THE USE OF MACHINE LEARNING MODEL FOR PREVENTIING AND DETECTING OF CROSS-SITE REQUEST FORGERY'S ATTACKS.

Ayodele Emmanuel & Hammed Mudasiru
Department of Computer Science, School of Pure and Applied Science The Federal Polytechnics Ilaro, Ogun State.
Email:Emmanuel.ayodele@federalpolyilaro.edu.ng

Abstract

The web-based Cross-Site Request Forgery (CSRF) attack stands as one of the most commonly encountered vulnerabilities that uses authorized users to execute unauthorized commands against reputable web applications. The available traditional mitigation methods which combine token-based verification with SameSite cookie attributes fail to stop modern advanced CSRF attack techniques. This study employs Artificial Neural Networks (ANNs) as a machine learning approach to detect and stop CSRF attacks while they happen. This research work presents a clever framework for detecting and defending against CSRF attacks in web applications that was constructed using Artificial Neural Networks technology. The system follows a modular architecture with three main parts which include user and admin interfaces together with a web application firewall layer that connects to an adaptive MySQL database. The experimental testing shows that this ANN detection system provides accurate results while being flexible through extensive performance and penetration tests which demonstrate its effectiveness in protecting dynamic web environments. The system learns continually to protect in the long term while embodying a transition towards defensive security strategies that operate proactively with intelligence. The paper ends by proposing deeper learning implementation along with expanded dataset utilization coupled with integrating ANN technology into multiple cybersecurity protection elements to enhance web application defense capabilities.

Keyword: web-based, Cross-Site Request Forgery, unauthorized, authorized, framework

INTRODUCTION

We have an unconventional and ever-increasing use of internet today which not only embraces purely social functions but also complex transactional activities. This popularity of engaging in the internet is slowly becoming possible thanks to the many internet users and devices available across the globe leading to the growth of a worldwide web. Unfortunately, this growth in the digital landscape has also attracted the rising number of extra-legal activities more so of the cyber space, wherein cybercrime and corporate hacking are common incidences (Elluri et. al. 2023).

The existence of these threats points out the need for effective cyber security. Meanwhile, the emergence of new technologies, especially those related to artificial intelligence (AI), deep learning, and machine learning (ML) seems to offer a good solution to this problem. Cutting-edge technologies of this kind may greatly improve the existing cyber security systems and thus make them capable of resisting a wide range of cyber-attacks (Karimy & Reddy, 2023).

Despite the improvements in the prevailing cyber security systems, the incidence of online crimes and attacks on websites remains very high in the world. Cases in point include the GitHub account hacking, a scare tactic of exposure of users sensitive information on Twitter, malware infiltrating WordPress plugins and security loopholes in Tomcat server. These include threats such as crypto jacking & ransom ware extortion schemes, and cross-site request forgery attacks (XSRF) and cross-site scripting (XSS) attacks have become two of the most common forms of cyber threats and inflict heavy damages (Berman et. al. 2019).These attacks exploit the vulnerabilities inherent in web applications and browsers, posing a significant risk to both individual users and organizations.

In today's cyber security plain detection and investigation of XSRF and XSS attacks becomes one of the tools that dictate the importance of performing these activities. In the ongoing threats media data protection and integrity of online system bodybuilding existing solutions has to come up with constructive approaches to these threats (Schreiber, 2020).

Cross site request forgery is a kind of an attack where the end user of a web application is made to perform actions on the web application that he or she is already authenticated in. Cross site request forgery attacks are known to be very diverse. They do not attack for requesting data from the server as the attacker can't view the forged request's reply. Social engineering, in simple terms, is the way an attacker engages the users of the web application with an activity which in turn accomplishes an

action of the attack. In case the user is not an admin, a working CSRF attack might 'fool' the user to make a status varying request in order to 'transfer' money to another email address, etc. Benchmarking if cutting across user levels, CSRF attack can bring the whole web application down if it is directed towards an administrative account (Jovanovic et. al., 2019).

