National College of Ireland

SSRF threat detection using AI/ML

MSc Research Project
MSc. in Cyber Security

Laxmi Kumthe
Student ID: x23126001

School of Computing
National College of Ireland

Supervisor: Mr. Vikas Sahni

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Laxmi Bhaskar Kumthe
Student ID:	x23126001
Programme:	MSc. in Cyber Security **Year:** 2023-24
Module:	H9PRAC – Research in Computing
Supervisor:	Mr. Vikas Sahni
Submission Due Date:	12th Aug 2024
Project Title:	Research Report
Word Count:	10356 **Page Count:** 22

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Laxmi

Date: 12/08/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

SSRF threat detection using AI/ML

Laxmi Kumthe - x23126001

Abstract

Server-Side Request Forgery (SSRF) vulnerabilities present serious security threats to web applications, allowing attackers to use these applications as gateways to gain unauthorized access to internal services or execute arbitrary commands. Although SSRF was recognized as a distinct threat in the 2021 OWASP Top 10 list of web security risks and has become more common in contemporary web applications, there is still a notable deficiency in systematic methods for detecting these vulnerabilities effectively. In this study, a machine learning model was developed that integrates Randomforest and Convolutional Neural Networks (CNNs) with Long Short-Term Memory (LSTM) networks for detecting Server-Side Request Forgery (SSRF) attacks to analyze and classify attack patterns effectively. The study involved preprocessing a comprehensive dataset, including feature selection and data balancing, to train the CNN-LSTM model, which achieved an accuracy of 99.65%. In comparison, a RandomForest model was also trained, achieving an accuracy of 98.41%. This high level of accuracy underscores the model's effectiveness in distinguishing between attack and non-attack scenarios. The GUI provides detailed feedback and logs of the entire prediction process, including preprocessing steps and the detection outcomes. An additional feature includes automated email alerts to notify users of detected attacks, enhancing the system's responsiveness.

Keywords - SSRF, AI/ML, Web application security, RandomForestClassifier, CNN-LSTM, SMOTE, Chi2.

1 Introduction

1.1 Background

Machine learning (ML), a branch of AI, enables systems to learn from extensive data, identify patterns, and make precise predictions or decisions. This adaptability makes it ideal for malware detection. By training on large datasets of both clean and malicious files, ML models can identify subtle features that differentiate benign software from malware. This capability is crucial for addressing the evolving nature of threats. For example, if an account is compromised, it might show unusual network activity or connect to suspicious servers. ML can detect these anomalies and alert security analysts. Additionally, ML can identify system-level irregularities, such as unexpected privilege escalations or unusual system usage.[1] Machine learning can significantly enhance the detection of Server-Side Request Forgery (SSRF) attacks by identifying patterns and anomalies that traditional methods might miss. By training machine learning models on vast datasets of both legitimate and malicious traffic, machine learning algorithms can learn to distinguish between normal and suspicious activities. These models can then analyze incoming requests in real-time, flagging those that deviate from typical behavior for further investigation. Machine learning can adapt to new attack vectors as they emerge, continuously improving its detection capabilities. This proactive approach helps

[1] https://www.akkio.com/post/malware-detection-using-machine-learning

in early identification and mitigation of SSRF attacks, safeguarding systems from potential breaches.

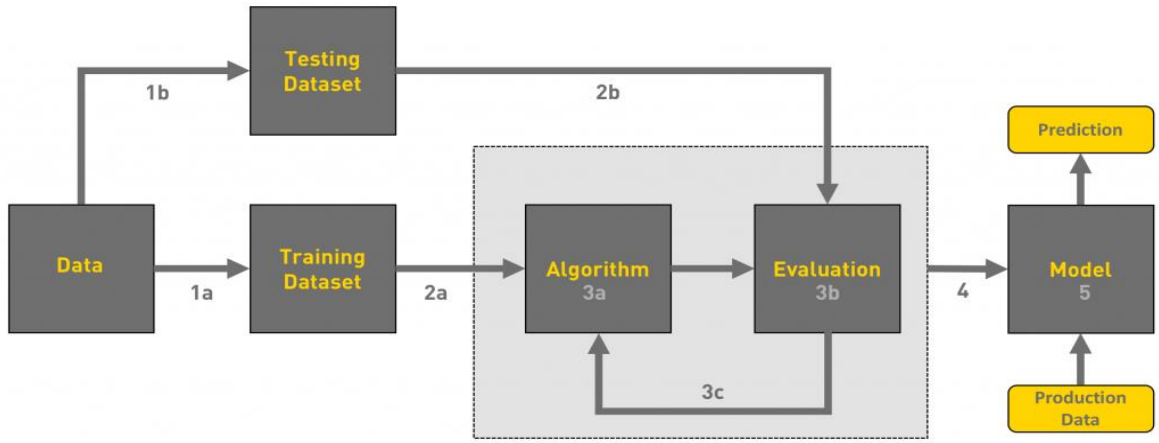

Figure (1): Shows simple architecture of machine learning model.[2]

Figure (1) shows a machine learning (ML) architecture diagram that outlines the process from data collection to prediction. Initially, raw data is collected and divided into two subsets: a training dataset for teaching the algorithm and a testing dataset for evaluating performance. The training dataset is fed into an algorithm, which learns to recognize patterns. The model is then evaluated using the testing dataset to measure its accuracy and ability to generalize. Based on the evaluation results, the model may be iterated and refined. Once sufficiently trained and evaluated, the final model is deployed to make predictions on new data. This structured approach ensures the development of robust and accurate ML models.

A Server-Side Request Forgery (SSRF) attack occurs when an attacker leverages server functionality to access or alter internal resources by manipulating the URLs utilized by the application. By supplying or altering a URL, the attacker can trick the server into reading or submitting data to unintended locations, potentially gaining access to internal services and data not meant for public exposure, such as HTTP-enabled databases and server configurations. This attack leverages the server's ability to handle URLs to bypass normal access controls and access sensitive information.[3] According to Bitdefender Labs, SSRF attacks have surged since late November 2022, targeting over 100,000 organizations globally, with most victims in the US and Europe, across various sectors including arts, consultancy, legal, manufacturing, real estate, and wholesale. The attacks exploit two chains, ProxyNotShell and OWASSRF, involving CVE-2022-41080 and CVE-2022-41082 vulnerabilities in Microsoft Exchange, with OWASSRF bypassing previously implemented mitigations. Bitdefender emphasizes a defense-in-depth strategy, focusing on patch management, detection of misconfigurations, multiple layers of security, and strong detection and response tools to protect against evolving cyber

[2] https://towardsdatascience.com/workflow-of-a-machine-learning-project-ec1dba419b94

[3] https://www.imperva.com/learn/application-security/server-side-request-forgery-ssrf/#:~:text=A%20Server%2DSide%20Request%20Forgery

threats.[4] Server-Side Request Forgery (SSRF), ranked 7th on the OWASP API Security Top 10 as of June 2023, occurs when a user-controlled URL passed through an API is processed by the back-end server. This creates a security risk if the server attempts to connect to the user-supplied URL, potentially allowing SSRF exploitation.[5] Various methods, primarily involving machine learning techniques, have been employed to detect and mitigate these attacks (Al-talak and Abbass, 2021).

A successful SSRF attack can lead to unauthorized actions or data access within an organization, affecting the vulnerable application or other back-end systems it communicates with. In some cases, the vulnerability may allow attackers to execute arbitrary commands. Additionally, SSRF exploits that connect to external third-party systems can result in malicious attacks that appear to arise from the organization hosting the application that is vulnerable.[6] This study is to develop a machine learning model that can accurately detect and prevent Server-Side Request Forgery (SSRF) attacks using AI and ML techniques. This study will employ publicly available datasets to train and validate the models. The graphical user interface (GUI) will be created using Python's Tkinter, allowing users to input data and view the prediction results on the same page. This approach involves pre-processing the dataset by removing null values, duplicates, and unnecessary columns, followed by feature selection and scaling. This study will then split the dataset into training and testing sets. Two machine learning models such as RandomForestClassifier and CNN-LSTM will be used to train and evaluate the dataset, with the model demonstrating higher performance being selected for prediction tasks. The final model will predict whether inputs are normal or indicate an attack, with results displayed in the GUI.

1.2 Aim and Importance

The aim of this study is to develop a machine learning model capable of accurately detecting SSRF attacks and distinguishing malicious requests.

