system is followed to make web applications safe. It describes a series of activities and processes that should be enforced across the entire product lifecycle, starting from requirements planning to decommissioning it. A secure development lifecycle includes involving security professionals at every step of system design and implementation. At the coding and testing stages, software compliance with security standards is verified and potential vulnerabilities are identified (Uçak and Tuna, 2023 and Gurung et al., 2020).

Similarly, (Open Worldwide Application Security Project) (OWASP, 2024); the Risk Rating methodology shows that The Top 10 website security risks for 2021 are Broken Access Control, Cryptographic Failures, Injection including Cross-site Scripting, Security Misconfiguration, Vulnerable and Outdated Components, Identification and Authentication Failures, Software and Data Integrity Failures, Security Logging and Monitoring Failures, Server-side Request Forgery (OWASP, 2024). Those significant dangers associated with web applications which the companies need to be more cautious about while developing software to keep their web applications secured are explained in detail by OWASP Foundation (OWASP, 2024).

2.3 Empirical Review

This section reviews the empirical studies and data related to the underlying theories and several models that support every basic development and evaluation of security vulnerability detections in web applications.

According to Aibekova and Selvarajah (2022), Different penetration testing attacks and the types are discussed on Kali Linux by the author in detail. It used a six-phase process, which are: Reconnaissance, Enumeration, Exploitation, Dictionary attack, Privilege Escalation and DOS Attack. During the reconnaissance phase, they scanned the target IP address and this finding identified the Open Port details in the Nmap tool and the next phase, Enumeration, they started to enumerate IP Address details like subnets, hosts, interfaces, DNS Records, Web Server details etc, using OpenNetAdmin tool. During the second phase of the assessment, they were able to assist each other in revealing more details about the web server. In the exploitation phase, the available Metasploit tool assisted them in the determination of such exploits as buffer overflow, code injection and web application. The fourth phase commenced with the use of a graphical password system wherein they were able to determine the existence of a hashed password file. Using that hashed password, they made use of six Ripper to crack it and were able to obtain the password of the target user. Now, they have proceeded to privilege escalation. In this stage, they examined each user individually; after gathering all the needed sources, they were waiting for the go-ahead to execute a DoS attack towards the target.

Devi and Kumar (2020) suggested the usage of ethical hacking in a bid to uncover weaknesses in the security of web applications. For this evaluation, they came up with a range of tools that included Nikto, OWASP's Zed attack proxy, Netcraft, Sparta and Nmap, all in the Kali Linux environment. Using OWASP's ZAP tool, the weakness was found to range from low to medium in all domains. The detected vulnerability is predominantly cross-site scripting, SSL, server leak, HTTP header, Retrieved x-powered, cookie without secure flag, URL rewriting, application error disclosure, etc. Hence, they reasoned that contained Nikto found more vulnerabilities than OWASP's ZAP tool.

Also, the investigation focused on the client and server-side attack possibilities in Kali Linux (Cisar and Pinter, 2019). The main advantage of using Kali Linux is its many built-in hacking tools, which significantly help us perform vulnerability analysis and security testing. In the context of client-side attacks, the authors discuss the need for such attacks when server-side attacks fail or when the attacker fails to obtain the proper IP address. The attackers use social engineering techniques to get important user details like names and social media account details. From those details, they start analyzing the user. It is helpful for them to understand more about the client. The most common client-side attack mentioned was the injection of Trojans into the device. Prevention measures were even suggested for the attack. Netdiscover and Zenmap were discussed for use in server-side attacks since neither requires much information. Target IP is sufficient. The tools can utilize the IP address to determine information like open ports, details about the OS, and services that are installed and running. After all this discussion, they demonstrated packet sniffing, DoS attacks, man-in-the-middle attacks, and fake access points.

Penetration testing on the Amazon Echo device was conducted using the Kali Linux operating system to exploit a denial-of-service attack, as demonstrated in the study by Anand and Singh (2021). Test assumptions placed cybercriminals already within the home network. Instead, two scenarios were set up in which DoS attack network traffic against the Amazon Echo device was monitored; one scenario performed the attack on Kali Linux and another instance for monitoring network during attack on same Kali Linux. The attacks caused a crash of the device, thereby disconnecting it from the network. This is monitored using the Wireshark tool in Kali Linux and they also exhibited packet drops by networks during attacks. All these processes led them to conclude that initiating a denial-of-service attack against Amazon Echos is quite trivial. All that attackers need to possess is access to the home network which eventually grants them access to critical information on devices interconnected within that particular network. For their demonstration, they captured all necessary information regarding the device using Nmap. Knowing the MAC address of the device, open ports were scanned using the SPARTA tool. Airodump captures basic service set identification of the router. They began attacking the target device thereafter using Metasploit Tool under Kali Linux.

1. Table of Comparison in Empirical Review

Study	**Year**	**Focus Area**	**Key Findings**	**Challenges Identified**
Aibekova and Selvarajah	2022	The Threats and Dimensions of Security Systems in Electronic Commerce	The use of penetration testing attacks to identify that phishing attack is the most common type of engineering attacks, and SQL injection and CSS as the most web applications vulnerabilities.	Rapid increase of evolving threats and limited resources were available.
Devi and Kumar	2020	Testing for security weaknesses of web applications using ethical hacking	Used ethical hacking technique to discover web applications vulnerabilities and detected security weaknesses in all domains from low to medium with OWASP's ZAP tool.	There is difficulty in detect complex vulnerabilities and expertise were limited.
Anand and Singh	2021	Penetration testing security tools	Use of penetration testing tools Nmap to identify vulnerabilities in order to strengthen network security	Penetration testing tools are too complex and there is need for proper planning and execution
Cisar and Pinter	2019	Some ethical hacking possibilities in Kali Linux environment	Client and server-side attack possibilities in Kali Linux, which significantly help to perform vulnerability analysis and security testing. This discovered SQL injection and XSS vulnerabilities	Kali Linux and other tools used were too complex. Thereby require expertise

2.4 Research Gap

The research gap found in the literature points to the need of automated, user-friendly technologies, especially aiming at e-commerce application security. Studies highlight the need to protect user information and ensure online transactions, although they may cover broad policies without paying particular attention to the tools available to non-technical users. Though these studies mostly focus on advanced testing techniques needing technical knowledge, existing research, including that by Gausiya et al. (2023) and Devi and Kumar (2020) investigates vulnerabilities, including SQL injection in online apps. Moreover, the present research mostly addresses e-commerce security problems from a theoretical standpoint, stressing vulnerabilities and their consequences instead of practical, user-centric solutions.

The difference emphasizes the need for a specialised interface enabling non-technical individuals to independently execute vulnerability testing. Creating a user-friendly tool meant to find and reduce prevalent vulnerabilities in e-commerce systems will not only improve security but also help companies in reducing financial and reputational risks connected with data leaks.

2.5 Conclusion

This project provides a more centralized detail on the rudiments and relevance of security vulnerability detections in e-commerce web applications. The e-commerce website has a maximum challenge to security in this digitally evolving scenario. Cyber threats are increasing in frequency as well as sophistication; therefore, the current issue revolves around safeguarding user information and verifying the authenticity of online transactions. This study discusses security and privacy issues in e-commerce, web applications vulnerabilities, extra some security issues in e-commerce, empirical studies and data related to the underlying theories and several models that support every basic development and evaluation of security vulnerability detections in e-commerce web applications.

3. Methods

In this section, we describe the methodology used in this study. The research used multiple security tools within a Django-based application to enhance assessments for possible SQL injection threats.

3.1 Design Philosophy and Approach

The idea is a strategy used to conduct pre-exploitation attacks for assessing incident response readiness. This design approach is driven by the current trends in cybersecuritys, which explains that SQL injection is a challenging vulnerability in the e-commerce industry (Yasmeen and Afaq, 2023), hence, it is important to develop a tool for proactive detection so that this will guide the mitigation steps.

3.2 System Architecture and Components

The architecture of this study is displayed in Figure 3.1. Input URL/IP address helps to submits a URL or IP address for scanning, this is what the end-user is capable of interacting with.
The Django framework (Django, n.d) interacts with views (which is an internal logic that ensures proper communication between the front end and back end) and models (which is an internal logic that ensures proper communication with the database) to process results and manage the entire scanning process. The Views is the fractional part of Django that interact with model and reply to request. The Model helps to give detailed description of data to be stored in the database. Asynchronous Engine handles background tasks, ensuring scans run without blocking user interactions. The result is shown in organised order. Figure 3.1 shows the steps involved from the input entry to seeing the expected result. In addition to the system architecture, a simple user

interface workflow diagram (see Figure 3.2) is used to show how users move from entering a URL to viewing the final scan results.

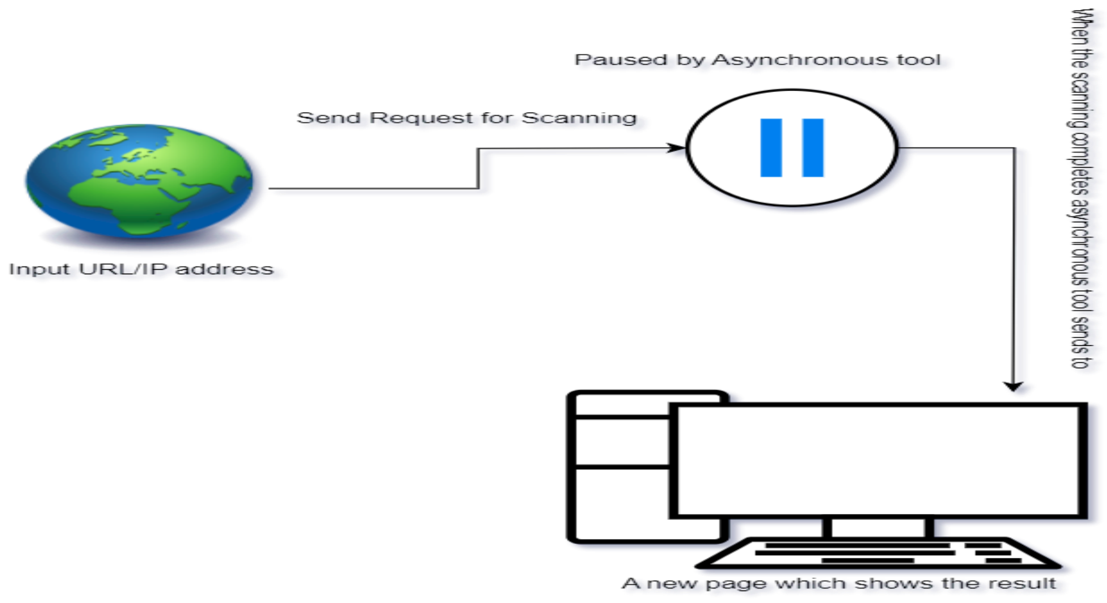

Paused by Asynchronous tool
Send Request for Scanning
Input URL/IP address
When the scanning completes asynchronous tool sends to
A new page which shows the result