LITERATURE REVIEW

The World Wide Web has become the most popular and cheapest form of communication across the world in the present day (Cao et al., 2021). The easy to use and the numerous services availed have made billions of individuals engage in it for various activities. More and more, people's interests are drawn to the likes of social networks, e-short stores, and virtual filings. In this case, the web is the primary tool, and one of its key features, which attracts internet users, is an appealing, well designed, and animated homepage (Luo et al., 2022).

In facilitating web experiences for users, server and client side scripts play a significant role. On the other hand, criminals or assailants seek to turn these applications to their advantage by creating direct or indirect ways of attacking the users of the network (Deng et al., 2022). For this reason, they aim to extract information such as usernames and passwords, sensitive information about individuals, session cookies, remote system access or the storage of malicious programs within the systems (Zhang et al., 2021).

Cross-site scripting or XSS has also emerged as one of the important threats payload for a majority if not all websites (Lee et al., 2022). In the statistical survey recently conducted by OWASP, XSS attacks are still the most harmful attacks. XSS topples broken access control, cryptography failures and is third faulty implementing security controls attackers made improvement in security controls from 7th position in 2017. Every application of two or more in practice is affected by the XSS problem which prompts the very high risk associated with it always.

Following are some of the XSS vulnerability threats (Schuckert et al., 2022): loss of regular users' website cookie data, reaching out to the browser's session data for any user, and impersonation of any user in order to perform malicious activities. These actions can freezes a website and take over normal users' systems for purposes of phishing attacks that solicit sensitive information from users, such as credit card details, using other users' computers for distributed denial of service attacks and introduction of worm virus codes into the network compromising the network security (Zhao et al., 2021).

In the field of network and information security, enhancing the detection of XSS vulnerabilities has attracted a lot of research attention (Kalouptsoglou et al., 2022). Still further, existing XSS detection techniques do not remain free from challenges. In feature engineering, too much time is taken to perform manual feature extraction which is further hampered by unavailability of expert lack of expert knowledge in the filing. Furthermore, it is hard to draw out the higher logical dimensions of intricate meanings (Zheng and Yin, 2022).

Web Application Security

Web application servers are notoriously difficult to secure because they must always be on standby to serve valid requests while denying access to fraudulent ones. However, users sometimes find it difficult to differentiate between a rogue user and a legitimate one. Blocking legitimate traffic can also be expensive. One promising way to defend a web application server is to deploy distinct security devices in a layered manner to filter unwanted incoming traffic. The Web Application Firewall is one of those defensive measures against outside incursion that is quite popular with many.

A WAF is application-layer firewall which examines all HTTP requests directed toward the web application and parts of WAF include HTTP filters which block harmful HTTP traffic. WAF can be described as an HTTP reverse proxy which inspects the content of the HTTP request being processed and returned by the proxy itself. WAFs are one of the Bard's defense mechanisms against the OWASP Top Ten that has been put to the test. WAFs have drastically changed over the past 20 years and the improvements include the integration of advanced analysis engine capable of identifying and mitigating other forms of attack including weaponized bots, credential stuffing and DDoS (distributive denial of service) attacks (Lawton, 2021).

Vulnerability scanning attacks are a form of attack in which a wide spectrum of HTTP requests is made to the web application in a systematic way to explore its weaknesses and vulnerabilities, and for reconnaissance purposes. Models that can detect and monitor anomalous and or malicious client behavior can be created by considering the client/server interactivity as a composition of some statistics including performance metrics, and threshold or rule-based models. A model of login attack classification may employ thresholds on the maximum number of login attempts or login failures before an alarm is raised.

Dynamic Application Security Testing

DAST or Dynamic Application Security Testing is a critical component of any application vulnerability management program and is regarded as a better alternative than WAF. WAF is applicable to filter user inputs and shield the application from external threats however there is need for constant identification and assessment of risk in order to mitigate rather than wait for exploitation of the threat. This will incorporate performing a penetration test to manual penetration test and include a dynamic application security testing tools or also known as DAST scanners.

Penetration testing is the term or practice of assaying a computer or a program, a network or an application for vulnerability that may be exploited by an intruder. It includes processes such as researching the target in advance to understand and seek possible weaknesses and then executing an attack mainly by tried methods such as SQL injections and cross-site scripting. The concluded report is then sent over to the concerned party, most often developers, to mitigate the risk posed by the discovered vulnerabilities. This too is an example of dynamic analysis since it involves running the program with inputs as opposed to a static programmatic assessment where the codes are not run at all.