This study makes a significant contribution by advancing the detection of Server-Side Request Forgery (SSRF) attacks through the application of an ensemble model combining Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) techniques, alongside a Random Forest model. By integrating deep learning techniques, this study enhances the accuracy and reliability of SSRF attack detection, surpassing the performance of traditional machine learning models. The performance of the machine learning models is also enhanced due to the feature selection method used in the study ensuring an effective attack detection model (Pasha and Mohamed, 2022).

1.3 Research questions

The study has the following research questions:

1) What machine learning algorithms can be used to perfect the detection of Server-Side Request Forgery (SSRF)?

---

[4] https://www.computerweekly.com/news/252529519/SSRF-attacks-hit-100000-businesses-globally-since-November

[5] https://salt.security/blog/api7-2023-server-side-request-forgery

[6] https://portswigger.net/web-security/ssrf

2) Is it possible to improve the model's ability to detect SSRF anomalies using machine learning by utilizing SMOTE and Chi2 feature selection?

1.4 Objectives

The objectives of the study are:

1. Pre-process the dataset by removing null rows and columns, duplicates, and unnecessary columns.
2. Select relevant features using Chi-squared (Chi2) feature selection.
3. Train the dataset using a RandomForestClassifier, CNN-LSTM models and evaluate its performance using various evaluation matrices.
4. Select and save the model with higher performance for prediction tasks.
5. Implement a GUI using Python's Tkinter to allow users to input data and view prediction results.

1.5 Limitations

A limitation of this study is its lack of live monitoring capabilities for attack detection. While the study successfully employs machine learning models and integrates them into a user-friendly GUI for offline analysis and prediction, it does not address real-time monitoring of network traffic or domain activities. The current setup relies on user-input data and pre-existing datasets, which limits its applicability to real-time threat detection and response. Live monitoring is crucial for promptly identifying and mitigating attacks as they occur, and the absence of this feature reduces the system's effectiveness in dynamic, real-world scenarios. Implementing real-time monitoring would require additional infrastructure and techniques, such as continuous data capture and analysis, to provide immediate feedback and alerts. Addressing this limitation could significantly enhance the system's capability to detect and respond to threats in real-time, offering more protection against evolving cyber threats.

Another limitation of this study is its focus solely on binary classification, which restricts the model to distinguishing between only two classes: attack and benign. This approach does not extend to classifying different types of attacks, which could be crucial for a more nuanced understanding and response to various threats. In a real-world scenario, differentiating between types of attacks such as SQL injection, cross-site scripting, or SSRF is essential for implementing targeted security measures and response strategies. The current binary classification model simplifies the problem but lacks the granularity needed for a comprehensive security assessment. Incorporating multi-class classification would allow for the identification and categorization of multiple attack types, enhancing the overall effectiveness of the system.

1.6 Assumptions

It is assumed that the feature selection enhances the performance of the machine learning model used for the detection of SSRF attacks.

1.7 Report Structure

The report has the following sections:

Section 1 Outlines the reason behind developing a machine learning approach for detecting SSRF anomalies, including the aim and specific objectives.
Section 2 Provides a review of current literature on the use of machine learning for identifying various attacks.

Section 3 Details the methodologies employed in creating the SSRF anomaly detection system using machine learning techniques.
Section 4 Covers the design specifications for this study.
Section 5 Discusses the implementation process of the developed system.
Section 6 Presents the results achieved from the system and evaluates its effectiveness.
Section 7 Concludes with a summary of the main methods used and proposes potential future enhancements.

2 Literature Review

A review of the different literature associated with detecting various attacks using machine learning is presented in this section.

2.1 Cyber attack detection using Machine Learning

A new web-based architecture for protecting web applications against CSRF attacks in a malicious environment was proposed in this study by (Srokosz, Rusinek and Ksiezopolski, 2018). The data consists of the proceedings of a federated conference on computer science and information systems. The study also extends the classic, static WAF approach with behavioral and historical analysis, based on actions done by the user in the past. The results show that the proposed approach reduces the need for additional forms of authorization, increasing the responsiveness and general feel of the application. This study highlights the importance of incorporating behavioral analysis into security measures, which enhances its robustness and adaptability to different attack patterns. This study uses a WAF approach combined with behavioral and historical analysis, rather than utilizing machine learning techniques, which could potentially enhance the system's ability to adapt to new and evolving CSRF attack patterns.

A framework called JAW that leverages hybrid property graphs (HPGs) to detect and analyze client-side Cross-Site Request Forgery (CSRF) vulnerabilities in JavaScript-based web applications was employed in this study by (Khodayari and Pellegrino, 2021). The data collected consisted of 4,836 web pages from 106 popular web applications, covering over 228 million lines of JavaScript code. JAW also generated symbolic models for 31 commonly used JavaScript libraries to improve the efficiency of the analysis. The study used declarative graph traversals on the constructed HPGs to identify forgeable client-side requests. The results showed that JAW uncovered 12,701 forgeable client-side requests affecting 87 web applications, and successfully created client-side CSRF threats against 203 forgeable requests in 7 web applications. The study also identified 25 distinct request templates that highlight the fields that can be controlled and the type of manipulation. This study demonstrates the effectiveness of using declarative graph traversals and hybrid property graphs (HPGs) to detect and analyze client-side CSRF vulnerabilities in web applications based on JavaScript, uncovering thousands of forgeable requests and identifying manipulable request templates. The main limitation of the study is that the vulnerabilities found are bound by the properties associated with the soundness of the static analysis tools used for building the property graphs. CSRF attacks have been detected using machine learning methods in several other studies (Calzavara et al. (2019); Agrawal, (2023); Kumar (2022); Muddham and Srimathi, (2022)). Machine learning techniques are also effective in detecting XSRF and XSS attacks (Kshetri, et al. (2024); Rathore, Sharma and Park, (2017); Kaur, Garg and Bathla (2023); Kaur and Garg (2021)).

Machine learning and artificial intelligence techniques were used to analyze cyber security datasets which was proposed in this study by (Yavanoglu and Aydos, 2017). The datasets used in the study include KDD Cup 1999, ECML-PKDD 2007, ISOT, HTTP CSIC

2010, CTU-13, and ADFA. Additionally, the study also discussed other datasets such as UNSW-NB15 and their characteristics. The study employed various machine learning algorithms such as Support Vector Machines, Naive Bayes, Decision Trees, and Genetic Algorithms, among others, to detect different types of cyber attacks like DDoS, botnets, and web application attacks. The results show that the proposed models achieved high detection rates, with one study reporting an accuracy of up to 98% on the normalized CAIDA dataset. This study highlights the effective use of various machine learning algorithms on multiple cybersecurity datasets to achieve high detection rates for different types of cyber attacks. However, the study acknowledges the limitations of the datasets, as they may not fully represent the characteristics of modern computer systems.

Machine learning (ML) and deep learning (DL) algorithms were used to detect cyber-attacks which was proposed in this study by (Dasgupta, Akhtar and Sen, 2020). The dataset used in the study consisted of various types of cyber-attacks, such as Denial of Service (DoS), probe, User-to-Root (U2R), and Remote-to-Local (R2L) attacks, along with normal network traffic. The study also employed feature extraction and selection techniques to improve the performance of the ML and DL models. The results showed that the proposed methods achieved high accuracy, with the best-performing model, a hybrid approach combining Artificial Neural Network (ANN) and Support Vector Machine (SVM), achieving an accuracy of 98.13% on the NSL-KDD dataset. However, the study acknowledged the limitations of the work, particularly in addressing zero-day attacks and the need for more comprehensive and representative datasets to ensure the generalization of the proposed techniques.

The application of machine learning techniques in the field of cybersecurity, focusing on prediction and classification tasks was discussed in this study by (Torres, Comesaña and García-Nieto, 2019). The data used in the study included various types of cybersecurity threats, such as malware, phishing, and spam. In addition to the primary machine learning methods, the study also discussed other approaches, including clustering, self-organizing maps, and decision trees. The results showed that support vector machines (SVMs) and Naive Bayes classifiers performed well in detecting and classifying cybersecurity threats, with accuracy rates ranging from 80% to 95%. The study also highlighted that the choice of error criteria, such as false positive and false negative rates, is crucial in cybersecurity applications. However, the study acknowledged the limitations of the reviewed methods, particularly in terms of their ability to adapt to evolving threats and the need for continuous model updates.