Figure 3.1: System Architecture Diagram

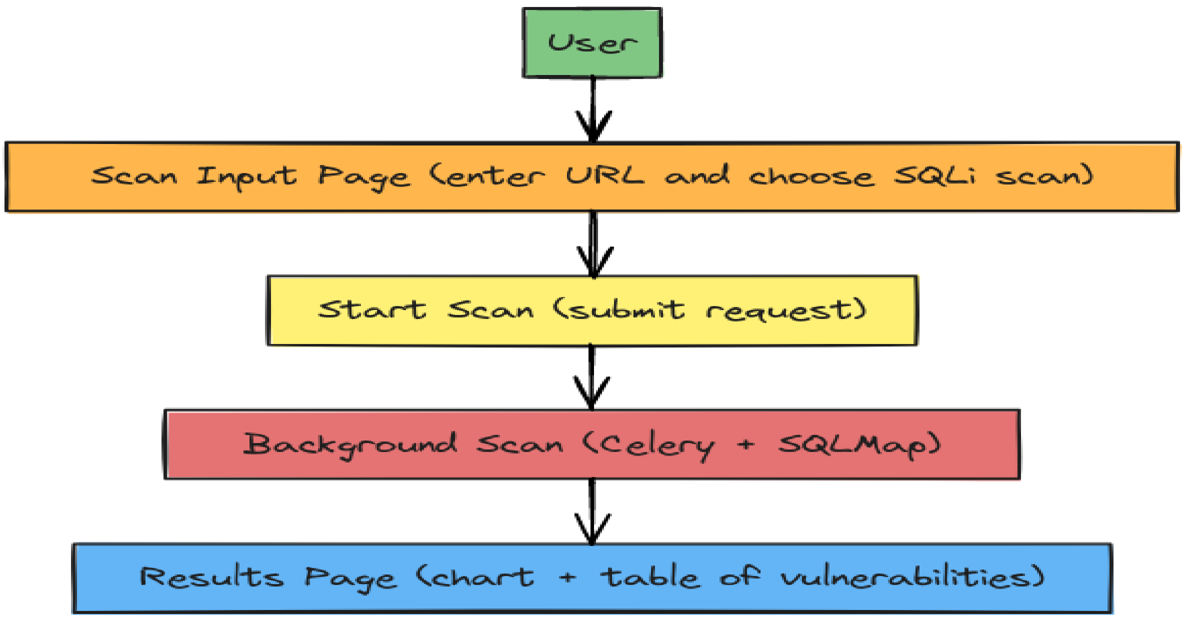

User
Scan Input Page (enter URL and choose SQLi scan)
Start Scan (submit request)
Background Scan (Celery + SQLMap)
Results Page (chart + table of vulnerabilities)

Figure 3.2: User Interface Workflow for SQL Injection Scan

3.2.1 Database Entity Relational Design

The database design is a critical component of the SQL Injection vulnerability scanner, as it defines the structural framework for storing and managing data. A well-designed schema ensures the system operates efficiently, supports scalability, and facilitates easy maintenance (Foster and Godbole, 2022). For this project, the core database design revolves around two entities: Scan and ScanResult.

Overview of Core Entities

The database schema includes two primary entities: Scan, which records details of the scanning process, and ScanResult, which stores the vulnerabilities identified during the scans. A one-to-many relationship connects these entities, ensuring logical data organisation.

3.2.2 Scan Entity

The Scan entity is the focal point of the system. It captures the key details of every scan initiated by a user. This includes the URL being analysed, the status of the scan (e.g., Pending, In Progress, Completed, or Failed), and timestamps indicating when the scan started and finished. In the case of errors, the Scan table also accommodates error messages, providing a mechanism for troubleshooting.

The attributes of the Scan entity are:

- Id (Primary Key): A unique identifier for each scan.
- Url: The URL submitted by the user for scanning.
- Status: The current status of the scan.
- created_at: The timestamp of when the scan was initiated.
- completed_at: The timestamp of when the scan was completed.
- error_message: A field to store any error messages if the scan fails.

2.2.3 ScanResult Entity

The ScanResult entity is designed to store the details of vulnerabilities detected during a scan. Each record in this entity corresponds to a specific vulnerability identified in the URL being scanned. The entity includes fields to describe vulnerability, its severity, and recommendations for remediation. This allows the system to provide users with actionable insights to address potential issues.

The attributes of the ScanResult entity are:

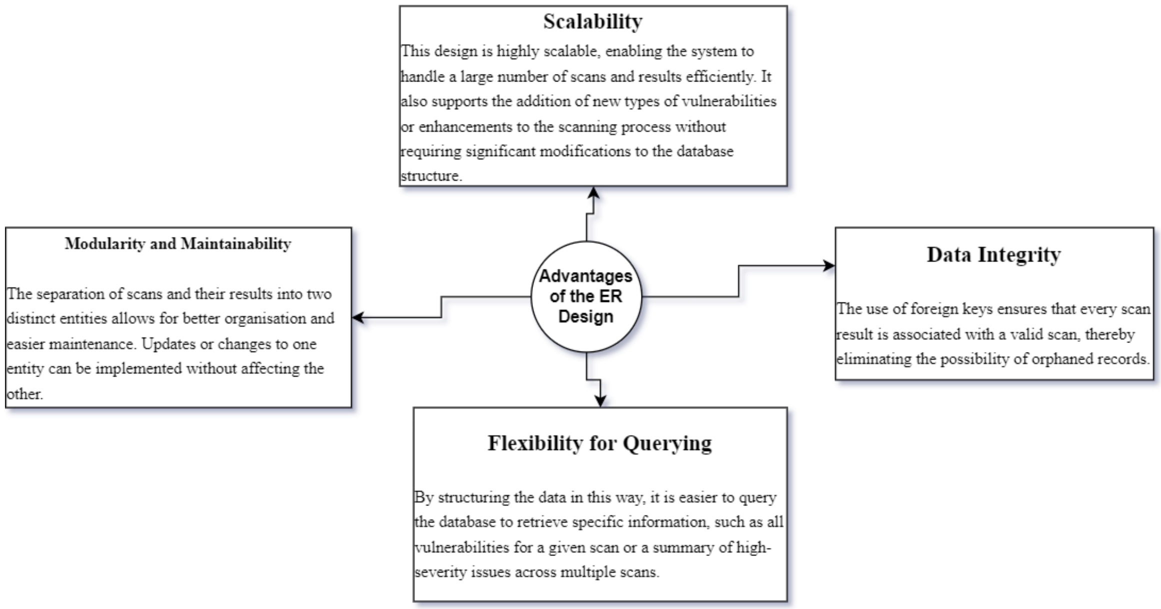
- Id (Primary Key): A unique identifier for each scan result.
- scan_id (Foreign Key): A reference to the associated Scan.
- vulnerability_type: The type of vulnerability identified (e.g., SQL Injection).
- Description: A detailed description of the vulnerability.
- Severity: The severity level of vulnerability (e.g., High, Medium, Low).

- Recommendations: Suggested actions to address vulnerability.
- created_at: The timestamp of when the result was created.

2.2.4 Relationship Between Entities

The relationship between the Scan and ScanResult entities is a one-to-many relationship (Foster & Godbole, 2022). This means that each scan record in the Scan table can be linked to multiple vulnerability records in the ScanResult table. For instance, scanning a single URL might uncover several SQL Injection vulnerabilities, each of which would be recorded as a separate result linked to the same scan.

This relationship is implemented using a foreign key: the scan_id field in the ScanResult table references the id field in the Scan table. This ensures data integrity by preventing the creation of results that are not tied to an existing scan (Figure 3.2).


Scalability
This design is highly scalable, enabling the system to handle a large number of scans and results efficiently. It also supports the addition of new types of vulnerabilities or enhancements to the scanning process without requiring significant modifications to the database structure.
Modularity and Maintainability
The separation of scans and their results into two distinct entities allows for better organisation and easier maintenance. Updates or changes to one entity can be implemented without affecting the other.
Advantages of the ER Design
Data Integrity
The use of foreign keys ensures that every scan result is associated with a valid scan, thereby eliminating the possibility of orphaned records.
Flexibility for Querying
By structuring the data in this way, it is easier to query the database to retrieve specific information, such as all vulnerabilities for a given scan or a summary of high-severity issues across multiple scans.

3.2.5 Entity-Relationship Diagram

Al-Ghifari and Azizah (2022) explained that an Entity-Relationship Diagram (ERD) is an effective way to visualise the database structure (see the section 3.1 for visual connection).

3.2.6 Technologies and Tools

The design and application of the web application vulnerability system is significantly influenced by the number of technologies and tools employed in building it. Although each instrument was chosen for a specific function, they all contribute to the general system performance. In this study, we outline and describe each tool/technology and how they were incorporated in the project.

3.2.7 Django

Django is a high-level traffic application that protects online applications from vulnerabilities. It has built-in attributes that allow the user to get the desired functionality easily. The Model-View-Template (MVT) design of Django facilitates logical arrangement of various entities involved in the web application vulnerability assessment system, thereby organising several aspects of the project (Soliev, Abdurasulova and Yakubov, 2022). Thus, the organisation in the system makes the data operation simple as a result of object rational mapping (ORM) framework.

3.2.8 Celery Server

Celery server is a basic Python tool for background worker processing of queue jobs in a distributed system (Handge, 2022). Based on Redis structure, this study employed celery server to manage efficiently long-running vulnerability scan execution (see figure 3.1). Importantly, the celery server remains agile and actively responsive allowing users to keep interacting with the system while asynchronously in background multiple scans are handled.

3.2.9 Pexpect

Pexpect is a Python module that facilitates control and automate interactions with command-line tools (Pexpect, n.d.). This project introduces pexpect to manage the interactive prompts generated by the security tools, specifically SQLMap. By automating responses to these prompts, Pexpect ensures that the scanning tools control their execution and operate smoothly without requiring manual intervention (Pexpect, n.d.). This automation is crucial for providing a user-friendly experience and ensuring that scans can be executed reliably through the web interface.

3.2.10 SQLMap

SQLMap is an automated penetration testing tool authorized to detect and exploit SQL injection weaknesses (Damele & Stampar, 2024). The SQLMap has a broad feature set that support several SQL injection techniques and databases. Based on this, this project introduces SQLMap to detect and exploit SQL injection vulnerabilities. Pexpect was used to automate the scan process, allowing users to initiate scans and view results through the web app.

3.3 Implementation Steps

3.3.1 Backend

Using Django's Model-View-Template (MVT) design, the backend of the application is rooted in its framework, emphasising the models, chores and opinions, which describes the implementation specifics.

3.3.2 Models

The models state the structure of the data, the application stores and controls (Soliev, Abdurasulova and Yakubov, 2022). "Scan" model is the main model used in this project, which captures all the relevant data on a scan including the scan type, target URL, user who started the scan, completion status, scan results, and scan time. This model guarantees that every scan-related information stays in the database, therefore enabling simple maintenance and retrieval. The "Scan" model also manages the link between several users and their corresponding scans, therefore guaranteeing that every user may only access their own scan data.

3.3.3 Views

Views is the link between models and templates that illustrates the way the web application receives user requests and provides suitable answers (Kumar and Nandal, 2024). The viewpoint produces output based on input logic received. Important points of view involve those for starting scans, monitoring scan conditions, and presenting scan findings. Validating the input data, the “initiate_scan” view generates a new scan record in the database and handles form submission for beginning a new scan. It then sends a Celery server to do the asynchronously scanned scan. While the "scan_results" view retrieves and shows the scan results once the scan is finished, the “scan_status" view offers real-time updates on the scan's development.

3.3.4 Tasks

In this project, tasks specify the type of scan to perform and invoke the required task function. Although it can be multifunctional but for this research, the focus was only on SQL injection. Long-running scan activities are executed asynchronously under management by Celery Server. This guarantees that the web application may get the expected result if a website involves SQL injection or not before showing the result page (Handge, 2022). Every kind of scan has a corresponding task running the suitable security tool and handling the outcomes. The tasks are meant to address several significant instances, including errors during scan running and making sure the database stores the results accurately. The main task, “perform_scan”, orchestrates the scanning process by determining the type of scan to perform and invoking the corresponding task function.

3.3.5 Frontend

The application frontend allow end users to have easy access to web interface required for starting scans and viewing scan results. It facilitates user engagement and interactive web pages using CSS, HTML and JavaScript.

3.3.6 Forms

The form is the basis for initiating a scan through user input. New users have the option to select the type of scan from various alternatives. Creating form and scan type selection is the initial step, then the user navigates to the Target's URL that matches the scan. Thereafter, the server processes the data received or captured via scanning and provides the scan results.

3.3.7 Templates

Templates are HTML files that aid form design and the presentation of scan results. The templates also help to detect security weaknesses, vulnerabilities, and errors. To manage the dynamics, the display of extra choice for SQL injection scans using JavaScript, and the initiation of scan HTML templates determine the structure of starting a new scan. Hence, the template provides end users with simple and basic knowledge concerning the security conditions of their web applications

Templates help to design the forms and show the scan results. Managing the dynamic display of extra choices for SQL injection scans using JavaScript, the "initiate_scan.html" template forms the structure for starting a new scan. The template “scan_results” shows the scan's findings, including any flaws or vulnerabilities found, as well as error(s) that were discovered in the HTML. The simple designs of the templates provide consumers with clear, practical knowledge of the security conditions of their web apps.