Detection

In order to construct a robust deterrent system, several tendencies of the enemy attackers need to be established. The Common Attack Pattern Enumeration and Classification (CAPEC) is an encyclopedia of common approaches that can be used against software attacks aimed at helping one recognize attack patterns. The attack patterns are the definitions of the attacks that are already known to the enemies who know how to take advantage of the application and its flaws. These patterns along with OWASP Top Ten which are the risks facing web applications, most of them being security related, are the core components for designing the places where detections will take place, an event of interest during the running of the application.

To put it another way, AppSensor detection points are the application input validation patterns. Any action, which is beyond this periphery of the application would activate that detection point and create an AppSensor event. One of the examples of a detection point can be: the maximum output limit of an internal function which retrieves data from the database, or the input limit to this same internal function. If the size of the resulting data set is too big than normal, then such would be an event which would be recorded.

Static detection

Static analysis tools are strategies for detection that examine the program source code for possible flaws, without executing the program (Liu et al., 2019). The study by Shar and Tan (2012), presents a static analysis vision for counteracting XSS code attacks that is anchored on the pattern profile-based code transforming approach. Further, this approach enabled the mapping towards identifying areas in the code where XSS threats existed, hence being able to delete them. The drawback is that it addresses only the server, and is ineffective in countering DOM based cross-site scripting attacks.

Ahmed and Ali (2016) developed a genetic algorithm that generates XSS attack-related test data. The methodology involved three classes of XSS attacks being entered into a database along with data labelled as genetic X-creating maximum data with XSS attack types and their validation. This testing technique targets the applications designed with PHP and MySQL. The final testing outcome confirmed that the produced test data could successfully detect numerous forms of XSS attacks.

Dynamic detection

Dynamic detection involves executing the program under test with the usage of sample inputs and evaluating the output results and the response content of the web page that was served by the web server. If the response contents have specific information, such a vulnerability is present (Hou et al., 2018). Fazzini et al. (2015) introduced an automation framework for web applications based on view-oriented user tasks. This approach is also composed of four layers: dynamic testing, web page parsing, Csp policy analysis and source code transformation.

Acquired the web application and also the documented test case according to csp, the trustworthiness of the encoded it in the server codes was compromised, and the web program execution was performed through dynamic detecting analysis. The Results proved that it could help revealing the XSS attack vulnerabilities.

Cross-site Request Forgery

CSRF is a type of attack where the assailant convinces the target to execute a harmful request. The attacker borrows the identity and the permissions the target has to take certain actions which the attacker does not intend to perform, to take actions on the victim's behalf. Most of the times

in most of the applications, the browser will by default append making use of the applications' authenticated user data such as session cookies, the users' IP address, the windows domain credentials, and many others. This means it is a lost course for the site if a user is already authenticated in the site because there is no way the site will know that the victim has sent a forged request and not an actual request from the victim. CSRF attacks target certain functions that achieve state changes in the server for instance when a client snake changes its password or email address, or even when a purchase is made.

The data in which the attacker is interested is usually external, so forcing the victim to provide it is of little use to the attacker, as the attacker gets no reply, the victim does. So, in CSRF attacks, there is a request to change the state of affairs. Save a CSRF attack in the most expose site can sometimes be done. These weaknesses are known as `persistent CSRF vulnerabilities'. This can be done by putting simply an IMG or IFRAME tag in a HTML accepted field or even more advanced, using a cross-site scripting attack. If an attack has the ability to keep a CSRF attack in a target, that target becomes easier to attack for the attacker.

Modern methodologies aimed at the detection and prevention of Cross-Site Request Forgery (CSRF) and Cross-Site Scripting (XSS) attacks are developed. These attacks are some of the most vicious threats afflicting the cyberspace (Yang et. al., 2023). This clearly explains this study's objective to improve web security through provision of technological advances. The essence of the methodology lies in the building up and cleansing of data. This means collecting various types of data including those that describe the attack inclusions of CSRF and XSS in many different ways, such data being gathered from multiple web applications.