A methodology for enhancing cyber detectors based on the random forest algorithm against adversarial attacks was proposed in this study by (Apruzzese *et al.*, 2019). The experimental evaluation is performed on a large public dataset of over 20 million network flows containing benign and malicious samples from different botnet families. The study also generates adversarial datasets by manipulating specific flow features to assess the robustness of the proposed detector. The results demonstrate that the distilled random forest detector outperforms the baseline undistilled model, achieving an average F1-score of 0.9777 and a detection rate up to 250% higher in adversarial settings. The study concludes that while the proposed approach represents a significant step towards more robust cyber defensive platforms, there is still room for further improvements to enhance the detection rates. The study's effectiveness in detecting adversarial attacks could potentially be limited by the absence of explicit feature selection techniques, which are crucial for optimizing model performance and robustness across diverse attack scenarios and datasets.

A two-step approach of feature selection based on Random Forest, where the first step selects the features with higher variable importance score and the second step outputs the final feature subset for classification which was proposed in this study by (Hasan *et al.*, 2016). The study used the KDD'99 intrusion detection dataset and derived a new dataset called RRE-KDD by eliminating redundant records from the original dataset. The study also used Random Forest

for model selection to optimize the number of random features, number of trees, and minimum node size. The experimental results show that the approach proposed in this study can select the most significant and relevant features, which reduces the time and number of input features, and enhances the classification accuracy. The test accuracy for Random Forest with 25 features is 91.90%, which is better than the accuracy of 91.41% with 41 features. The study's results may be influenced by the absence of comprehensive data preprocessing steps, such as handling missing values, normalization, or addressing class imbalance, which are essential for ensuring robust feature selection and optimal model performance across different datasets and scenarios. Feature selection techniques like Chi-Square have also been successfully used for selecting the best features in machine learning based intrusion detection (Thaseen, Kumar and Ahmad, (2018); Rustam and Ariantari (2018); Kocher and Kumar (2021)).

Applying machine learning techniques, such as Bayesian networks, decision trees, clustering, and neural networks, to various cyber security problems was proposed in this study by (Das and Morris, 2021). The data used in the study included network packet data, NetFlow data, and a MODBUS dataset from a simulated gas pipeline, which contained 35 distinct attacks. Additionally, the study discussed other methods like inductive learning, Hidden Markov Models, Genetic Algorithms, and Genetic Programming, and their applications in cyber security. The results showed that the J48 decision tree algorithm performed the best among the evaluated methods, achieving an area under the ROC curve (AUC) of 0.995, precision of 0.992, and recall of 0.992 on the MODBUS dataset. The study offers a comprehensive evaluation of these methods' effectiveness in detecting and mitigating cyber threats. The inclusion of different datasets, including network packet data and a MODBUS dataset from a simulated gas pipeline with 35 distinct attacks, enhances the study's applicability. This study focuses on specific datasets like network packet data and NetFlow data may not directly translate to the unique characteristics and challenges posed by SSRF attacks, which often require specialized feature engineering and model adaptation to effectively detect and mitigate in web application environments.

Various popular machine learning classification techniques such as Bayesian Network, Naive Bayes, Random Forest, Decision Tree, Random Tree, Decision Table, and Artificial Neural Network, to detect cyber intrusions was employed in this study by (Alqahtani *et al.*, 2020). The dataset used in the study was the KDD'99 cup dataset, which contains 4,898,431 instances with 41 attributes and different categories of cyber-attacks. The study also discussed the data preprocessing steps, including attack class labeling and feature extraction. The results showed that the Random Forest classifier-based IDS model consistently performed better than other classifiers, with the highest accuracy, precision, recall, and F1-score. The study found that the Random Forest model achieved the best performance, indicating its effectiveness in detecting various cyber-attacks. However, while Random Forest performed exceptionally well in detecting various cyber-attacks on the KDD'99 cup dataset, its application effectiveness may vary when detecting specific types of attacks like SSRF, which require tailored feature selection and model tuning to capture the unique characteristics.

A deep random forest model (FS-DPRF) that combines feature segmentation and a deep structure of parallelized random forest for network intrusion detection was proposed in this study by (Liu *et al.*, 2020). The data used in the study includes four intrusion detection datasets: NSL-KDD, UNSW-NB15, CICIDS2017, and CICIDS2018. The study also introduces a cache replacement optimization for parallelization on the Spark environment. The results show that the proposed FS-DPRF model outperforms the parallel random forest (PRF) and DSSVM algorithms, achieving F1-measure scores of 0.9897 on NSL-KDD, 0.9737 on UNSW-NB15, 0.9661 on CICIDS2017, and 0.9572 on CICIDS2018, which are higher than the deep neural network (A-DNN) method. The limitation of the study is that the model consumes a lot of memory, and the current structure is not suitable for GPU acceleration.

A systematic literature review (SLR) methodology was used to summarize the use of AI and ML methods, specifically the use of classifiers, in the detection of cyber security attacks which was employed in this study by (Ali *et al.*, 2020). The data consisted of 63 research articles retrieved from the online libraries of ScienceDirect and Google Scholar, which were further filtered and refined to 21 articles. The study also applied inclusion and exclusion criteria to select the relevant studies. The meta-level analysis of the selected studies revealed that Support Vector Machine (SVM), Random Forest (RF), Decision Tree (DT), and Artificial Neural Network (ANN) are the most frequently used classifiers in the area of cyber security attack detection. The study concluded that these classifiers have been extensively used in various application domains, such as smart grid ecosystems, mobile environments, and internet of things (IoT) devices, with promising results. However, the study was limited to the analysis of published research articles and did not include other forms of literature, such as gray literature or non-English papers.

2.2 SSRF Attack detection using machine learning

Machine learning algorithms such as Random Forest, KNN, SVM, Naive Bayes Classifier, Logistic Regression, and Decision Tree was used to analyze the probability of incidents in determining forgery vulnerabilities in server-side request attack exposure which was proposed in this study by (Aanandha *et al.*, 2024). The dataset was obtained from the CISA standard of known exploited vulnerabilities with different attributes. The study also performed a vulnerability analysis to identify potential system weaknesses. The results show that the Random Forest, SVM, Naive Bayes Classifier, Logistic Regression, and Decision Tree models achieved an accuracy of 100%, while the KNN model achieved an accuracy of 96%. The study suggests that the Random Forest, SVM, Naive Bayes Classifier, Logistic Regression, and Decision Tree models are the best performers in predicting server-side request forgery vulnerabilities. This study demonstrates that machine learning models can effectively predict server-side request forgery vulnerabilities with high accuracy. One limitation of the study is that it does not address the potential overfitting of the machine learning models, especially given the perfect accuracy rates of 100% for several algorithms. Overfitting can occur when a model performs exceptionally well based on the data used for training but fails to generalize to new, unseen data.

SSRFuzz a novel methodology to effectively identify SSRF vulnerabilities in PHP web applications, which consists of three phases: an SSRF oracle-based picker to identify sensitive sinks, dynamic taint inference to pinpoint feasible input points, and mutation-based fuzzing to generate testing payloads was presented in this study by (Wang *et al.*, 2021). The study evaluated SSRFuzz on 27 real-world web applications, comprising 15.6 million SLOC and 110.9K files. Additionally, the study compared SSRFuzz against existing tools, including BurpSuite and SSRFmap. The results show that SSRFuzz identified 28 vulnerabilities, including 25 previously unknown, and obtained 16 new CVE IDs. This study highlights effective methodologies for identifying SSRF vulnerabilities, providing valuable insights and techniques that can inform and enhance the development of machine learning models for SSRF attack detection. One limitation of this study is the lack of feature selection, which could have potentially enhanced the precision and efficiency of the SSRFuzz methodology in identifying SSRF vulnerabilities.

A novel defense approach to protect internal services from Server-Side Request Forgery (SSRF) attacks, which involves modifying incoming requests to redirect them to a helper server that fetches resources from the Internet was discussed in this study by (Jabiyev *et al.*, 2021). The data for the study consisted of more than 60 HackerOne SSRF vulnerability reports from 2014 to 2019, which were manually analyzed to understand the attack landscape. The study also developed a prototype by improving the functionality of a popular reverse proxy

application and deployed a set of web applications that are vulnerable to evaluate the effectiveness of the proposed approach. The results show that the solution can prevent all in-band SSRF attacks, with only one application requiring minimal developer collaboration, and the performance impact on web applications is negligible. The study's limitation is that it cannot prevent out-of-band SSRF attacks, as the payload is not transferred between the attacker and the victim application.