3.3.8 User Interface Design Process

The user interface was designed with non-technical users in mind. The main goal was to keep the process simple: enter a URL, choose the scan type, and clearly see the results. The design followed three basic steps: understanding user needs, creating simple sketches, and improving the interface during development.

First, the target user group was defined as e-commerce owners and staff who do not have a security background but need to know if their websites are at risk. From this, three key requirements were identified: the interface should use plain language, minimise the number of inputs, and present results in a way that is easy to read at a glance.

Second, low-fidelity paper sketches and basic digital mock-ups were created for the main pages. These sketches focused on a clean layout with a single URL input box, a simple drop-down list for the scan type, and a clear “Start Scan” button. The results page was also sketched with a bar chart at the top to show vulnerability severity and a table underneath with more details and recommendations.

Third, the interface was refined during implementation. As the backend logic and Celery tasks were added, the layout was adjusted to keep loading states and error messages visible. For example, a simple status message was added so users can see when a scan is “Pending”, “In Progress”, or “Completed”. Informal feedback from peers was used to remove unnecessary technical terms and to make the labels clearer. This small, iterative process helped to align the interface with the needs of non-technical users without adding extra complexity.

3.4 Security Tools Integration

The system's main functionality emanated from the integration of multiple security technologies into the web application, thereby enhancing holistic vulnerability evaluations. Each security technology in the system is tailored to identify a particular security weakness while the integration of these tools guarantees a robust vulnerability assessment system.

3.4.1 Scan for SQL injection

This research employs SQLmap for its unique ability to detect and exploits SQL injection flaw in web applications. The automated attribute of the tool simplifies the process of detecting flaws, thereby making it a valuable tool for users. The integration of pexpect with SQLmap required automating SQLmap's command-line interaction to enhance easy control of difficult queries and user inputs. Hence, this integration enhances the tool to identify and exploit SQL vulnerabilities in web applications.

3.4.2 Mitigation Steps

After the vulnerability is scanned and detected in the e-commerce application, then the figure 3.3 demonstrates the mitigation steps. After scanning, the vulnerability can be found or not. If the vulnerability is found, proceed to implement the relevant framework to fix it; if the vulnerability is not found, check for another vulnerability that is implemented in the application.

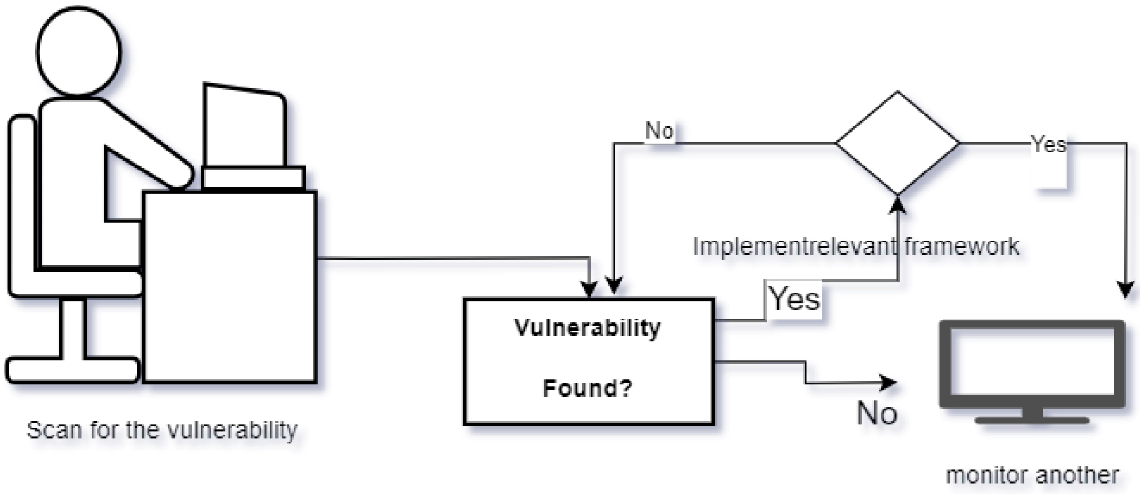


Figure 3.3. SQL Vulnerability mitigation Approach

3.5 Error Handling and Logging

3.5.1 Error Logging

Good error logging is identified by diagnosing attributes, performance tracking records, and system reliability. With the use of Django's in-built logging system, the web application vulnerability detection system gathers and stores error reports. Django's in-built logging system record initiation time, completion time, error and detected during scan. Importantly, the historical log errors are evaluated by the administrator for decision-making purposes. Thus, this preserves an audit record of all activities, detailed logging guarantees scanning process transparency and responsibility.

3.5.2 Exception Handling

Exceptional handling helps to maintain normal functions of web application during abnormal conditions to ensure that anomalies do not crash the web application. Every Celery Server in the backend does scans with try-except blocks to manage anomalies. An extensive error message is recorded and the scan record in the database is changed with the error information when an error arises. This method guarantees consumers clear feedback on any problems found during their scan requests and helps to stop the program from crashing. Furthermore, exception handling guarantees that, should a scan fail, partial findings are still accessible, therefore enabling users to learn about the security situation of their web application.

3.6 Data Collection and Visualisation

Although no direct primary data is collected or pre-existing datasets are used, this approach focuses on generating operational data through the interaction of users with the developed application (Saura, Ribeiro-Soriano and Palacios-Marqués, 2021).

The application itself serves as a tool to allow users to input their own URLs for testing against SQL injection vulnerabilities. The data is generated dynamically as users interact with the system.

To ensure the results generated by the application are effectively presented and meaningful to users, appropriate visualisation techniques are integrated into the system (Qin et al., 2020). These visualisations are designed to summarise the SQL injection vulnerabilities detected during the scanning process, providing an intuitive representation of the findings. This section outlines the visualisation strategies employed in the application. A bar chart is used to depict the count of vulnerabilities by their severity levels (e.g., High, Medium, and Low). A tabular view is used to provide a detailed breakdown of the detected vulnerabilities.

3.7 Ethical Considerations

In research, it is important to clearly define the ethical standpoint of a study. This ethical standpoint is most important when the research involves data gathering, either primarily or secondarily. In this study, there is no direct data gathering, maybe from the participants (primary) or using the available dataset (secondary). The only data gathered in this study is the result output during the scanning of the SQL injection attacks; hence, this research does not have any ethical implications.

3.8 Summary

This chapter covered the research methodology, which includes the system architecture, dynamic user input handling, and implementation of backend and frontend components, the mitigation steps, and so on. Chapter 4 documents the analysis and results of the developed application.

4. Results

This chapter provides a discussion on the development and evaluation as shown in the previous chapter. It begins by presenting the Entity-Relationship (ER) diagram, which outlines the system's database design and highlights its core entities. The chapter then details the development process, showcasing the implementation of code and the integration of system components. A thorough testing process is discussed, focusing on the scanner's ability to detect vulnerabilities and the accuracy of its results on different websites. Mitigation strategies for addressing identified vulnerabilities are proposed, alongside a comparison with existing studies to evaluate the system's effectiveness and innovation. Finally, the chapter answers the research question, which demonstrates how the developed system bridges existing gaps and provides a practical solution for non-technical users.

4.1 ER diagram

Figure 4.1 shows the ER diagram as described in Chapter 3 (Section 3.1.3) of this study. It highlights key entities such as *Scan* and *ScanResult*, which form the backbone of the system. The *Scan* entity records critical details about each scan, including its URL, status, and timestamps. This ensures comprehensive tracking of scanning activities. The *ScanResult* entity captures detected vulnerabilities, their severity, and recommended mitigation strategies. A one-to-many relationship between Scan and *ScanResult* ensures logical organisation.

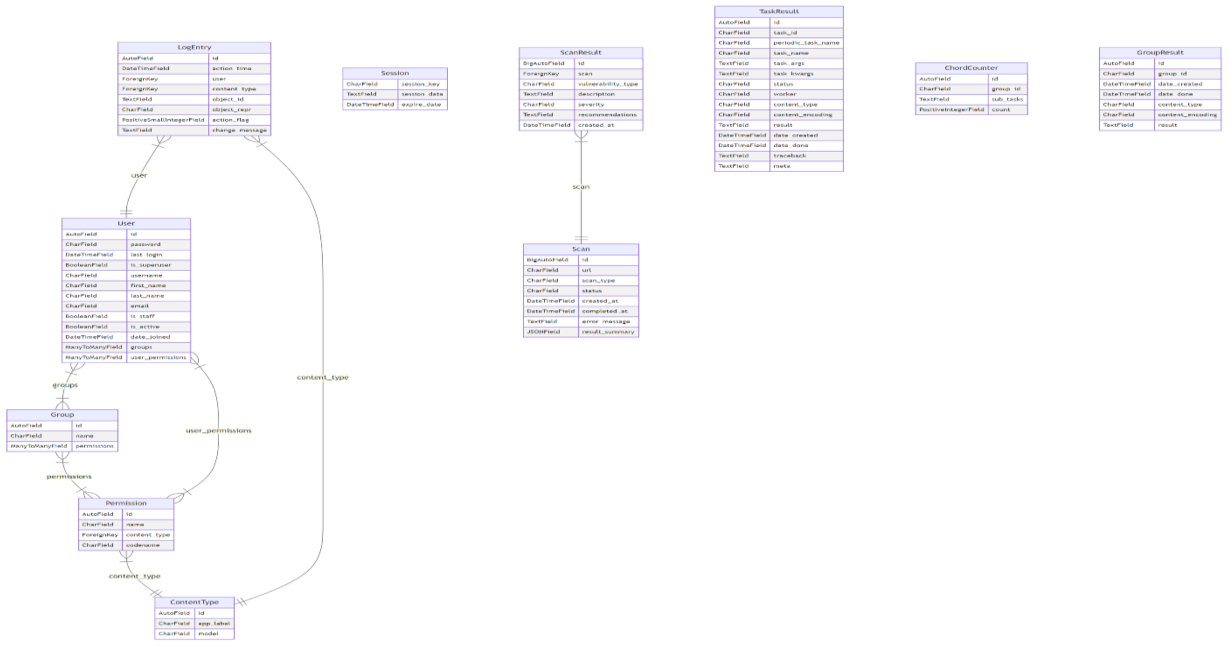
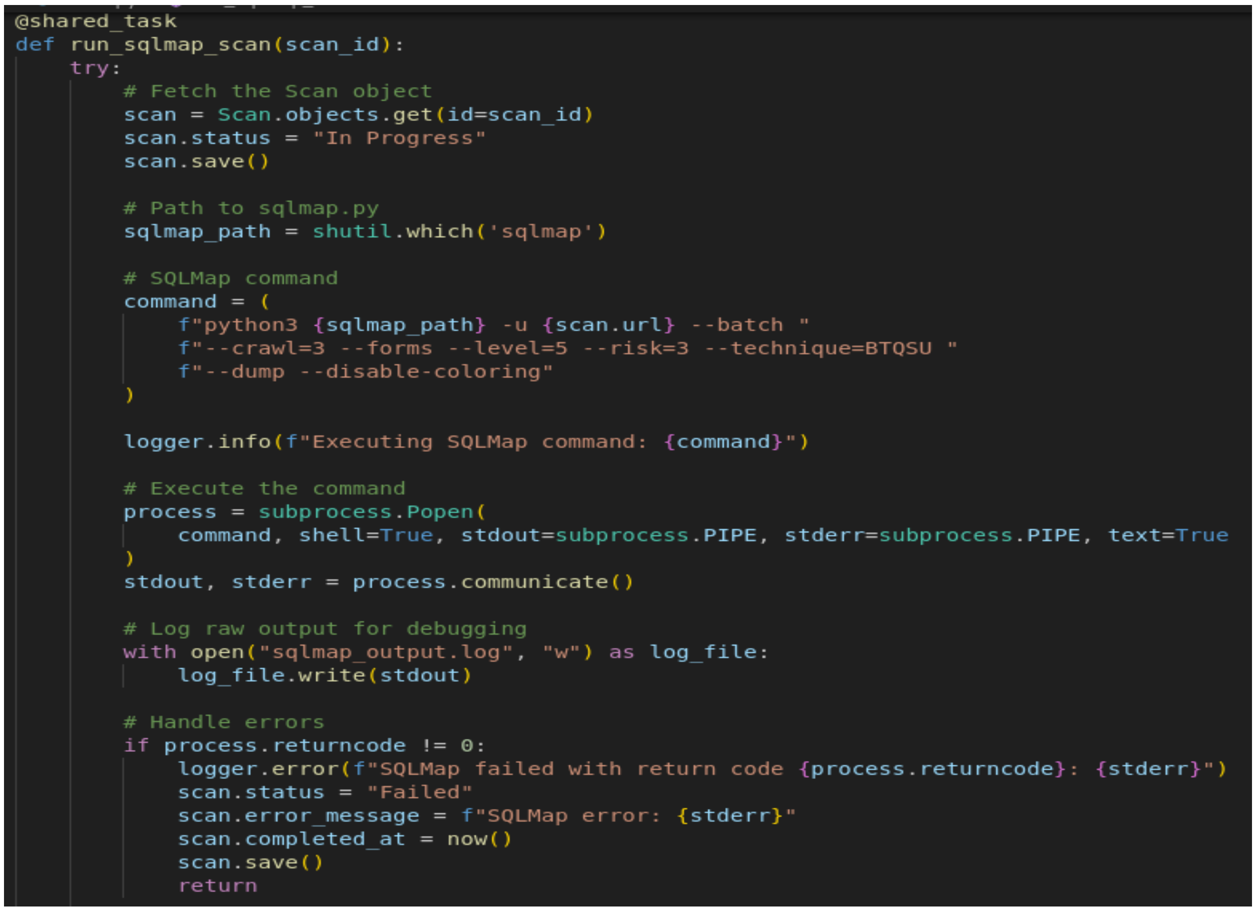