In order to mitigate the risk of CSRF, the developers of web applications have to put in place certain specific measures against it (Madan et. al., 2023). Where it is possible to add extra user action without compromising on usability too much, it may be possible to force a repetition of the authentication process or use OTPs/CAPTCHAS so that cross-origin requests do not go unnoticed. However, in most scenarios, passive or automated means of protection is preferred: as of now, the newest Same Site cookie attribute serves to restrict cookies from being included in cross domain requests, thus addressing the main issue of CSRF and should be used for all newly developed web applications. Sadly, this remedy is not common practice and most existing web applications generally prevent cross-site requests utilizing one or more of the following methods:

- That is, checking the values of standard HTTP request headers (for example the Referrer and Origin headers) associated with the page making the request.
- That is, checking presence of custom HTTP request headers, cross origin X-Requested-With for example, which cannot be set otherwise.
- Finally, checking for existence of anti-CSRF tokens that are difficult to predict and are added by server to secure forms. The recent paper mentioned above gives analysis of cons and pros of such assorted techniques (Elluri et. al., 2023).

Using a token for a 'like' button is effective to counter the attack outlined above; however, it is not ideal to have a token on the homepage of a social networking site as it may unnecessarily cause the blocking of valid cross site requests such as those from users who visit the social networking site after searching for it using a search engine. Ultimately, many times, web developers face the challenge of finding the 'sweet spot' where adequate anti-CSRF measures can be placed (Khakimov, 2023).

Machine Learning to the Rescue

As demonstrated by the CSRF example in the above section, it is beneficial to augment the vulnerability detection tools with additional semantic information in order to reduce the number of false positives and false negatives in its results. At the minimum, one would wish to devise a technique that would help in categorizing all HTTP requests into security sensitive and non-security sensitive requests such that the analysis is only done on security sensitive requests. Nonetheless, this proves to be especially difficult on the World Wide Web, because requests written in Hypertext Transfer Protocol have a shallow level of syntactic resolution and there are lots of freelance practices (Ahmad et. al., 2021). For instance, there are quite a number of valid implementations of a button that enables "liking" a piece of content whose unique string is 3aa5bf. These include:

- Making a GET request to a page like.php that has one parameter id = 3aa5bf.
- Making a GET request to a page manage.php that supplies an id = 3aa5bf along with another parameter action = like.
- Making a POST request to the page manage.php that includes a JSON string such as {id: 3aa5bf, action: upvote}.

To an experienced security tester, all these requests seem to convey the same information; however, they can be represented explicitly in different ways that

could make it difficult to represent all the known variations out there (Ahmad et. al., 2021).

Artificial Neural Networks

Artificial Neural Networks are composed of artificial components known as units, which correspond to biological neurons. These units are structured in a number of recognizable layers that together make the entire Artificial Neural Network system. A layer may consist of up to a dozen units or go to millions of them since this will depend on how sophisticated the neural networks will have to be in order to learn the concealed relationships in the data.

Typically known as an ANN, the structure is also composed of an input layer, an output layer, and one or more hidden layers. The input layer consists of the data provided to the neural net in order to analyze or learn something. Afterward, this data is processed in one or more hidden layers that convert the data into a useful form for the output layer. Finally, the output layer produces a response which, in the case of Artificial Neural Networks, corresponds to the input data offered.

METHODOLOGY

Organizations face difficulties in securing their web applications because of complex web systems along with harmful scripting languages that avoid automatic code assessment. The detection of vulnerabilities through static techniques still provides some useful functionality despite their known difficulties. The research adopts a systematic review model that carries out planning and reporting stages alongside conducting activities by applying quantitative metrics to evaluate machine learning modeling results through accuracy and precision and recall measures.

The process of dataset labeling for training and testing requires manual classification which is influenced by exploratory data analysis together with qualitative methods. This research work utilizes the waterfall model as its software development life cycle (SDLC) framework which guides progression from system design through to support functions. The system combines hardware and software elements for its interface which relies on HTML CSS and JavaScript programing languages.