Long Short-Term Memory (LSTM), a deep learning technique, to build a model for detecting Server-Side Request Forgery (SSRF) attacks was implemented in this study by (Al-talak *et al.*, 2021). The dataset used in the study was obtained from the Canadian Institute for Cybersecurity of the University of New Brunswick, which covered various types of SSRF attacks. The data was preprocessed and transformed into a numerical form to facilitate the training process of the LSTM deep learning algorithm. Additionally, feature scaling was applied to normalize the range of independent variables. The LSTM model achieved an accuracy of 0.969, a precision of 0.975, a recall of 0.965, and an F1-score of 0.970, indicating the strength of the model in detecting SSRF attacks. This study demonstrates the effectiveness of using deep learning techniques, specifically LSTM, for detecting SSRF attacks, providing a strong benchmark and methodology that can guide the implementation and evaluation of your CNN-LSTM model for similar purposes. The study notes that the limitation is the need to explore other machine learning and deep learning models to compare their performance with the LSTM model.

2.3 Summary

A summary table is provided in Table (1).

The table (1) shows research on Server-Side Request Forgery (SSRF) vulnerabilities reveals several critical areas for advancement. The demonstrated effectiveness of various machine learning models in predicting SSRF vulnerabilities have a notable lack of attention to the risk of overfitting, which can impact the generalizability of proposed models. They do not incorporate feature selection, which could improve precision and efficiency. The need for more comprehensive methods that integrate feature selection, data balancing and data scaling have advanced defense strategies, and diverse model evaluations to enhance the detection and prevention of SSRF vulnerabilities.

Table (1): Literature survey based on Server-Side Request Forgery (SSRF)

Author	Proposed Methods.	Dataset Used	Result	Limitation
Aanandha et al., (2024)	Random Forest, KNN, SVM, Naive Bayes Classifier, Logistic Regression, Decision Tree	CISA standard of known exploited vulnerabilities	Random Forest, SVM, Naive Bayes Classifier, Logistic Regression, and Decision Tree: 100% accuracy; KNN: 96% accuracy	Potential overfitting of models due to perfect accuracy rates for several algorithms
Wang et al., (2021)	SSRFuzz (SSRF oracle-based picker, dynamic	27 real-world web applications,	Identified 28 vulnerabilities, 25 previously unknown,	Lack of feature selection could have enhanced the

Author	Proposed Methods.	Dataset Used	Result	Limitation
	taint inference, mutation-based fuzzing)	15.6 million SLOC, 110.9K files	obtained 16 new CVE IDs	precision and efficiency of SSRFuzz
Jabiyev et al., (2021)	Modification of incoming requests to redirect to a helper server	More than 60 HackerOne SSRF vulnerability reports (2014-2019)	Prevented all in-band SSRF attacks; minimal developer collaboration needed; negligible performance impact	Cannot prevent out-of-band SSRF attacks as payload is not transferred between attacker and victim application
Al-talak et al., (2021)	Long Short-Term Memory (LSTM) for detecting SSRF attacks	Canadian Institute for Cybersecurity dataset	Accuracy: 0.969, Precision: 0.975, Recall: 0.965, F1-score: 0.970	Needs exploration of other machine learning and deep learning models for performance comparison

3 Research Methodology

This research study introduces detecting Server-Side Request Forgery (SSRF) attacks using advanced machine learning techniques. SSRF is a critical security vulnerability that allows attackers to make unauthorized requests from the server, potentially compromising sensitive information. The study aims to develop a machine learning model to accurately identify and classify SSRF attacks, using a combination of RandomForestClassifier and CNN-LSTM architectures.

Figure (2) is the block diagram that outlines the workflow of this study. The process begins with data acquisition from publicly available datasets, including benign, malicious datas. Each dataset is labeled appropriately to differentiate between benign and malicious instances. The study consolidates these datasets, ensuring consistent feature representation by aligning and standardizing columns across all data sources. Missing values are addressed, and categorical features are encoded using LabelEncoder to facilitate model training. To address class imbalance, the study employs Synthetic Minority Over-sampling Technique (SMOTE) to generate a balanced dataset. Feature selection is performed using Chi2 to identify the most relevant features, which are then scaled using StandardScaler to standardize the data for model training. This preprocessing step ensures that the input data is well-suited for machine learning algorithms.

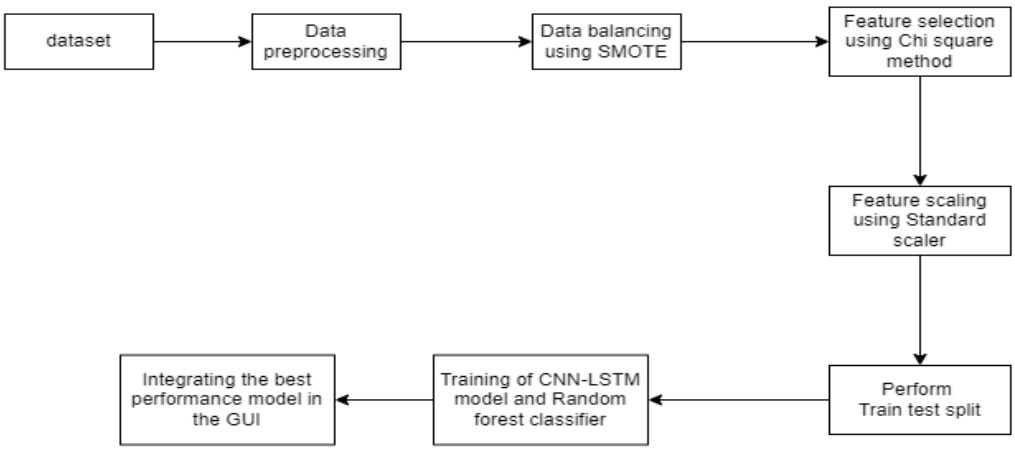

Figure (2): The workflow of detecting SSRF by Using Machine Learning.

Two distinct models are trained and evaluated: a RandomForestClassifier and a CNN-LSTM network. The RandomForestClassifier is initialized and trained on the preprocessed data, with hyperparameters set for optimal performance. The model is evaluated based on classification metrics such as accuracy, F1-score providing insights into its effectiveness in detecting SSRF attacks. A CNN-LSTM model is constructed to capture spatial and temporal patterns in the data. The model architecture includes Conv1D layers for feature extraction, MaxPooling1D for down-sampling, and LSTM layers for capturing sequential dependencies. Dropout layers are used to prevent overfitting, and the model is trained with early stopping and checkpointing mechanisms to enhance performance.

The CNN-LSTM model is evaluated similarly, with a focus on accuracy and classification report metrics. The study also uses Python Tkinter GUI to enable user interaction with the trained models. The GUI allows users to input data and receive real-time predictions on whether the data indicates normal behavior or an attack. The trained models and necessary preprocessing components, such as scalars and label encoders, are saved for future use. This research provides a practical solution for detecting SSRF attacks using machine learning, offering insights into model performance and potential improvements. The combination of RandomForestClassifier and CNN-LSTM demonstrates the effectiveness of integrating traditional and deep learning techniques in addressing complex security challenges.

3.1 Dataset

In this study, the CIC-Bell-DNS 2021 dataset[7] is employed to develop and evaluate machine learning models for detecting and preventing Server-Side Request Forgery (SSRF) attacks (*CIC-Bell-DNS 2021 | Datasets | Research | Canadian Institute for Cybersecurity | UNB*, 2021). This dataset was developed by the Canadian Institute for Cybersecurity in collaboration with Bell Canada, offers a comprehensive collection of DNS (Domain Name System) traffic data designed to support network security research and anomaly detection. This dataset includes both legitimate and malicious DNS traffic, providing a broad spectrum of scenarios from typical network operations to simulated attack patterns. The dataset is used to train and test models such as RandomForestClassifier and CNN-LSTM by providing a rich mix of DNS traffic data, which includes both benign and malicious examples. This diverse dataset aids in feature extraction and selection, helping to identify critical indicators of SSRF attacks. Additionally, it addresses class imbalance issues through techniques like SMOTE, ensuring a balanced representation of different traffic types. By evaluating the models using this realistic

[7] https://www.unb.ca/cic/datasets/dns-2021.html

dataset, the study ensures that the detection mechanisms are effective and applicable to real-world scenarios. Ultimately, the CIC-Bell-DNS 2021 dataset is crucial for building robust and practical solutions for SSRF attack detection.