Figure 4.1 The ER Diagram

4.2 Development Environment Setup

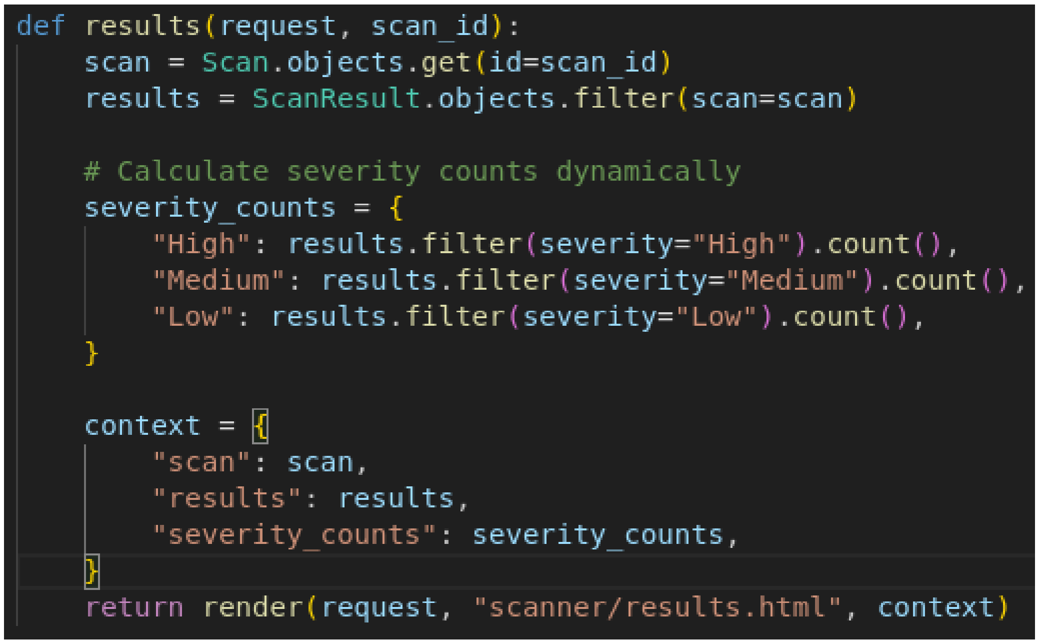
Figure 4.2 illustrates the SQL injection scanning function within the system. The process begins by fetching the scan object based on the provided scan ID, updating its status to "In Progress", and saving it. The SQLmap path is dynamically retrieved to ensure compatibility, followed by constructing the SQLmap command string with various flags, such as risk level and techniques. The subprocess.Popen module is used to execute the command, capturing output and errors. Logs are written to a file for debugging purposes. In cases of failure, the scan's status is updated to "Failed," and error messages are logged for troubleshooting.



```
@shared_task
def run_sqlmap_scan(scan_id):
    try:
        # Fetch the Scan object
        scan = Scan.objects.get(id=scan_id)
        scan.status = "In Progress"
        scan.save()

        # Path to sqlmap.py
        sqlmap_path = shutil.which('sqlmap')

        # SQLMap command
        command = (
            f"python3 {sqlmap_path} -u {scan.url} --batch "
            f"--crawl=3 --forms --level=5 --risk=3 --technique=BTQSU "
            f"--dump --disable-coloring"
        )

        logger.info(f"Executing SQLMap command: {command}")

        # Execute the command
        process = subprocess.Popen(
            command, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE, text=True
        )
        stdout, stderr = process.communicate()

        # Log raw output for debugging
        with open("sqlmap_output.log", "w") as log_file:
            log_file.write(stdout)

        # Handle errors
        if process.returncode != 0:
            logger.error(f"SQLMap failed with return code {process.returncode}: {stderr}")
            scan.status = "Failed"
            scan.error_message = f"SQLMap error: {stderr}"
            scan.completed_at = now()
            scan.save()
            return
```

Figure 4.2: Perform scan task

Figure 4.3 demonstrates the function responsible for managing and displaying scan results. The function retrieves the scan object and associated scan results using the scan ID. It dynamically calculates severity counts for identified vulnerabilities, categorising them as High, Medium, or Low. This allows for clear representation of the scan's outcome. The calculated data, along with the scan details and results, are structured into a context dictionary. Finally, the *render* method is invoked to generate the results page (*results.html*) using the context.



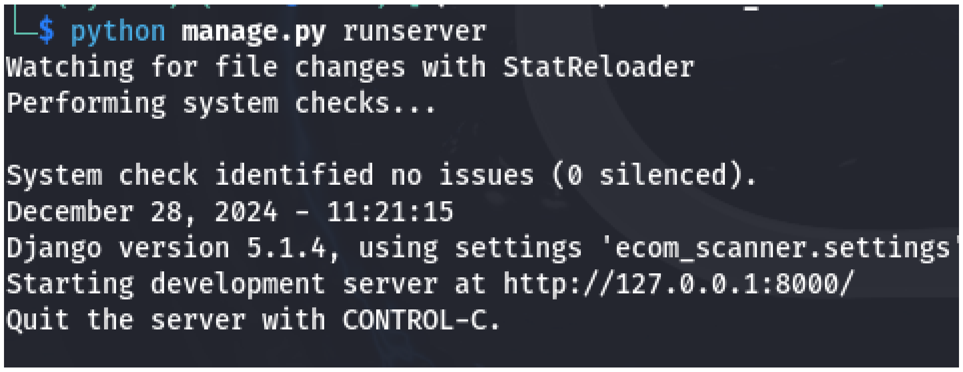
```python
def results(request, scan_id):
    scan = Scan.objects.get(id=scan_id)
    results = ScanResult.objects.filter(scan=scan)

    # Calculate severity counts dynamically
    severity_counts = {
        "High": results.filter(severity="High").count(),
        "Medium": results.filter(severity="Medium").count(),
        "Low": results.filter(severity="Low").count(),
    }

    context = {
        "scan": scan,
        "results": results,
        "severity_counts": severity_counts,
    }
    return render(request, "scanner/results.html", context)
```

Figure 4.3: Processing scan results

Using Django's ORM, the function in Figure 4.3 updates the status of the scan and stores the results after a scan has concluded, therefore showing the flawless integration between the database of the web application and SQLMap output.

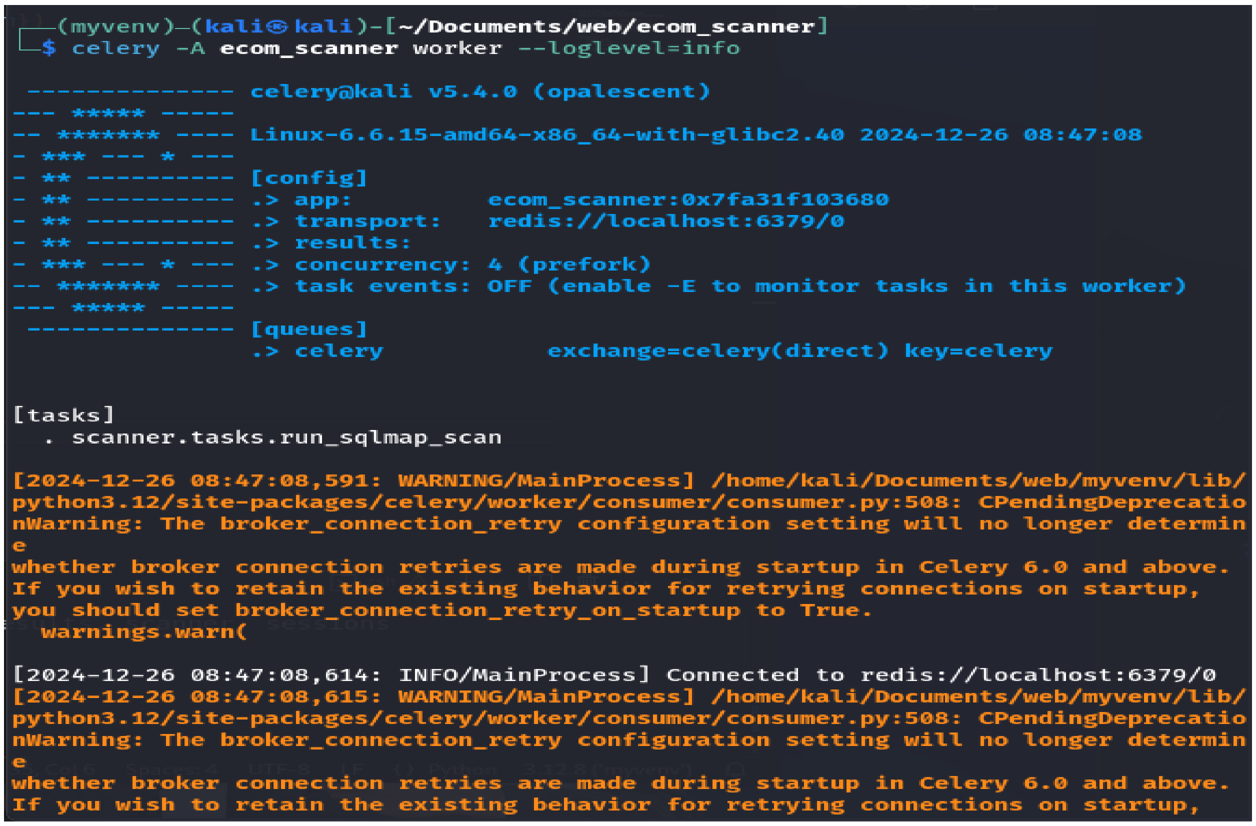
4.3 Running the Necessary Servers

The figure 4.3 and figure 4. Django server and Celery server, respectively. These tools are important for the successful running of the web application. The Django server helps to run the application on the web browser, the celery-server is important for the synchronous tasks.



```
└─$ python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
December 28, 2024 - 11:21:15
Django version 5.1.4, using settings 'ecom_scanner.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Figure 4.4 Django Local Server Running

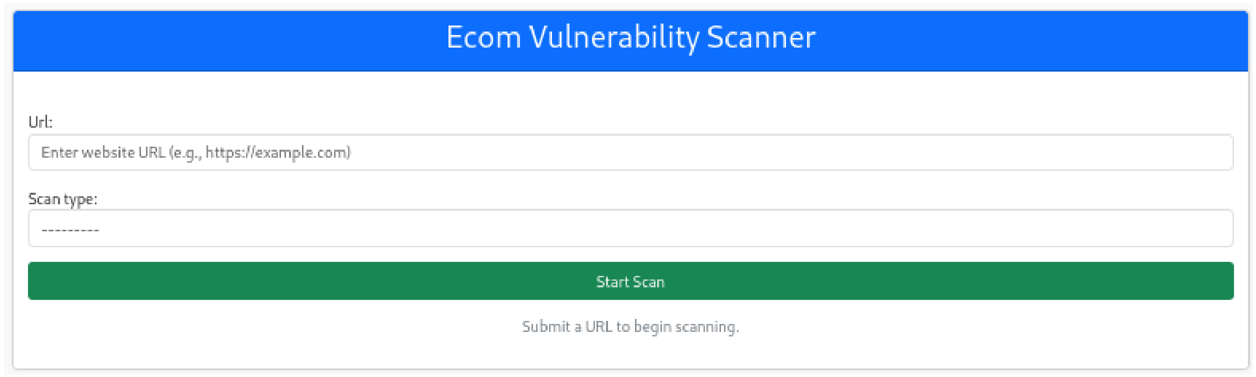