The collection of data involved multiple qualitative research techniques that included, Interviews with domain experts, Questionnaires targeting student engagement, Internet research for secondary data, both published and unpublished reference materials served the research as written resources. The preprocessing of datasets in mining plays a fundamental part for both improved model outcomes and simplified training operations. The abstracted material from Kaggle provides several examples tagged for CSRF and XSS detection purposes. The process uses three natural language processing techniques which include encoding and tokenization with vector embedding methods. The data distribution method separates data into three parts which support exact model measurement and prevents data confusion during model building while measuring final model effectiveness.

Overview of the System Design

This model for detecting and preventing cross-site request forgery vulnerability is created for the benefit of everyone. The application is developed in python programming language. This system consists of artificial neural network based machine learning system. The Artificial Neural Network model developed for the purpose of CSRF detection consists of parameters that illustrate the architecture of the network; number of layers, number of neurons in each layers and respective activation functions, training methods and predicting new requests methods.

The data set used is split into parts known as training data and testing data. Out of that, 75 parts of the data set are used for training purposes. The other parts of the dataset are used for testing purposes. Some of the algorithms are used for testing the data to check the performance of the model. The optimal model is chosen regarding the training and test datasets. The training data set is different from the test data set and the output is fed into the program model.

Database Specification and Design

It is important to offer an effective mechanism for storing and preserving sensitive data. Such a data storage module is termed a database. The DBMS utilized for this work is MySQL server. The database developed for this system is called 'csrf' and comprises around 8 tables. Depicted below is the structure of a part titled as 'user' shown in Table 3.1.

Field name	Field length	Field type
id	11	Int
mat_number	300	VarChar
password	300	VarChar
date_created	14	Date

Table3.1user's table

Flowchart of the system

The system has a flowchart that explains its functioning and is divided into various parts based on the interface provided and the actions each of the system users is able to perform (Bohl, 2019). The application was designed and built in various stages as detailed in the figure 3.3 below. All the design and development works that have been finished are referred to as Achievements. The following paragraphs elaborate every stage certainly.

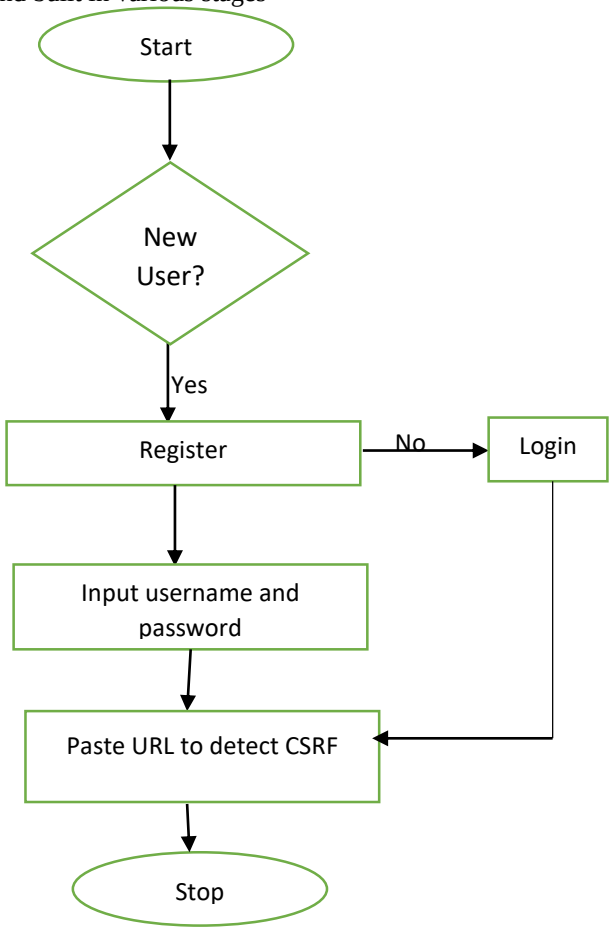

Figure 3.3 Flowchart Diagram

Data Flow Diagram

A Data Flow Diagram (DFD) is a chart that depicts the progression of information within a system. It shows how data travels among the various system elements, such as processes, data storage, outside entities, and data movement. The principal aim of a DFD is to depict the processing and movement of data in a precise manner so as to assist the users in grasping how the system operates and its relationships.