3.2 Data preprocessing

Data preprocessing involves the systematic cleaning, transforming, and organizing of raw data to make it suitable for analysis and model training (Hussein *et al.*, 2019). This step typically includes handling values that are missing, normalizing or scaling data, encoding categorical variables, and addressing class imbalances to ensure the data is accurate and in a format that enhances the performance of machine learning models. In this study, data preprocessing is used to prepare the CIC-Bell-DNS 2021 dataset for training machine learning models aimed at detecting Server-Side Request Forgery (SSRF) attacks. The preprocessing steps involve several critical tasks: first, handling missing values by filling them with zeros to avoid disruptions during model training. Next, categorical variables are encoded using LabelEncoder to convert them into numerical formats suitable for machine learning algorithms. The preprocessing steps are essential for enhancing the quality of the dataset, ensuring that the models are trained on clean, well-structured data, and improving the overall effectiveness of SSRF detection and prevention mechanisms.

3.3 Data Balancing

Data balancing refers to the process of adjusting the distribution of classes within a dataset to ensure that each class is equally represented (Almomani *et al.*, 2021). This is crucial in machine learning to prevent models from being biased towards the majority class, which can lead to poor performance in the minority class. In this study, data balancing is employed to address the class imbalance present in the CIC-Bell-DNS 2021 dataset, which can negatively impact the performance of machine learning models for detecting Server-Side Request Forgery (SSRF) attacks. The imbalance is corrected using Synthetic Minority Over-sampling Technique (SMOTE), a method that generates synthetic samples for the underrepresented class, thereby equalizing the number of instances in each class. Implemented through the `SMOTE` class from the imbalanced-learn library, this technique effectively enhances the dataset by creating new, plausible examples of the minority class. This balancing process is crucial for training models like RandomForestClassifier and CNN-LSTM, as it ensures that the models are exposed to a balanced distribution of both benign and attack data. Consequently, the models are less likely to be biased towards predicting the majority class and can more effectively learn to identify and differentiate between SSRF attacks and normal traffic. By employing SMOTE, the study improves the reliability and accuracy of the models in detecting SSRF attacks, leading to more robust and fair evaluation of their performance.

3.4 Feature selection

Feature selection is a process used to identify and retain the most relevant attributes or variables from a dataset, which significantly impacts the performance of machine learning models (Li *et al.*, 2017). By eliminating unwanted or redundant features, feature selection improves model efficiency, reduces overfitting, and enhances overall accuracy. In this study, feature selection is utilized to refine the CIC-Bell-DNS 2021 dataset, focusing on identifying the most pertinent features for detecting Server-Side Request Forgery (SSRF) attacks. The process involves using the Chi2 statistical test to evaluate the relationship between each feature and the target variable, selecting those that exhibit the strongest associations. This ensures that only the most informative features are retained for model training. Implemented through the `SelectKBest`

method, feature selection reduces the dimensionality of the data, which helps to streamline the machine learning models by eliminating noise and irrelevant information. By concentrating on the top-ranked features, the study enhances the models' ability to identify SSRF attacks with greater precision. The reduced feature set also accelerates the training process and mitigates the risk of overfitting. This strategic selection of features is crucial for building effective models that can accurately detect SSRF attacks while maintaining computational efficiency.

3.5 Feature scaling

Feature scaling is a technique used to normalize the range of independent variables or features in a dataset to ensure that they contribute equally to the model's learning process (Wan, 2019). This process adjusts the scale of features so that they have similar ranges, which prevents features with larger ranges from disproportionately influencing the model's performance. In this study, feature scaling is used to standardize the CIC-Bell-DNS 2021 dataset, preparing it for effective machine learning model training. The scaling process involves transforming the features so that they have a mean of zero and a standard deviation of one, which is achieved using the StandardScaler from the scikit-learn library. This is implemented by fitting the scalar to the data and then applying it to both the training and testing datasets to ensure consistency. Standardization is particularly important in this study as it allows algorithms like RandomForestClassifier and CNN-LSTM to process features on a comparable scale, thus improving their convergence during training and ensuring fair weight distribution among features. By scaling the data, the study mitigates potential issues related to feature magnitude differences, such as skewed model performance and slower training times. This preprocessing step enhances the overall effectiveness of the models by ensuring that each feature is treated equitably, leading to more accurate detection of Server-Side Request Forgery (SSRF) attacks and better model performance across different types of network traffic.

3.6 Training

Machine learning model training involves using algorithms to learn patterns from a dataset by adjusting the model's parameters to minimize prediction errors. This iterative process involves feeding the algorithm a labeled dataset, allowing it to adjust and optimize its internal parameters based on the differences between predicted and actual outcomes. In this study, machine learning model training is employed to develop and refine models for detecting Server-Side Request Forgery (SSRF) attacks using the CIC-Bell-DNS 2021 dataset. The training process involves two key models: RandomForestClassifier and CNN-LSTM. For the RandomForestClassifier, the model is trained using a subset of the preprocessed dataset, where it learns to distinguish between benign and malicious DNS traffic based on various features. This training is conducted with hyperparameters tuned to optimize performance, and the model’s effectiveness is evaluated using metrics like accuracy and F1-score. The CNN-LSTM model, on the other hand, leverages convolutional and recurrent layers to capture temporal patterns in DNS traffic data. The model is trained using sequences of data with the aim of learning complex patterns that indicate SSRF attacks. Training involves specifying the model architecture, compiling it with appropriate loss functions and optimizers, and fitting it to the data using techniques like early stopping and model checkpointing to avoid overfitting. By employing these training techniques, the study develops robust models capable of accurately detecting SSRF attacks, thereby improving network security and threat detection capabilities.

4 Design Specification

4.1 Chi2

Chi2 (Chi-square) is a statistical test used in machine learning for feature selection (Rupapara *et al.*, 2023). It evaluates the relationship between categorical variables by comparing the observed frequencies with expected frequencies. Chi2 determines the importance of each feature's association with the target variable, aiding in the identification of the most pertinent features for the model.

4.2 SMOTE

Synthetic Minority Over-sampling Technique (SMOTE) is an approach designed to tackle class imbalance in machine learning datasets (Dablain, Krawczyk and Chawla, 2023). It generates synthetic samples for the minority class by creating new instances that are similar, but not identical, to the original minority class examples. This method enhances the class distribution balance by augmenting the dataset with these new synthetic samples. By doing so, SMOTE improves the performance of models trained on imbalanced datasets by offering a more representative set of examples for the underrepresented class.

4.3 CNN-LSTM

The CNN-LSTM model combines Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks to use both spatial and temporal patterns in sequential data (Lu *et al.*, 2020). CNNs are used for feature extraction from spatial data, such as images or sequences, by applying convolutional layers to capture hierarchical features and patterns. This is particularly effective for identifying local structures and reducing dimensionality. LSTMs, a type of Recurrent Neural Network (RNN), are then employed to analyze these features over time or sequence. LSTMs are designed to capture long-term dependencies and temporal relationships by using memory cells that can retain information across many time steps, making them well-suited for sequential or time-series data. In a CNN-LSTM model, the CNN layers first process the input data to extract relevant features. These features are then fed into LSTM layers, which learn the temporal dependencies and sequential patterns in the data. This combination enables the model to effectively capture both spatial and temporal information, making it powerful for tasks involving sequences with complex patterns.

4.4 Randomforest

Random Forest is an ensemble learning method used for classification and regression tasks (Paul et al., 2018). It operates by constructing a multitude of decision trees during training and outputs the mode of the classes (classification) or mean prediction (regression) of the individual trees. Each decision tree in the Random Forest is trained on a random subset of the data, and each split within the tree is made based on a random subset of features. This approach helps in reducing overfitting and increases the model's generalization capability. The main advantages of Random Forest include its robustness to overfitting, ability to handle large datasets with numerous features, and effectiveness in estimating feature importance. It is particularly useful in scenarios where there are complex interactions and non-linear relationships between features. By aggregating the results of multiple decision trees, Random Forest provides a more stable and accurate prediction compared to a single decision tree, making it a popular choice for a wide range of machine learning tasks.