```
┌──(myvenv)─(kali㉿kali)-[~/Documents/web/ecom_scanner]
└─$ celery -A ecom_scanner worker --loglevel=info

 -------------- celery@kali v5.4.0 (opalescent)
--- ***** -----
-- ******* ---- Linux-6.6.15-amd64-x86_64-with-glibc2.40 2024-12-26 08:47:08
- *** --- * ---
- ** ---------- [config]
- ** ---------- .> app:         ecom_scanner:0x7fa31f103680
- ** ---------- .> transport:   redis://localhost:6379/0
- ** ---------- .> results:
- *** --- * --- .> concurrency: 4 (prefork)
-- ******* ---- .> task events: OFF (enable -E to monitor tasks in this worker)
--- ***** -----
 -------------- [queues]
                .> celery           exchange=celery(direct) key=celery


[tasks]
  . scanner.tasks.run_sqlmap_scan

[2024-12-26 08:47:08,591: WARNING/MainProcess] /home/kali/Documents/web/myvenv/lib/
python3.12/site-packages/celery/worker/consumer/consumer.py:508: CPendingDeprecatio
nWarning: The broker_connection_retry configuration setting will no longer determin
e
whether broker connection retries are made during startup in Celery 6.0 and above.
If you wish to retain the existing behavior for retrying connections on startup,
you should set broker_connection_retry_on_startup to True.
  warnings.warn(