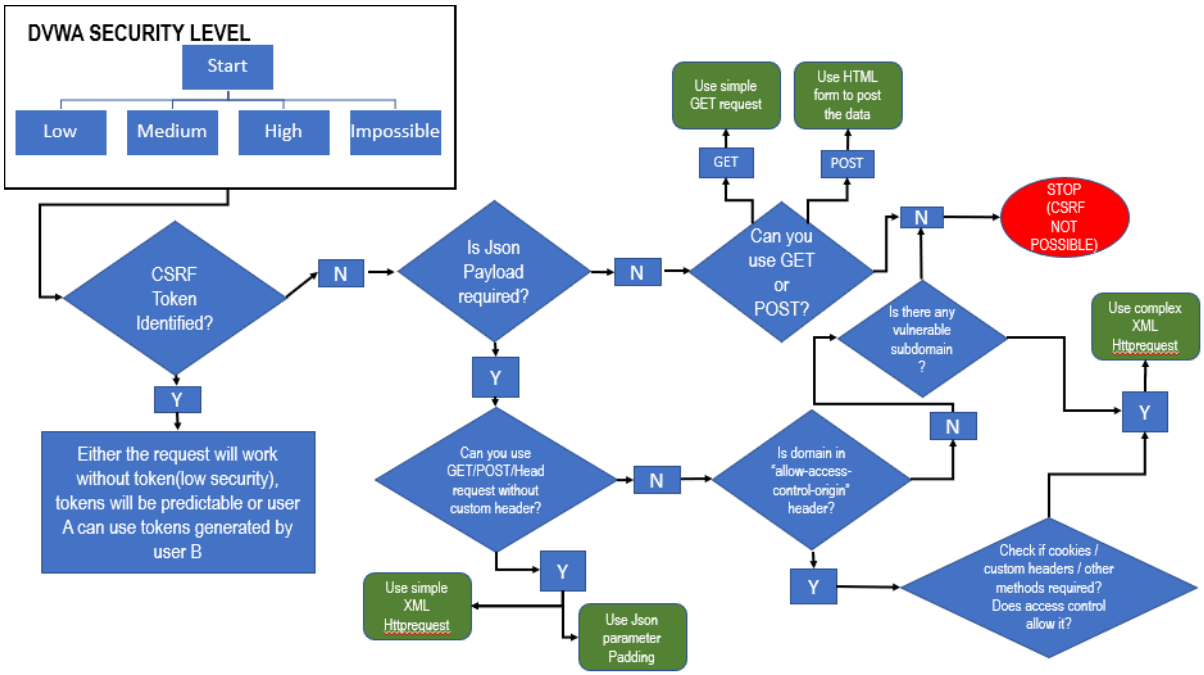

Figure 3.6. The Data Flow Diagram of the System

3.6.4 System Architecture

This design for a 2-tier system architecture outlines each level of the application as well as its components. This design encompasses the various operational layers such as client layer, end-user layer, and the application layer where the main application is used. The database level includes the other physical part of the data management of the system, the database level describes the logical aspect of the system which contains the records of the system.