5 Implementation

This study develops machine learning models to detect Server-Side Request Forgery (SSRF) attacks using the CIC-Bell-DNS 2021 dataset. It involved preprocessing the dataset to handle missing values and normalize feature scales. Data balancing was performed using SMOTE to

address class imbalances, ensuring fair model training. Feature selection with Chi2 helped identify the most relevant features, while feature scaling standardized data for consistent model performance. Two machine learning models were trained: RandomForestClassifier, which learns from multiple decision trees to improve accuracy, and a CNN-LSTM model that combines convolutional layers for feature extraction with LSTM layers to capture temporal dependencies. The study aimed to enhance the detection of SSRF attacks by employing these techniques to build robust models that can accurately differentiate between benign and malicious network traffic.

In this study, the integration of machine learning predictions into a user-friendly graphical user interface (GUI) was central to detecting SSRF (Server-Side Request Forgery) attacks. The GUI, developed using Tkinter, facilitates interaction with the CNN-LSTM model by providing a straightforward platform for users to input data and receive predictions. The core functionality was encapsulated in the `predict()` function, which played a crucial role in the prediction process. When the `predict()` function was triggered, it first loaded the pre-trained CNN-LSTM model along with necessary preprocessing tools, including a scaler and a label encoder, which was essential for ensuring the model's predictions were accurate and relevant. The function then retrieves user input from the GUI, processes this input by converting it into a numerical format compatible with the model, and reshapes it to fit the model's input requirements. This step was crucial as the model expects data in a specific shape for accurate prediction. Once the data was properly formatted, the function used the CNN-LSTM model to make predictions regarding the potential presence of SSRF attacks. The results, which indicate the likelihood of an attack, were displayed to the user through the GUI. If the model detects an SSRF attack, the system further enhances security by sending an email alert. This email functionality was integrated into the GUI through the `send_email()` function, which configured email settings, composes the message, and sends notifications about detected threats. The GUI itself was divided into two main tabs: "Home" and "Check." The "Home" tab serves as an introductory page with a brief description of the tool and a button for navigating to the "Check" tab. In the "Check" tab, users can input data into a designated entry field, view a log of activity in a text box, and initiate the prediction process using a button. The text box logs each stage of the prediction process, providing users with detailed feedback from model loading to prediction results and email notifications.

6 Result and Evaluation

6.1 Result

The classification report for a RandomForestClassifier model shows its performance metrics. Precision measures the accuracy of positive predictions, with class '0' achieving a precision of 0.97 and class '1' attaining a perfect precision of 1.00. Recall evaluates the model's ability to identify all relevant cases, with class '0' having a recall of 1.00 and class '1' having a recall of 0.97. The F1-Score, which is the harmonic mean of precision and recall, is 0.98 for both classes, indicating a balanced performance. Support reflects the number of actual occurrences in the dataset, with 78,838 instances for class '0' and 79,267 instances for class '1'. The model's overall accuracy stands at 0.9841, meaning it correctly predicted 98.41% of the instances. Additionally, the ROC AUC Score of 0.9966 signifies excellent capability in distinguishing between classes.

The classification report for the CNN-LSTM model highlights its performance metrics. Precision quantifies the accuracy of positive predictions, with both classes '0' and '1' achieving a precision of 1.00, indicating perfect accuracy in positive predictions. Recall assesses the model's ability to identify all relevant cases, with a recall of 1.00 for both classes, signifying

that the model successfully detected all relevant instances. The F1-Score, representing the harmonic mean of precision and recall, is 1.00 for both classes, reflecting an ideal balance between these metrics. Support reveals the number of actual instances per class, with class '0' having 78,838 instances and class '1' having 79,267 instances. The Accuracy of the model stands at 0.9965, indicating that the model accurately predicted the class for 99.65% of instances. Additionally, the Macro Avg and Weighted Avg rows, which average the performance metrics across all classes, both show values of 1.00 for precision, recall, and F1-score, demonstrating consistent performance across the different classes. The model has been saved as 'random_forest_model.pkl'. For any further questions or additional clarification on these metrics, please feel free to inquire.

```
Selected features: Index(['State', 'typos', '3gram', 'dec_32', 'tld', 'char_distribution',
       'Registrar', 'Domain', 'TTL', 'oc_8', 'subdomain', '2gram', 'IP', 'ASN',
       'Emails', 'puny_coded', 'Country.1', 'numeric_percentage', 'dec_8',
       'Registrant_Name'],
      dtype='object')
```

Figure (3): Selected features

Figure (3) outlines a list of features associated with internet domains, which are used for machine learning purposes. The State feature indicates the current status of the domain, such as whether it is active or inactive. typos represent the count of typographical errors present in the domain name, which can be useful for detecting potential phishing or fraud. 3gram refers to trigrams, which are sequences of three consecutive characters from the domain name, used for text analysis and pattern recognition. 2_3x might represent a combined measure of bigrams and trigrams, offering a more detailed text analysis. char_distribution shows the frequency distribution of different characters in the domain name, which can help in identifying unusual patterns. The Registrar feature identifies the entity responsible for registering the domain. Domain denotes the actual name of the domain. TTL (Time to Live) is a DNS setting that specifies how long a domain's record is cached before being refreshed. The oc_8 feature could represent a specific dataset-related metric, possibly indicating occurrence counts or other relevant statistics. subdomain details the part of the URL that precedes the main domain, potentially indicating sub-categories or sections of the domain. 2gram denotes bigrams, or pairs of two consecutive characters, used similarly to trigrams for text analysis. The IP feature lists the IP address associated with the domain, which can be useful for network analysis and tracking. ASN1 (Autonomous System Number 1) is used to identify the network to which the domain belongs. Email(s) lists any email addresses associated with the domain, which can be useful for identifying domain owners or potential spam. Puny_code is the encoded representation of internationalized domain names, allowing non-ASCII characters to be used in domain names. Country specifies the country where the domain is registered, which can be useful for geographical analysis. Lastly, numeric_percentage indicates the percentage of numeric characters within the domain name, which can help in assessing the domain's complexity or potential for obfuscation. Registrant Nameidentifies the individual or organization who registered the domain.

GUI was built as a desktop application. The interface in which detection of SSRF attacks takes place contains a title and two main tabs: "Home" and "Check"(Figure (4)). At the top of the interface is the title, "SSRF Attack Detector," which clearly identifies the purpose of the tool. The "Home" tab features a prominent label with the same title and includes a "Check" button that allows users to navigate to the "Check" tab. In the "Check" tab, users are presented with two main sections. The "Features" section includes a label indicating its purpose and an entry box where users can input features, such as numerical values ("307589315, -19995573906193613"). This section also contains a "Predict" button that initiates the

prediction process. The "Activity Log" section, on the other hand, provides real-time feedback on the system's operations. It features a label and a text box that displays various status messages related to the preprocessing steps and prediction process. For instance, messages such as "Loading the model...", "Model loaded successfully.", "Loading the scaler...", and "Scaler loaded successfully." are shown, culminating in a final message like "Predicted attack: ['benign']." Additionally, the interface includes a message box that displays the prediction result, which in this case is "Predicted attack: ['benign']." The message box also contains an "OK" button to close the alert. This setup allows users to input their data, observe the real-time processing and results, and understand the predicted outcomes clearly. The design ensures that users can interact effectively with the machine learning model and obtain timely feedback on potential SSRF attacks.