[2024-12-26 08:47:08,614: INFO/MainProcess] Connected to redis://localhost:6379/0
[2024-12-26 08:47:08,615: WARNING/MainProcess] /home/kali/Documents/web/myvenv/lib/
python3.12/site-packages/celery/worker/consumer/consumer.py:508: CPendingDeprecatio
nWarning: The broker_connection_retry configuration setting will no longer determin
e
whether broker connection retries are made during startup in Celery 6.0 and above.
If you wish to retain the existing behavior for retrying connections on startup,
```

Figure 4.3: The Celery Server running.

4.4 Results of the scanned result

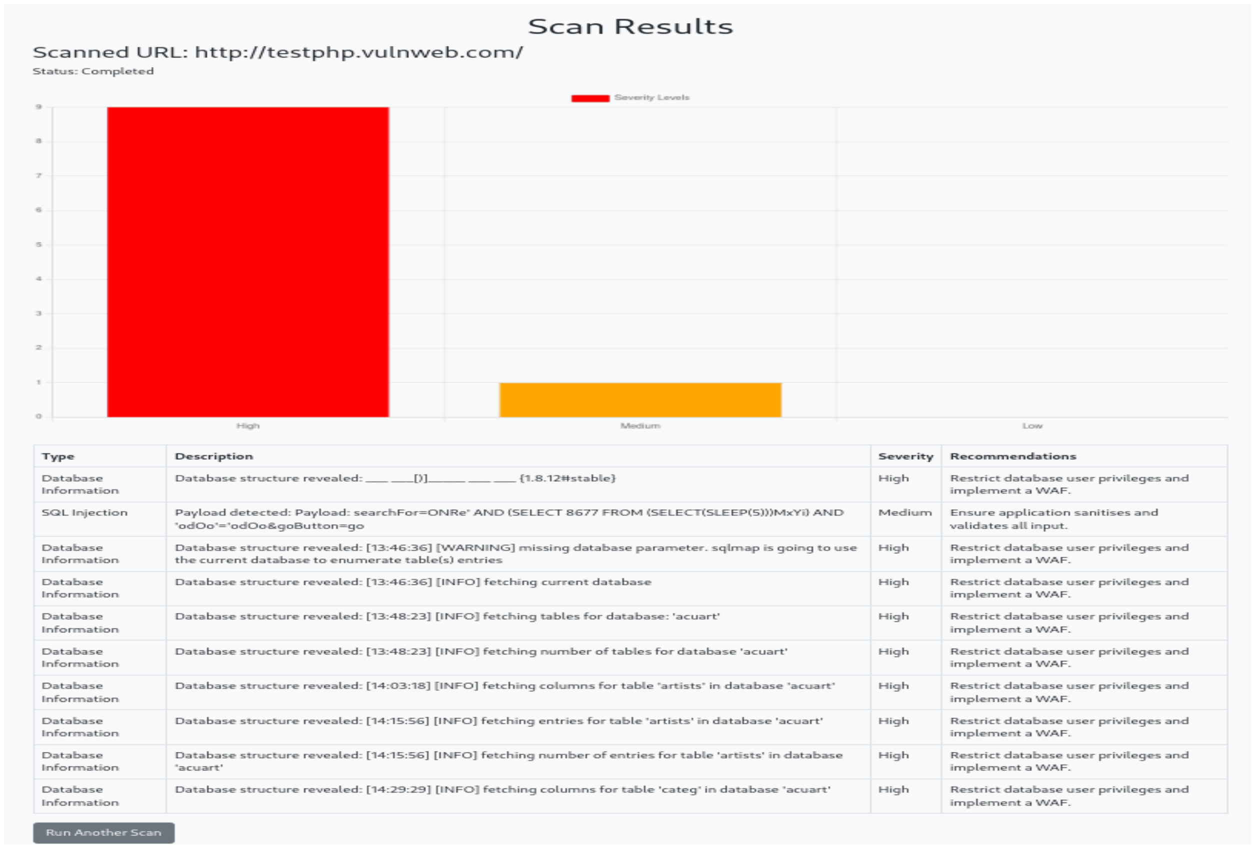
Figure 4.5 presents the user interface of the application, showcasing its simplicity and accessibility. The interface requires the user to input a URL and select a scan type (in this case, only SQL injection is available) before initiating the scan. The "Start Scan" button ensures an intuitive experience, making the tool user-friendly for non-technical users. By streamlining input requirements, this interface eliminates complexities often associated with vulnerability scanning tools, supporting the aim of providing practical solutions for identifying vulnerabilities like SQL injection. To make the layout clearer, Figure 4.9 presents a simple mock-up of the main interface. It shows the URL input field, scan type selector, and "Start Scan" button on one page, with a clean structure that avoids unnecessary options. This visual representation supports the aim of making the scanner easy to use for non-technical users.


Ecom Vulnerability Scanner
Url:
Enter website URL (e.g., https://example.com)
Scan type:

Start Scan
Submit a URL to begin scanning.

Figure 4.5 The interface to start the scanning process

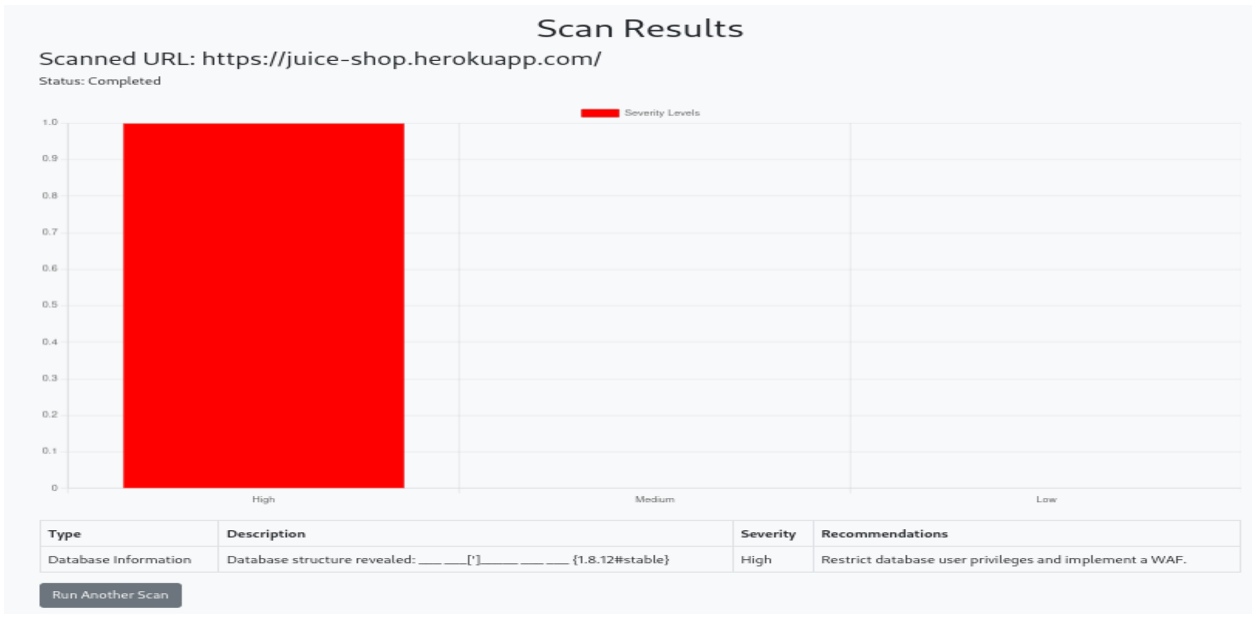
Figure 4.6 details the scan results of a specific target URL (testphp.vulnweb.com). A comprehensive overview is provided, summarising the severity of detected vulnerabilities through a bar chart and a tabular representation. The chart clearly illustrates that high-severity vulnerabilities dominate, highlighting significant risks. The table complements this by categorising each finding by type, description, severity, and recommended mitigation. High-severity issues often relate to database structure revelations and SQL injection, emphasising the critical need for security measures like Web Application Firewalls (WAF) and proper input validation.



Type	Description	Severity	Recommendations
Database Information	Database structure revealed: ___ ___[)]______ ___ ___ {1.8.12#stable}	High	Restrict database user privileges and implement a WAF.
SQL Injection	Payload detected: Payload: searchFor=ONRe' AND (SELECT 8677 FROM (SELECT(SLEEP(5)))MxYi) AND 'odOo'='odOo&goButton=go	Medium	Ensure application sanitises and validates all input.
Database Information	Database structure revealed: [13:46:36] [WARNING] missing database parameter. sqlmap is going to use the current database to enumerate table(s) entries	High	Restrict database user privileges and implement a WAF.
Database Information	Database structure revealed: [13:46:36] [INFO] fetching current database	High	Restrict database user privileges and implement a WAF.
Database Information	Database structure revealed: [13:48:23] [INFO] fetching tables for database: 'acuart'	High	Restrict database user privileges and implement a WAF.
Database Information	Database structure revealed: [13:48:23] [INFO] fetching number of tables for database 'acuart'	High	Restrict database user privileges and implement a WAF.
Database Information	Database structure revealed: [14:03:18] [INFO] fetching columns for table 'artists' in database 'acuart'	High	Restrict database user privileges and implement a WAF.
Database Information	Database structure revealed: [14:15:56] [INFO] fetching entries for table 'artists' in database 'acuart'	High	Restrict database user privileges and implement a WAF.
Database Information	Database structure revealed: [14:15:56] [INFO] fetching number of entries for table 'artists' in database 'acuart'	High	Restrict database user privileges and implement a WAF.
Database Information	Database structure revealed: [14:29:29] [INFO] fetching columns for table 'categ' in database 'acuart'	High	Restrict database user privileges and implement a WAF.

Figure 4.6 Testing on testphp.vulnweb.com

Figure 4.7 and Figure 4.8 showcases the results of scanning another URLs (juice-shop.herokuapp.com and itsecgames.com). Similar to Figure 4.6, the bar chart reflects a predominance of high-severity vulnerabilities. The concise tabular data identifies database-related weaknesses, further validating the scanner's ability to detect critical security flaws. Recommendations consistently stress restricting database privileges and implementing WAFs, reinforcing best practices for securing web applications.



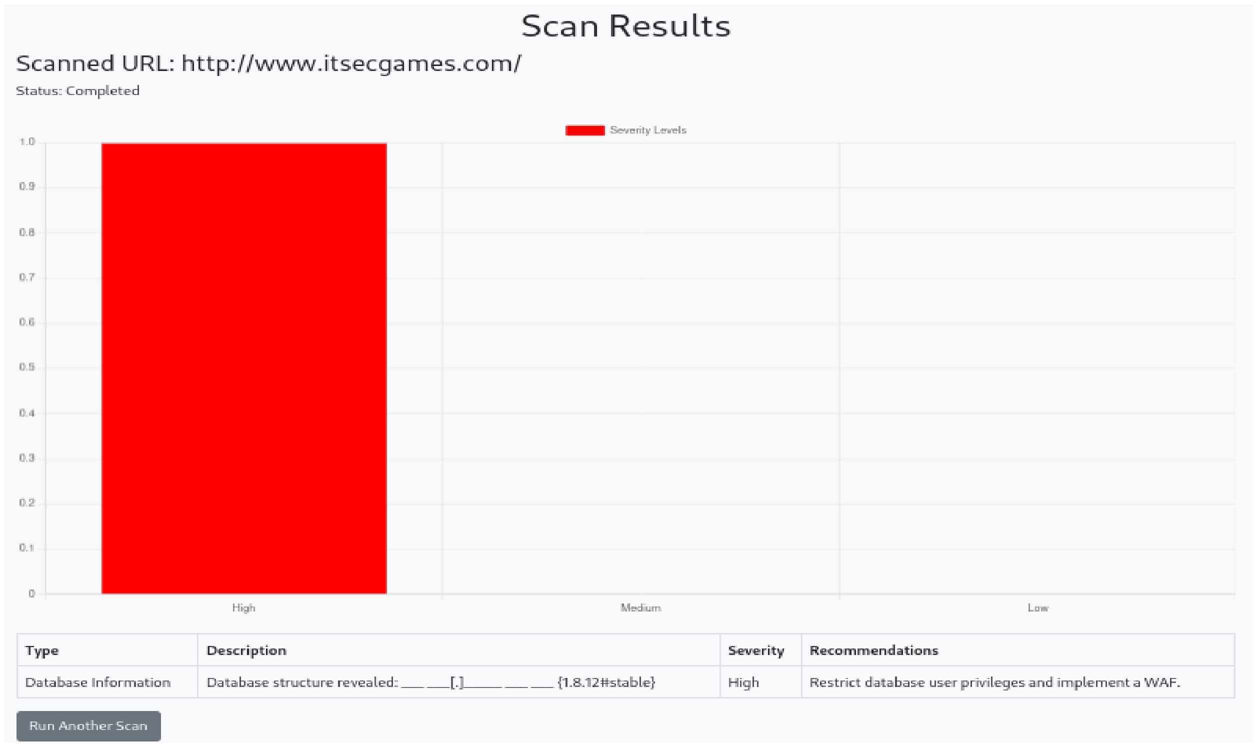
Scan Results

Scanned URL: https://juice-shop.herokuapp.com/

Status: Completed

Type	Description	Severity	Recommendations
Database Information	Database structure revealed: ___ ___['']_____ ___ ___ {1.8.12#stable}	High	Restrict database user privileges and implement a WAF.

Figure 4.7 Testing on juice-shop.herokuapp.com



Scan Results

Scanned URL: http://www.itsecgames.com/

Status: Completed

Type	Description	Severity	Recommendations
Database Information	Database structure revealed: ___ ___[.]_____ ___ ___ {1.8.12#stable}	High	Restrict database user privileges and implement a WAF.

Figure 4.8 Testing on itsecgames.com

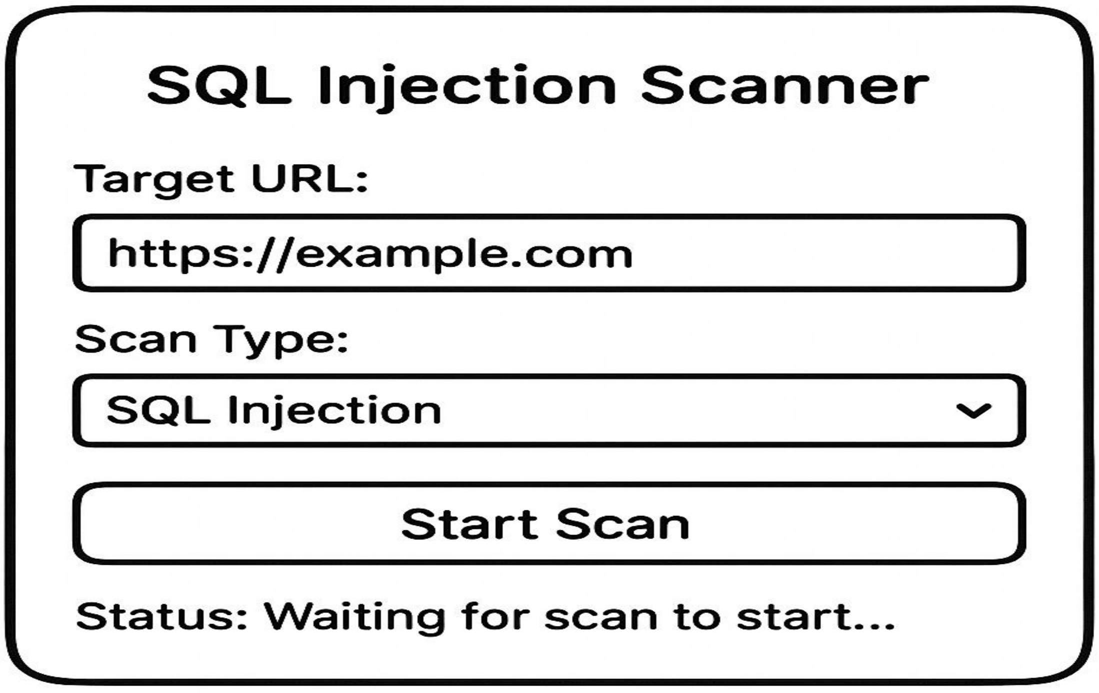


Figure 4.9: Example user interface mock-up for starting a scan

4.5 Challenges and Solutions

SQL injection (SQLi) remains one of the most critical threats to web applications, particularly in e-commerce systems where sensitive user data is at stake. Despite the availability of detection and prevention tools, numerous challenges persist, as identified in our research. This section discusses these challenges alongside solutions supported by recent studies.

4.5.1: Sophistication of Modern SQL Injection Techniques

Research by Alqahtani, Eghan and Rilling (2016) highlights the increasing sophistication of SQLi attacks. Attackers now employ advanced techniques such as blind SQL injection, time-based injection, and logical inference to bypass detection mechanisms. These techniques involve injecting queries that produce no visible output, making them exceptionally challenging to identify using traditional approaches.

Dynamic analysis, as validated in several recent study (Kuroki et al., 2020), has emerged as an effective countermeasure. Unlike static approaches that rely on predefined patterns, dynamic tools like SQLMap (which was used in this study) simulate real-world attack scenarios to uncover vulnerabilities.

4.5.2: Lack of Security Awareness in E-Commerce Teams

One of the critical challenges noted in this research and supported by Zhuang (2023) is the lack of security expertise among non-technical stakeholders in e-commerce businesses. Team members unfamiliar with SQL injection concepts often struggle to interpret vulnerability reports or implement recommended solutions, leading to delays in remediation.

Simplifying user interfaces and providing actionable insights can empower non-technical users to address vulnerabilities effectively. Tools with intuitive dashboards, like the one illustrated in Figure 4.5, and clear recommendations (e.g., "Restrict database privileges" or "Implement a Web Application Firewall") bridge the gap between technical complexity and user understanding. As stated by Zhuang (2023), usability-focused designs significantly improve the adoption and efficacy of security tools.

4.5.3. Ease of view of the Scan Result

Providing real-time remarks about continuous scanning proved still another challenging task. Users must be informed about the development of scans without directly refreshing the page so as to enhance their interaction and experience with the tool. To address this, periodic AJAX requests were thus executed on the client side. Then dynamically displayed to the user, enquires related to server for scan status updates.

This modification considerably improved the interaction of the software. Particularly during lengthier scans, users could find advantage in real-time view of the scan development. This not only matched user expectations but as w ell reduced perceived wait times, hence increasing system user-friendliness.

4.6 Comparison with Existing Studies

The methods applied in this work fit current trends in vulnerability assessment and web application security. For example, according to 2020 Devi and Kumar research, the investigation revealed several security flaws in web applications discovered by Nikto-OWASP ZAP integration. This project also aggregates various tools to highlight their relevance in contemporary security policy. Jarupunphol et al. (2023) underlined their importance in identifying high-risk vulnerabilities in web programs and also evaluated vulnerability assessment tools including OWASP ZAP and Burp Suite. This comparison highlights the requirement of applying different technologies to have a comprehensive awareness of the security posture of an application.

In another study, Karangle et al. (2019) assessed Uniscan's and Nikto's efficacy for URL vulnerability assessment; Uniscan was better fit for thorough testing including SQL injection, while Nikto was perfect fit for surface-level scans(Karangle et al., 2019). This result validates Nikto's choice on this project for efficient first assessments. By means of an Interactive Application Security Testing (IAST) approach integrating technologies like ZAP-API and

Jenkins for a comprehensive security evaluation, Setiawan et al. (2020) also exposed various vulnerabilities. This method conforms with the present work's use of ZAP-Cli, which revealed the instrument's effectiveness in vulnerability identification.

Lastly, Abdullah (2020) evaluated open-source web application vulnerability scanners underlining the critical role automated tools like as Nikto and OWASP ZAP perform in safeguarding web applications (Ab Abdullah, 2020). Thus, the combination of these tools reflects industry's best standards and guarantees security evaluations using just the online interface.

4.7 Mitigation Strategy

Reducing SQL injection vulnerabilities in e-commerce systems is a vital task that needs to follow accepted security policies and compliance rules. As demonstrated by current studies, the complexity of modern SQL injection methods calls for a multi-layered strategy to guarantee strong protection.

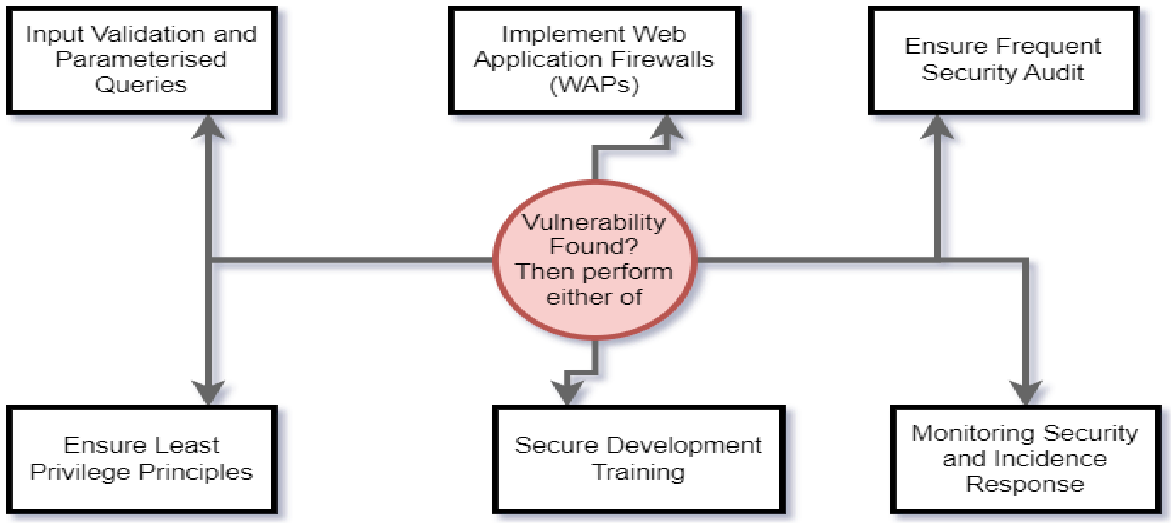

Input Validation and Parameterised Queries
Implement Web Application Firewalls (WAPs)
Ensure Frequent Security Audit
Vulnerability Found? Then perform either of
Ensure Least Privilege Principles
Secure Development Training
Monitoring Security and Incidence Response

Figure 4.9 Mitigation Strategy

4.7.1 Input Validation and Parameterised Queries

Recent research highlights the need for parameterised searches and input validation as main defences against SQL injection. Chukwuemeka's (2024) shows that treating user input as mere data rather than code helps parameterized searches to significantly stop harmful SQL statements from being carried out. This approach is further reinforced by guidelines from OWASP, which recommend avoiding dynamic SQL queries altogether.

Implementing Web Application Firewalls (WAFs)

Web Application Firewalls (WAFs) have proven effective in mitigating SQL injection attacks by filtering malicious input before it reaches the application layer. Research by Alotaibi and Vassilakis (2023) shows how well pre-configured criteria and machine-learning algorithms identify and prevent harmful payloads. WAF integration with e-commerce systems offers a further security layer, therefore greatly lowering the risk of exploitation.

4.7.2 Frequent Security Audit

Early in the development life, regular security audits and code reviews are absolutely crucial for spotting weaknesses. As Devi and Kumar (2020) point out, automated scanning techniques like SQLMap and methodical code reviews help to find possible injection sites. As Alghawazi, Alghazzowicz, and Alarifi (2022) have stated, penetration testing should supplement these audits to replicate real-world attack circumstances.

4.7.3 Least Privilege Principle

Using the least privileged principle is another absolutely important mitigating strategy. Limiting database user rights reduces the possible impact of a successful injection attack, according to research by Baklizi et al. (2023). E-commerce systems, for instance, should limit database accounts to particular roles such as read-only access, therefore eliminating the ability to run administrative commands.