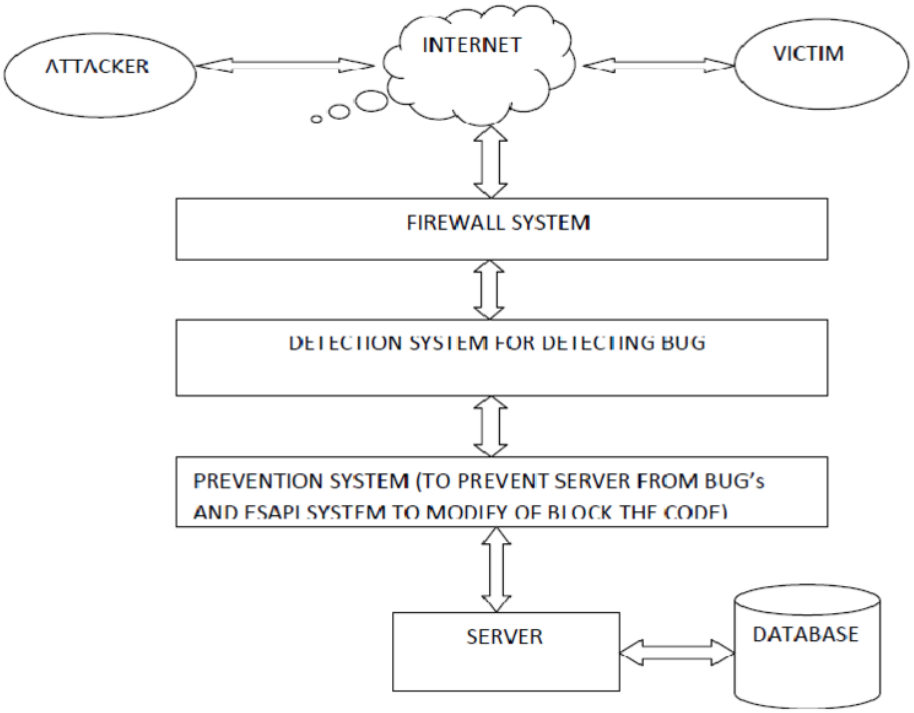

Figure 3.7. The Architectural Diagram of the System

DISCUSSION

In web application security, Cross-Site Request Forgery is an easily exploitable security vulnerability which in turn uses the trust that a user has for a site for unauthorized actions. These actions may lead to data loss, transfer of money, and theft of user credentials.

Conventional counter measures against CSRF attacks undertaken by the research include the use of anti CSRF tokens. These countermeasures have been efficient but present some drawbacks especially with the advent of new forms of attacks. The research work attempts to deal with CSRF attacks by making use of Artificial Neural Networks (ANNs) so as to predict them prior to an occurrence. The research work takes advantage of the ability of ANN to recognize patterns to spot user and request behaviors that are out of the ordinary and are likely to cause CSRF attacks.

This project is composed of the following stages:

1. **Research stage:** A collection of data had been made which included both legitimate request patterns and malicious request patterns.
2. **Analysis stage:** Parameters like the source of the requests, the status of the sessions, the referrer headers and the actions patterns were analyzed.
3. **Model training:** A trained ANN classifier was some preliminary distinguishing between harmful and no harmful CSRF attacks trained on these matrices.
4. **Infrastructure development:** In order to provide protection against CSRF attacks, the ANN model internally incorporated within the architecture of the web application.

According to the experimental outcomes, the ANN-based detection system is quite efficient such that the number of CSRF attacks is brought down to a great extent as opposed to tokens methods which are more accurate and faster.

CONCLUSION

Utilizing Artificial Neural Networks to detect and mitigate Cross-Site Request Forgery (CSRF) security risks is a worthy development in the field of

web security. Based on the knowledge gained from legitimate and fake web traffic requests the neural network model was able to detect irregular behaviors in a more effective manner that traditional methods in mitigating CSRF risks like anti-CSRF tokens.

The ANN based technique has some benefits:

- Ability to improve detection accuracy due to the analysis and aberrations in request behavior that may not be possible with the other techniques.
- Prevention takes place in real time, hence making it feasible for use in active web instances.
- The ANN model is retrained with new datasets cutting down on the time to deploy the model but ensuring acceptable performance as the model learns new attack patterns.

In spite of these benefits, the research also underlined some limitations such as the difficulty of fitting the modulus because of the scale of the population requiring well formatted and structured entities.

RECOMMENDATIONS

Although the ANN model's performance has been quite encouraging, other techniques (e.g., deep learning models, hyperparameter optimization) would be installed to improve detection performance while reducing false positives. The approach can be further developed by adding large and various datasets to enhance the system's applicability to various web applications. In addition, the ANN architecture is likely to be adjusted for faster and more accurate detection, especially under high traffic conditions. Other means of augmenting the capability of the system are provided below:

1. **Dataset Expansion:** Enriching the dataset with a wider range of newer CSRF variants would help in making the model more resilient. It is important to keep on tracking new forms of attacks to keep the model relevant.
2. **Usage with Other Security Solutions:** As effective as the ANN-based detection is, it is wise to include other prevention solutions such as Strong Encryption, Secure Coding Standards, and routine assessments of security to augment protection against web attack vectors.
3. **Maintenance and Updating of the Model:** With the development of more advanced attack methodologies, it is necessary to update the dataset and ANN architecture and apply them to the application after a specified period.
4. **Training the Users:** Users tend to practice best security methods such as avoiding phishing and thus make them less susceptible to CSRF attacks. Threats enhancing their technical sides with the education of the users eliminate any gaps in defense.