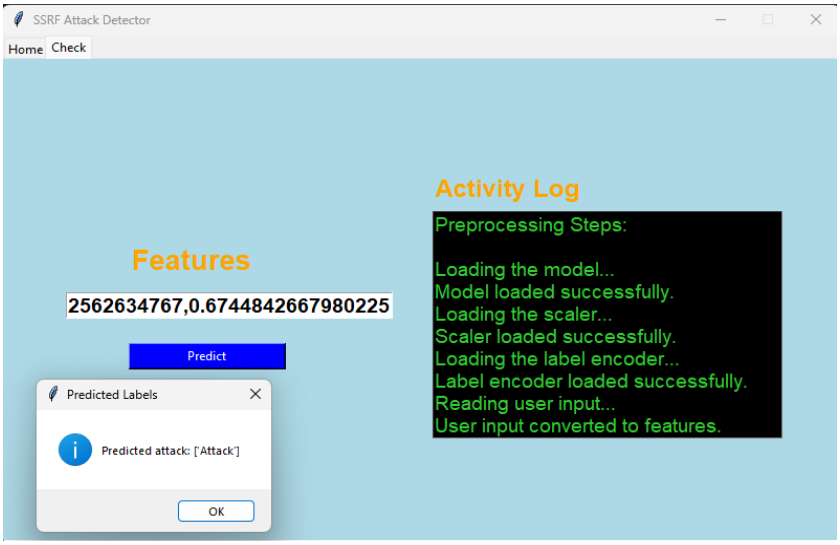

Figure (4): Predicted as attack

The "SSRF Attack Detector" application interface is designed for ease of use and functionality. At the top of the interface, the title "SSRF Attack Detector" clearly identifies the application's purpose. The interface is organized into two main tabs: "Home" and "Check." The "Home" tab prominently features a label with the application's title and includes a "Check" button that allows users to navigate to the "Check" tab. In the "Check" tab, users interact with two main sections. The "Features" section comprises a label and an entry box where users can input feature values, such as "2562634767, 0.6744842667980225." There is also a "Predict" button that initiates the prediction process. The "Activity Log" section, meanwhile, provides real-time feedback through a text box. This section logs various messages related to the processing steps, including "Loading the model...", "Model loaded successfully.", "Loading the scaler...", and "Scaler loaded successfully.", leading up to the final prediction result, such as "Predicted attack: ['Attack']." A message box further enhances the user experience by displaying the final prediction result, such as "Predicted attack: ['Attack']" as shown in the figure (4) and includes an "OK" button to close the message box. The functionality of the interface allows users to input feature values, trigger the prediction model, and view the results and preprocessing steps dynamically. If the model detects an attack, the system can also send an email alert, ensuring timely notifications of potential threats. This design effectively facilitates user interaction with the machine learning model and provides clear, real-time feedback on predictions.

The research questions of the study were answered, and these are:

What machine learning algorithms can be used to perfect the detection of Server-Side Request Forgery (SSRF)?

To enhance the detection of Server-Side Request Forgery (SSRF), various machine learning algorithms can be employed. The study demonstrates the effectiveness of integrating Convolutional Neural Networks (CNNs) with Long Short-Term Memory (LSTM) networks, achieving a high accuracy of 99.65% in SSRF detection. This approach uses CNNs for feature extraction and LSTMs for sequential pattern recognition, making it adept at identifying complex attack patterns. Additional methods like RandomForest, which achieved an accuracy of 98.41% in the study, can be beneficial due to their robustness and ability to handle diverse data features.

Is it possible to improve the model's ability to detect SSRF anomalies using machine learning by utilizing SMOTE and Chi2 feature selection?

Improving the model's ability to detect SSRF anomalies using machine learning can be achieved through techniques like SMOTE and Chi2 feature selection. In the study, SMOTE (Synthetic Minority Over-sampling Technique) was employed to address class imbalances by generating synthetic samples, which enhanced the model's ability to detect rare SSRF attacks effectively. This balancing approach ensures that the model is trained on a more representative dataset, reducing bias toward the majority class. Additionally, Chi2 feature selection was utilized to identify the most relevant features, thereby improving the model's performance by focusing on the most significant variables and reducing dimensionality. By eliminating irrelevant or less important features, Chi2 helps in building a more efficient and accurate model. The combined use of SMOTE and Chi2 not only aids in managing imbalanced data but also enhances feature relevance, leading to a more robust detection system for SSRF anomalies.

6.2 Evaluation

The study provides a comprehensive analysis of domain-related features, incorporating machine learning models to enhance domain classification and detection. The dataset includes critical features such as typographical errors, character distributions, and DNS attributes, which are instrumental in evaluating domain legitimacy and potential threats. The use of advanced models like CNN-LSTM and RandomForestClassifier demonstrates a robust approach to handling complex data, yielding high precision, recall, and accuracy. The integration of these models into a user-friendly GUI with real-time prediction and email alert functionalities adds practical value, facilitating effective monitoring and response to potential attacks.

7 Conclusion and Future Enhancement

This study introduces a machine learning-based approach to detecting Server-Side Request Forgery (SSRF) attacks using a convolutional neural network (CNN) and long short-term memory (LSTM) model. This approach demonstrates superior performance compared to Random Forest model, offering enhanced accuracy and effectiveness in identifying SSRF anomalies. The implementation integrates a user-friendly graphical user interface (GUI) that facilitates real-time data input and prediction. The study demonstrates the effectiveness of the CNN-LSTM model in identifying potential SSRF attacks with high precision and recall, achieving robust performance metrics. The GUI enhances user interaction by providing clear feedback and detailed logs of the prediction process, while also enabling email notifications for detected attacks. However, the study has notable limitations, including the absence of live monitoring capabilities and the restriction to binary classification, which excludes the differentiation of various attack types. Future work should address these limitations by

incorporating real-time monitoring and expanding the model to classify different attack categories, thus offering a more comprehensive security solution.

Future enhancements to this study could focus on several key areas to improve the SSRF attack detection system. Integrating live monitoring capabilities into the SSRF attack detection system represents a significant enhancement that would substantially improve its effectiveness and responsiveness. By implementing real-time monitoring, the system could continuously analyze network traffic and other relevant data streams to detect and respond to SSRF attacks as they occur. This dynamic approach would enable the system to identify potential threats immediately, rather than relying on periodic checks or batch processing.

Incorporating advanced techniques such as ensemble methods or hybrid models presents a promising avenue for future work to enhance the detection accuracy of SSRF attacks and minimize false positives. Ensemble methods, which combine multiple models to make a final prediction, can utilize the strengths of different algorithms and improve overall performance by aggregating their outputs. Hybrid models, which integrate various types of machine learning approaches, such as combining CNNs with GRUs or transformers, can capture both spatial and sequential features more effectively.

References

Aanandha, K. *et al.* (2024) 'Probability of Incidents in Determining Forgery Vulnerabilities in Server-Side Request Attack Exposure with Predictive ML Analysis', *Afr.J.Bio.Sc.*, pp. 5955–5967. https://www.afjbs.com/uploads/paper/33ed1bf13e3e90d5a3dbeef26f3b79ea.pdf. [Accessed 19 Jul. 2024]

Agrawal, S. (2023) *Mitigating Cross-Site request forgery (CSRF) attacks using reinforcement learning and predictive analytics*. https://www.researchberg.com/index.php/araic/article/view/189. [Accessed 24 Jun. 2024]

Ali, R. *et al.* (2020) 'A Systematic review of artificial intelligence and machine learning techniques for cyber security,' *Communications in computer and information science*, pp. 584–593. https://doi.org/10.1007/978-981-15-7530-3_44. [Accessed 19 Jul. 2024]

Almomani, I. *et al.* (2021) 'Android ransomware detection based on a hybrid evolutionary approach in the context of highly imbalanced data,' *IEEE Access*, 9, pp. 57674–57691. https://doi.org/10.1109/access.2021.3071450. [Accessed 19 Jul. 2024]

Alqahtani, H. *et al.* (2020) 'Cyber intrusion detection using machine learning classification techniques', in *Communications in computer and information science*, pp. 121–131. https://doi.org/10.1007/978-981-15-6648-6_10. [Accessed 19 Jul. 2024]

Al-talak, K. *et al.* (2021) 'Detecting Server-Side Request Forgery (SSRF) Attack by using Deep Learning Techniques*', International Journal of Advanced Computer Science and Applications*. https://thesai.org/Downloads/Volume12No12/Paper_30-Detecting_Server_Side_Request_Forgery.pdf [Accessed 22 June. 2024]

Apruzzese, G. *et al.* (2019) 'Hardening Random Forest Cyber Detectors Against Adversarial Attacks*'. journal-article.* https://arxiv.org/pdf/1912.03790. [Accessed 17 June. 2024]

Calzavara, S. *et al.* (2019) 'Mitch: A Machine Learning Approach to the Black-Box Detection of CSRF Vulnerabilities,' *2019 IEEE European Symposium on Security and Privacy (EuroS&P)* [Preprint]. https://doi.org/10.1109/eurosp.2019.00045.