4.7.4 Secure Development Training

Secure coding techniques must be explained to developers so they may identify and fix problems throughout development. According to Dawadi, Adhikari, and Srivastava (2023), regular training courses and seminars raise developers' knowledge of risks like SQL injection, therefore guaranteeing that safe coding techniques are maintained constantly.

4.7.5 Security Monitoring and Incident Response

Lastly, using an incident response strategy and real-time monitoring will help to greatly improve the resilience of an e-commerce platform. Studies by Kumari et al. (2022) support the monitoring of database activity for signs of injection attacks using devices including Intrusion Detection Systems (IDS). A well-prepared response strategy guarantees fast containment and recovery in the case of an incident.

4.8 Answers to Research Questions

RQ1: How may a user interface be built to let non-technical users efficiently conduct vulnerability testing on e-commerce online applications?
The need for simplicity and intuitability in user interface design for vulnerability testing tools is underlined in this paper. Navigating difficult security technologies might be difficult for

nontechnical users. Alghawazi, Alghazzawi, and Alarifi's (2022) research shows that user interfaces should be simple to interact with, even without third-party guidelines. These ideas guarantee that users may start scans without advanced technological understanding. Visual aids such as graphs and charts also help to simplify scan result interpretation. For example, classifying vulnerabilities according to severity—high, medium, and low—helps users to prioritise remedial resources most wisely. The study's use of a simple URL input field and a "Start Scan" button shows how minimalist design helps to promote more general acceptance of security methods.

RQ2: What automated techniques and tools can be used to identify common security vulnerabilities in e-commerce web applications, and how can they be integrated into a vulnerability detection tool?
Finding weaknesses like SQL injection mostly depends on automated tools like. Research by Alotaibi and Vassilakis (2023) confirms how well these tools simulate attack scenarios and expose hidden flaws. By examining database responses to created queries, SQLMap, for instance, automatically finds SQL injection points. As the present study shows, by using Django ORM, integration of such technologies with e-commerce systems may be accomplished in a simple approach. Including regular scans and real-time alarms guarantees ongoing security surveillance as well. Modern e-commerce security plans revolve around automation, as studies by Dawadi, Adhikari and Srivastava (2023) underline that it not only accelerates vulnerability identification but also lowers human error.

RQ3: How can the user interface of a vulnerability detection tool be designed to effectively communicate complex security concepts to non-technical users, and what features can be included to facilitate broader participation in web application security?
Effective communication of complex security concepts involves presenting information in an accessible manner. Research by Baklizi et al. (2023) validates the value of translating technical jargon into actionable insights. For example, instead of displaying raw SQL injection payloads, the interface should provide recommendations like “Implement input validation” or “Use parameterised queries”. Features such as dashboards summarising scan outcomes and severity levels are invaluable. The study’s inclusion of real-time scan updates and categorised vulnerability reports addresses these needs, ensuring users remain informed throughout the scanning process. Additionally, providing links to educational resources within the tool can empower users to understand security issues better, fostering a culture of proactive vulnerability management.

4.9 Summary

The result of the methodology described in the last chapter is fully discussed in this chapter. The next chapter records the reflection and the project evaluation.

5. Discussion

5.1 Overview of Project

This project aims to create an easy-to-use interface to help non-technical users in e-commerce web application common security vulnerability identification and scanning for especially SQL injection. To offer effective vulnerability identification and mitigation, the system combines several technologies—including Django, SQLMap, and Celery. By means of automation and elegant design, it closes the distance between intricate cybersecurity technologies and user accessibility. Testing on other platforms proved how well it could identify vulnerabilities and offer practical recommendations. The initiative also investigated mitigating techniques and compared findings with industry standards, therefore highlighting its contribution to improve the security of e-commerce systems.

5.2 Objectives Review

The project has fulfilled its primary objectives successfully. Development of a user-friendly interface allowed non-technical users of e-commerce online apps to efficiently do vulnerability searches. Including automated techniques like SQLMap, the system precisely found vulnerabilities in line with Objective 1. Examining one of the most important vulnerabilities, SQL injection, gave important new perspectives on its frequency and influence, therefore addressing Objective 2. Objective 3: A detailed literature review guided the scanning system's design. Finally, in line with Objective 4, the project suggested and evaluated successful mitigating strategies.

5.3 Evaluation of Methodology and Results

The strategy used for this project was based on the design and development of a system capable of identifying SQL injection (SQLi) vulnerabilities in e-commerce web applications. The project aimed to offer a user-friendly, automated platform for locating and reducing significant vulnerabilities. This was accomplished by integrating tools such as Django, SQLMap, and Celery.

5.3.1 Evaluation of SQLMap's ability to identify SQL integration vulnerabilities

SQLMap was an essential part of the approach, and it was employed because of the substantial capabilities it has in identifying SQL injection vulnerabilities. Research shows that SQLMap is advantageous; it has been shown to effectively attack SQL injection points in a number of test cases, including simulated real-world environments such as the Damn Vulnerable Web Application (DVWA) (Abdullah et al., 2023). Specifically, when evaluated across various levels of application security, another study demonstrated its superior performance compared to other tools such as Acunetix and Burp Suite (Bouafia et al., 2023).

5.3.2 Automation and User Accessibility

Because Pexpect was integrated with SQLMap, which enabled automated command-line interactions, users could do scans without much technical knowledge. Studies that advocate for simplifying security technologies to encourage wider adoption among non-technical users align with this recommendation (Zhuang, 2023). The combination of automation and real-time feedback provided by the Celery server allowed the application to maintain a high degree of usability while conducting asynchronous scans.

5.3.3 Mitigation Strategies

Modern research validates the project's suggested mitigating strategies, which include web application firewalls (WAFs) and input validation. Researchers recognize parameterized queries and input validation as effective defences against SQLi, as they prevent the execution of malicious inputs as commands (Chukwuemeka, 2024). Numerous studies have validated the technique of WAFs, which provides an additional layer of security by filtering malicious inputs before they reach the application layer (Alotaibi & Vassilakis, 2023).

5.3.4 Limitations of the Methodology

Although the methodology was successful in addressing SQL injection vulnerabilities, its scope was limited to tackling this specific type of vulnerability. Two common vulnerabilities seen in web-based applications are remote code execution (RCE) and cross-site scripting (XSS). This limitation emphasises the need for a more all-encompassing approach, including other web application vulnerabilities. In comparison to other vulnerability scanners like OWASP ZAP and Nikto, SQLMap's results show that multi-tool integration usually results in a more thorough evaluation (Ibrahim & Rosli, 2023).

5.3.5 User Interface and Results Presentation

In particular, for users who are not technically savvy, the user interface was developed to make interaction simple and straightforward. To ensure that the results were easily understandable and instantly usable, the system classified vulnerabilities according to their severity and provided recommendations that could be implemented. Bar charts and tables are two types of data presentation that have been shown to improve user comprehension and decision-making (Baklizi et al., 2023).

5.3.6 Testing and Validation

Testing was performed on different numbers of web platforms, such as testphp.vulnweb.com, juice-shop.herokuapp.com, and itsecgames.com. The results of this testing demonstrated that the tool is capable of identifying high-severity SQL injection vulnerabilities in a consistent manner. For instance, tests revealed serious database exposure issues, which align with the findings of earlier research highlighting the prevalence of SQL injection in online applications and its impact (Ojagbule et al., 2018).

5.4 How Will the Proposed Mitigation Strategy Be Able to Resolve an Attack

The suggested mitigation strategy is a holistic strategy for fighting SQL injection (SQLi) attacks in that it utilises both preventive and reactive mechanisms aimed at protecting web applications from attacks at multiple levels. By focusing on proactive prevention and reactive measures, the strategy ensures a robust defence against potential threats.

Input validation and parameterised queries are used as the core of the preventive actions. Input validation guarantees that user-entered information conforms to specific formats and rejects any input that does not conform to the established standard. This process acts as the first line of defence, preventing malicious SQL code from being processed by the application. Parameterised queries enhance this protection by isolating the user input from the SQL code that is applied to the database. In contrast to conventional dynamic SQL, parameterised queries make malicious commands injected into such code useless, as the database externally codes user inputs merely as data. Research confirms that, in doing so, the technique has a dramatic effect on reducing the success rate of SQL injection attacks by attackers (Chandrachood, 2022).

The use of web application firewalls (WAFs) is a crucial line of defence. These systems are capable of monitoring and filtering incoming traffic that can be blocked from requests with SQLi attack patterns. Next-generation WAFs use machine learning and rule-based pre-construction of rules to identify and adjust to new LPEs. For instance, if an adversary uses obfuscated payloads, the WAF can identify patterns and signatures as the threat in real time and stop the request. Studies show that the combination of WAFs markedly improves a system's resistance to advanced SQLi attacks, especially for e-commerce systems (Gu et al., 2020).

Regular security audits and code reviews are fundamental for ensuring a web application's security. In these audits, tools like SQLMap reproduce the attack scenarios and target vulnerabilities before they are exploitable by attackers. Furthermore, manual review certifies that any application updates or new features do not introduce novel security vulnerabilities. With the integration of automated testing and human review, organisations can obtain an integrated strategy for the identification and defence of vulnerabilities (Bouafia et al., 2023).

After technical defences, secure development practices are crucial to long-term security. It is a learnt practice for developers to follow secure coding best practices, for example, not using dynamic SQL and using good error-handling constructs. This preventive strategy makes it possible to identify and deal with possible weaknesses during the development phase. In addition, secure development is also a part of culture to promote security competency in teams, reducing the probability of vulnerabilities that may harm attackers (Kaur, 2019).