If these steps are taken, it will also contribute to improving the existing system in terms of CSRF attacks detection and prevention without compromising accuracy and flexibility with respect to the changing web security threat landscape.

References

Ahmad, A., Khan, M., Cheah, Y. Y., Abdullah, A., & Ahmad, F. (2021). *Machine learning based and deep learning based network intrusion detection systems: A review*.

Ahmed, N., & Ali, H. (2016). *A Genetic Algorithm for Detecting XSS Attacks*.

Alma, S., & Das, D. (2020). *Deep learning-based method for detecting attacks in web applications*.

Berman, D. S., Buczak, A. L., Chavis, J. S., & Corbett, C. L. (2019). *A survey of deep learning methods for cyber security*.

Bhandari, A. (2023). *Exploratory Data Analysis in Machine Learning Models*.

Bohl, J. (2019). *Flowchart design in software engineering*.

Cao, Y., Xu, W., & Liu, L. (2021). *Trends in web communication and their impact on cyber threats*.

Choudhury, D. (2019). *Software Development Life Cycle Models*.

Deng, H., Xie, T., & Li, J. (2022). *Modern attack vectors in web applications*.

Elluri, D., Patel, N., & Jain, R. (2023). *Review of Web Application Attacks and CSRF Mitigation Techniques*.

Fazzini, P., Chen, X., & Orso, A. (2015). *Automated Web Application Testing Framework*.

Hou, D., Li, Y., & Zhang, Y. (2018). *Dynamic detection of XSS vulnerabilities using response analysis*.

Jovanovic, N., Kruegel, C., & Kirda, E. (2019). *Preventing Cross-Site Request Forgery Attacks*.

Kalouptsoglou, I., Ntantogian, C., & Xenakis, C. (2022). *Survey on XSS detection techniques in web applications*.

Karimy, A., & Reddy, N. (2023). *Emerging Technologies for Cyber Security: AI and ML Applications*.

Kaur, S., & Garg, D. (2021). *Machine Learning and Deep Learning Methods for Web Vulnerability Detection*.

Khakimov, I. (2023). *Modern CSRF Prevention Strategies and Implementation Challenges*.

Lawton, G. (2021). *Evolution of Web Application Firewalls and Security Testing*.

Lee, J., Kim, S., & Park, C. (2022). *Cross-site scripting attack trends and mitigation strategies*.

Liu, J., Zhang, Z., & Wu, Y. (2019). *Static analysis for web security: An overview*.

Luo, W., Jin, C., & Wang, H. (2022). *User behavior and web security interfaces*.

Madan, A., Verma, P., & Sinha, A. (2023). *CSRF Mitigation in Modern Web Applications*.

Mehryar, M., Rostamizadeh, A., & Talwalkar, A. (2012). *Foundations of Machine Learning*.

Schreiber, D. (2020). *Strategies to Detect and Defend Against XSRF and XSS*.

Shar, L. K., & Tan, H. B. K. (2012). *Pattern-Based Static Code Analysis for XSS Attacks*.

Schuckert, M., Hoffmann, C., & Berger, T. (2022). *The Impact of XSS on Web Application Security*.

Yang, L., Wu, Z., & Feng, M. (2023). *CSRF and XSS Attacks: Current Defense Trends*.

Yan, L. (2021). *Using Machine Learning for Detecting XSS Attacks*.

Zhang, L., Wang, Y., & Tan, J. (2021). *Security Threats in Modern Browsers and Web Applications*.

Zhao, X., Liu, Q., & Li, H. (2021). *Modern Threats to Web-Based Systems and XSS Vulnerability Impacts*.

Zheng, R., & Yin, Y. (2022). *Advanced Feature Engineering Challenges in Security*.

Zhou, J. (2019). *Cross-site scripting detection using HMM and MLPs*.