Dablain, D., Krawczyk, B. and Chawla, N.V. (2023) 'DeepSMOTE: Fusing deep learning and SMOTE for imbalanced data,' *IEEE Transactions on Neural Networks and Learning Systems*, 34(9), pp. 6390–6404. https://doi.org/10.1109/tnnls.2021.3136503. [Accessed 5 Jul. 2024]

Das, R. and Morris, T.H. (2021) 'Machine Learning and Cyber Security', *journal-article*. https://www.researchgate.net/profile/Thomas-Morris-2/publication/328815330_Machine_Learning_and_Cyber_Security/links/5c77f0c392851c695047e65a/Machine-Learning-and-Cyber-Security.pdf. [Accessed 5 Jul. 2024]

Dasgupta, D., Akhtar, Z. and Sen, S. (2020) 'Machine learning in cybersecurity: a comprehensive survey', *Journal of Defense Modeling and Simulation*, 19(1), pp. 57–106. https://doi.org/10.1177/1548512920951275. [Accessed 11 Jul. 2024]

Hasan, Md.A.M. *et al.* (2016) 'Feature Selection for Intrusion Detection Using Random Forest', *Journal of Information Security*, 07(03), pp. 129–140. https://doi.org/10.4236/jis.2016.73009. [Accessed 28 Jun. 2024]

Hussein, A.S. *et al.* (2019) 'A-SMOTE: a new preprocessing approach for highly imbalanced datasets by improving SMOTE', *the International Journal of Computational Intelligence Systems/International Journal of Computational Intelligence Systems*, 12(2), p. 1412. https://doi.org/10.2991/ijcis.d.191114.002. [Accessed 11 Jul. 2024]

Jabiyev, B. *et al.* (2021) 'Preventing Server-Side Request Forgery Attacks', *The 36th ACM/SIGAPP Symposium on Applied Computing (SAC '21)*, p. 10. https://dl.acm.org/doi/pdf/10.1145/3412841.3442036. [Accessed 24 Jun. 2024]

Kaur, J. and Garg, U. (2021) 'A detailed survey on recent XSS Web-Attacks Machine Learning detection techniques,' *2021 2nd Global Conference for Advancement in Technology (GCAT)* [Preprint]. https://doi.org/10.1109/gcat52182.2021.9587569. [Accessed 26 Jun. 2024]

Kaur, J., Garg, U. and Bathla, G. (2023) 'Detection of cross-site scripting (XSS) attacks using machine learning techniques: a review,' *Artificial Intelligence Review*, 56(11), pp. 12725–12769. https://doi.org/10.1007/s10462-023-10433-3. [Accessed 26 Jun. 2024]

Khodayari, S. and Pellegrino, G. (2021) 'JAW: Studying Client-side CSRF with Hybrid Property Graphs and Declarative Traversals', *Proceedings of the 30th USENIX Security Symposium*. https://www.usenix.org/system/files/sec21-khodayari.pdf. [Accessed 6 Jul. 2024]

Kocher, G. and Kumar, G. (2021) 'Analysis of Machine Learning Algorithms with Feature Selection for Intrusion Detection Using UNSW-NB15 Dataset,' *SSRN Electronic Journal* [Preprint]. https://doi.org/10.2139/ssrn.3784406. [Accessed 01 Jul. 2024]

Kshetri, N. *et al.* (2024) 'AlgoXSSF: Detection and Analysis of Cross-Site Request Forgery (XSRF) and Cross-Site Scripting (XSS) Attacks via Machine Learning Algorithms,' *2024 12th International Symposium on Digital Forensics and Security (ISDFS)* [Preprint]. https://doi.org/10.1109/isdfs60797.2024.10527278. [Accessed 24 Jun. 2024]

Kumar, T.C.R.S., Pokala Jyothi, Mr.P. Prashanth (2022) *Machine learning for web vulnerability detection: The case of cross site request forgery*. https://www.publishoa.com/index.php/journal/article/view/1267. [Accessed 24 Jun. 2024]

Li, J. *et al.* (2017) 'Feature selection', *ACM Computing Surveys*, 50(6), pp. 1–45. https://doi.org/10.1145/3136625. [Accessed 17 Jul. 2024]

Liu, Z. *et al.* (2020) 'A deep random forest model on SpArk for network intrusion detection', *Journal of Mobile Information Systems*, 2020, pp. 1–16. https://doi.org/10.1155/2020/6633252. [Accessed 5 Jul. 2024]

Lu, W. *et al.* (2020) 'A CNN-LSTM-Based model to forecast stock prices', *Complexity*, 2020, pp. 1–10. https://doi.org/10.1155/2020/6622927. [Accessed 28 Jun. 2024]

Muddham, N. and Srimathi, K. (2022) 'USING MACHINE LEARNING TO DETECT CROSS-SITE REQUEST FORGERY,' *ResMilitaris*, 12(6). https://resmilitaris.net/uploads/paper/4379d6de3c0834f66bb4a062e2d1a064.pdf. [Accessed 24 Jun. 2024]

Pasha, S.J. and Mohamed, E.S. (2022) 'Advanced hybrid ensemble gain ratio feature selection model using machine learning for enhanced disease risk prediction', *Informatics in Medicine Unlocked*, 32, p. 101064. https://doi.org/10.1016/j.imu.2022.101064. [Accessed 12 Jul. 2024]

Paul, A. *et al.* (2018) 'Improved random forest for classification', *IEEE Transactions on Image Processing*, 27(8), pp. 4012–4024. https://doi.org/10.1109/tip.2018.2834830. [Accessed 20 Jul. 2024]
Rathore, S., Sharma, P.K. and Park, J.H. (2017) 'XSSClassifier: An efficient XSS attack detection approach based on machine learning classifier on SNSs,' *Journal of Information Processing Systems* [Preprint]. https://doi.org/10.3745/jips.03.0079. [Accessed 26 Jun. 2024]

Rupapara, V. *et al.* (2023) 'Chi-Square and PCA Based Feature Selection for Diabetes Detection with Ensemble Classifier', *Intelligent Automation & Soft Computing*, 36(2), pp. 1931–1949. https://doi.org/10.32604/iasc.2023.028257. [Accessed 10 Jul. 2024]

Rustam, Z. and Ariantari, N.P. a. A. (2018) 'Comparison between support vector machine and fuzzy Kernel C-Means as classifiers for intrusion detection system using chi-square feature selection,' *AIP Conference Proceedings* [Preprint]. https://doi.org/10.1063/1.5064211. [Accessed 01 Jul. 2024]

Srokosz, M., Rusinek, D. and Ksiezopolski, B. (2018) 'A new WAF-based architecture for protecting web applications against CSRF attacks in malicious environment', *Annals of Computer Science and Information Systems* [Preprint]. https://doi.org/10.15439/2018f208. [Accessed 12 Jul. 2024]

Thaseen, I.S., Kumar, Ch.A. and Ahmad, A. (2018) 'Integrated intrusion detection model using Chi-Square feature selection and ensemble of classifiers,' *Arabian Journal for Science and Engineering*, 44(4), pp. 3357–3368. https://doi.org/10.1007/s13369-018-3507-5. [Accessed 01 Jul. 2024]

Torres, J.M., Comesaña, C.I. and García-Nieto, P.J. (2019) 'Review: machine learning techniques applied to cybersecurity', *International Journal of Machine Learning and Cybernetics*, 10(10), pp. 2823–2836. https://doi.org/10.1007/s13042-018-00906-1. [Accessed 20 Jun. 2024]

Wan, X. (2019) 'Influence of feature scaling on convergence of gradient iterative algorithm', *Journal of Physics Conference Series*, 1213(3), p. 032021. https://doi.org/10.1088/1742-6596/1213/3/032021. [Accessed 17 Jun. 2024]

Wang, E., *et al.* (2021) 'Where URLs Become Weapons: Automated Discovery of SSRF Vulnerabilities in Web Applications', *National University of Defense Technology*. https://www.jianjunchen.com/p/ssrfuzz.sp24.pdf. [Accessed 10 Jun. 2024]

Yavanoglu, O. and Aydos, M. (2017) 'A review on cyber security datasets for machine learning algorithms', *IEEE Access.* [Preprint]. https://doi.org/10.1109/bigdata.2017.8258167. [Accessed 19 Jun. 2024]