The real-time monitoring and incident response mechanisms play an important role in containment if the attack is successful. Intrusion Detection Systems (IDS) are in constant use to analyse database activity for anomalous behaviour, i.e., repeated failed login attempts or unpredicted database queries. As soon as an attack is suspected, automatic alerts are generated, and the system immediately acts to restrict that access and prevent another exploitation. This fast response reduces damage and also offers useful data on post-incident analysis (Gogoi et al., 2021).

Post-attack recovery is one of the core elements of the mitigation plan. Following a breach, there are detailed evaluations of the attack vector and the patching of exploited vulnerabilities. What can be drawn from such events feeds into the security of the system as it evolves to meet the challenges of contemporary attacks. Continuously updating defence mechanisms, for instance, WAF rules or input validation procedures, also enhances the system's robustness to SQLi attacks (Aggarwal et al., 2021).

In conclusion, the proposed mitigation strategy integrates preventive measures, dynamic defences, and responsive mechanisms to address SQLI attacks comprehensively. Through a combination of such layers, the strategy guarantees a strong defence against present and future attacks, secures confidential information, and safeguards the integrity of the system.

5.5 Practical Implications for E-commerce Deployment

The developed tool can be applied in real e-commerce environments in several simple ways. First, it can be deployed as an internal web application that runs on a company server. In this case, staff members log into the interface, enter the URLs of their own web applications, and review the SQL injection results on a regular basis. This option is suitable for small and medium-sized businesses that do not have a dedicated security team but want a basic and repeatable security check.

Second, the tool can be integrated into an existing e-commerce management panel or dashboard. For example, an administrator could see a “Security Scan” tab next to other site management options. This would allow security checks to become part of normal website maintenance rather than a separate, specialist task.

Third, the tool can be connected to automated workflows such as scheduled scans. For instance, scans could be run weekly or after every major code change, and the results reviewed by a technical support person. This makes it easier to detect problems early instead of waiting until after a breach occurs.
In all cases, the clear presentation of severity levels and practical recommendations means that the tool can support decision-making. Even if remediation work is carried out by developers or external consultants, non-technical staff can still understand the basic risk level and prioritise which applications should be fixed first.

5.6 Limitations

While the tool performs well for detecting SQL injection vulnerabilities, it has some important limitations. First, the current implementation focuses mainly on SQL injection and does not cover other common web attacks such as cross-site scripting (XSS), cross-site request forgery (CSRF) or remote file inclusion. This means that a web application that "passes" the scan may still contain other serious vulnerabilities.

Second, the tool relies on SQLMap as the main scanning engine. Although SQLMap is powerful, it can be slower on large or complex applications, and some advanced configurations may still require expert tuning. The interface hides most of this complexity, but this also limits the level of fine-grained control that advanced users might want.

Third, the system assumes that the user has permission to scan the target URL. Running scans against third-party sites without consent would be unethical and may be illegal in some countries. The tool does not enforce this; instead, it depends on users acting responsibly.

Finally, the evaluation in this project was based on well-known vulnerable test sites and a limited number of targets. More extensive testing on a wider variety of real e-commerce platforms would provide a stronger picture of performance in practice.

5.7 Future Enhancements

There are several possible improvements that can be made to strengthen and extend the tool in future work. One important direction is to support additional vulnerability types such as XSS, CSRF and insecure direct object references. Integrating multiple scanners or adding custom checks would help to provide a more complete view of the security posture of an e-commerce application.

Another promising enhancement is the use of machine learning techniques to support automated detection and prioritisation. For example, simple models could be trained to group similar findings, highlight patterns that suggest higher risk, or predict which vulnerabilities are more likely to be exploited. This would help non-technical users focus on the most important results first.

The user interface can also be improved further. Future versions could include more interactive charts, filters to view results by date or application, and basic explanations or links to educational resources for each vulnerability type. These additions would make the tool not only a scanner but also a learning aid for users who want to understand security concepts more deeply.

Finally, integration with continuous integration and deployment pipelines would allow the tool to be used earlier in the software development lifecycle. Automatic scans during development and testing would help developers to fix issues before code reaches production, which is usually cheaper and more effective than fixing them after a system goes live.

Reference

Abdullah, H. S., Hamad, Z. O., & Khalind, O. S. (2023). *Analysis of SQLMAP Efficacy in Exploiting SQL Injection Vulnerabilities in Web Applications: A Case Study on DVWA*. *10*, 13–18. https://doi.org/10.1109/iceans58413.2023.10630454

Aggarwal, P., Kumar, A., Michael, K., Nemade, J., Sharma, S. and Pavan Kumar C (2021). Random Decision Forest Approach for Mitigating SQL Injection Attacks. *2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*. Doi:https://doi.org/10.1109/conecct52877.2021.9622689.

Al-Ghifari, M.F. and Azizah, F.N. (2022). *Development of Application for Entity-Relationship Diagram Conversion to Logical Schema of NoSQL Column Oriented*. [online] IEEE Xplore. Doi:https://doi.org/10.1109/ICoDSE56892.2022.9972074.

Alarfaj, F.K. and Khan, N.A. (2023). Enhancing the Performance of SQL Injection Attack Detection through Probabilistic Neural Networks. *Applied Sciences*, [online] 13(7), p.4365. doi:https://doi.org/10.3390/app13074365.

Alghawazi, M., Alghazzawi, D. and Alarifi, S. (2022). Detection of SQL Injection Attack Using Machine Learning Techniques: A Systematic Literature Review. *Journal of Cybersecurity and Privacy*, [online] 2(4), pp.764–777. doi:https://doi.org/10.3390/jcp2040039.

Aliero, M.S., Ghani, I., Qureshi, K.N. and Rohani, M.F. (2019). An algorithm for detecting SQL injection vulnerability using black-box testing. *Journal of Ambient Intelligence and Humanized Computing*, 11(1), pp.249–266. doi:https://doi.org/10.1007/s12652-019-01235-z.

Alotaibi, F.M. and Vassilakis, V.G. (2023). Toward an SDN-Based Web Application Firewall: Defending against SQL Injection Attacks. *Future Internet*, [online] 15(5), p.170. doi:https://doi.org/10.3390/fi15050170.

Alqahtani, S.S., Eghan, E.E. and Rilling, J. (2016). Tracing known security vulnerabilities in software repositories – A Semantic Web-enabled modelling approach. *Science of Computer Programming*, 121, pp.153–175. doi:https://doi.org/10.1016/j.scico.2016.01.005.

Baklizi, M., Atoum, I., Hasan, M.A.-S., Abdullah, N., Al-Wesabi, O.A. and Otoom, A.A. (2023). Prevention of Website SQL Injection Using a New Query Comparison and Encryption Algorithm. *International Journal of Intelligent Systems and Applications in Engineering*, [online] 11(1), pp.228–238. Available at: https://www.ijisae.org/index.php/IJISAE/article/view/2462? [Accessed 29 Dec. 2024].

Bouafia, R., Benbrahim, H. and Amine, A. (2023). Automatic Protection of Web Applications Against SQL Injections: An Approach Based On Acunetix, Burp Suite and SQLMAP. Doi:https://doi.org/10.1109/icoa58279.2023.10308827.

Chandrachood, A. (2022). Strategies for Protecting Against SQL Injection Vulnerabilities in Web Applications. *Journal of Artificial Intelligence & Cloud Computing*, 1(3), pp.1–3. doi:https://doi.org/10.47363/jaicc/2022(1)307.

Chukwuemeka, G. (2024). *(Teaching Aide) PACKET FILTERING FIREWALLS*. [online] Jacobson CPSC. Available at: https://wpsites.ucalgary.ca/jacobson-cpsc/2024/11/18/teaching-aide-packet-filtering-firewalls/.

Damele, B. and Stampar, M. (2024). *sqlmap: automatic SQL injection and database takeover tool*. [online] sqlmap.org. Available at: https://sqlmap.org/.

Dawadi, B.R., Adhikari, B. and Srivastava, D.K. (2023). Deep Learning Technique-Enabled Web Application Firewall for the Detection of Web Attacks. *Sensors*, 23(4), p.2073. doi:https://doi.org/10.3390/s23042073.

Foster, E.C. and Godbole, S.V. (2022). *Database Systems*. Doi:https://doi.org/10.1201/9781003275725.

Gogoi, B., Ahmed, T. and Dutta, A. (2021). Defending against SQL Injection Attacks in Web Applications using Machine Learning and Natural Language Processing. *2021 IEEE 18th India Council International Conference (INDICON)*. Doi:https://doi.org/10.1109/indicon52576.2021.9691740.

Gu, H., Zhang, J., Liu, T., Hu, M., Zhou, J., Wei, T. and Chen, M. (2020). DIAVA: A Traffic-Based Framework for Detection of SQL Injection Attacks and Vulnerability Analysis of Leaked Data. *IEEE Transactions on Reliability*, [online] 69(1), pp.188–202. doi:https://doi.org/10.1109/TR.2019.2925415.

Handge, N. (2022). *Understanding Celery Part 1: Why to use Celery? And what is Celery?* [online] Medium. Available at: https://medium.com/scalereal/understanding-celery-part-1-why-use-celery-and-what-is-celery-b96bf958cd80 [Accessed 16 Dec. 2024].

Herman, H., Riadi, I. and Kurniawan, Y. (2023). Vulnerability Detection With K-Nearest Neighbor and Naïve Bayes Method using Machine Learning. *International Journal of Artificial Intelligence Research*, 7(1), pp.10–10. doi:https://doi.org/10.29099/ijair.v7i1.795.

Kaur, P. and Kaur, N. (2019). Mitigation of SQL injection vulnerability during development of web applications. *International Journal of Web Science*, 3(2), p.104. doi:https://doi.org/10.1504/ijws.2019.10023828.

Kumar, M. and Nandal, R. (2024). *Role of Python in Rapid Web Application Development Using Django*. [online] Social Science Research Network. Doi:https://doi.org/10.2139/ssrn.4751833.

Kumari, A., Patel, R.K., Sukharamwala, U.C., Tanwar, S., Raboaca, M.S., Saad, A. and Tolba, A. (2022). AI-Empowered Attack Detection and Prevention Scheme for Smart Grid System. *Mathematics*, [online] 10(16), p.2852. doi:https://doi.org/10.3390/math10162852.

Kuroki, K., Kanemoto, Y., Aoki, K., Noguchi, Y. and Nishigaki, M. (2020). Attack Intention Estimation Based on Syntax Analysis and Dynamic Analysis for SQL Injection. *IEEE*. Doi:https://doi.org/10.1109/compsac48688.2020.00-41.

Liu, M., Li, K. and Chen, T. (2019). Security testing of web applications. *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. Doi:https://doi.org/10.1145/3319619.3322026.

Ojagbule, O., Wimmer, H. and Haddad, R.J. (2018). *Vulnerability Analysis of Content Management Systems to SQL Injection Using SQLMAP*. [online] IEEE Xplore.Doi:https://doi.org/10.1109/SECON.2018.8479130.

Pexpect (n.d.). *Pexpect version 4.8 -Pexpect 4.8 documentation*. [online] pexpect.readthedocs.io. Available at: https://pexpect.readthedocs.io/en/stable/.

Qin, X., Luo, Y., Tang, N. and Li, G. (2020). Making data visualization more efficient and effective: A survey. *The VLDB Journal*, [online] 29(1), pp.93–117. Available at: https://link.springer.com/article/10.1007/s00778-019-00588-3.

Saura, J.R., Ribeiro-Soriano, D. and Palacios-Marqués, D. (2021). From user-generated data to data-driven innovation: A research agenda to understand user privacy in digital markets. *International Journal of Information Management*, [online] 60(102331), p.102331. doi:https://doi.org/10.1016/j.ijinfomgt.2021.102331.

Soliev, B.N., Abdurasulova, D. and Yakubov, M.S. (2022). USING THE DJANGO FRAMEWORK FOR E-COMMERCE PROCESSES. *Journal of Integrated Education and Research*, [online] 1(6), pp.229–233. Available at: https://ojs.rmasav.com/index.php/ojs/article/view/625.

Yasmeen, G. and Afaq, S.A. (2023). *The Critical Analysis of E-Commerce Web Application Vulnerabilities*. [online] www.igi-global.com. Available at: https://www.igi-global.com/chapter/the-critical-analysis-of-e-commerce-web-application-vulnerabilities/325544.

Zhuang, Y.S. (2023). *Investigating effects of non-technical limitations on customer online shopping behaviour in the Namibian retail sector*. [online] repository.unam.edu.na. Available at: https://repository.unam.edu.na/handle/11070/3793.