The Pennsylvania State University
The Graduate School

SECURITY PATTERN DETECTION IN SOFTWARE CODE USING MACHINE LEARNING ALGORITHMS

A Thesis in

Cybersecurity Analytics and Operations

by
Joonyoung Cha

Master of Science

May 2022

The thesis of Joonyoung Cha was reviewed and approved by the following:

Jungwoo Ryoo
Professor of Information Sciences and Technology
Thesis Adviser

David Fusco
Professor of Information Sciences and Technology

Phillip Laplante
Professor of Software Systems and Engineering

Mary Beth Rosson
Professor of Information Sciences and Technology
Head of Department of Information Sciences and Technology

Abstract

Security patterns, defined as reusable building blocks of secure software code architecture, provide solutions to recurring security flaws and problems in specific contexts. Implementing non-standard or incomplete security patterns may create vulnerabilities that cybercriminals can exploit to execute various attacks on a computer system. Security patterns must be accurately identified and used to enhance software code quality and security features. This study examines the possibility of using machine learning algorithms to detect security patterns in software code. The proposed framework for our research is the Security Pattern Detection (SPD) and its internal pattern matching technique, Non-uniform Distributed Matrix Matching (NDMM). The machine learning algorithms selected for our study are Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM). The primary data for the study were collected by interviewing experts who agreed to participate in the study. The purposive sampling method was used to select experts in machine learning algorithms and security pattern detection in software code. The experts' responses were analyzed and, in conjunction with findings from recent studies on CNN and LSTM, used to develop a comprehensive discussion of the prospect of using machine learning algorithms to detect security patterns in software code.

Keywords: *Security pattern, software code, detection, machine learning, security pattern detection (SPD), convolutional neural network, long short-term memory, security, vulnerability*

Table of Contents

List of Tables

List of Figures

CHAPTER 1: INTRODUCTION

1.1 Background

Software security is vital in today's digitized world for various reasons. Private and public sectors have a legal and moral obligation to protect corporate and customer data from unauthorized access. Organizations must implement innovative and effective strategies to enhance the resilience of their systems against cyberattacks. Organizations must protect their systems to avoid financial, reputational damage, and other risks of successful cyberattacks (Harrington, 2020). The process of maintaining the security of a software system is not straightforward. Alvi & Zulkernine (2021) acknowledge that securing a software system is a dynamic battle to protect legitimate access rights and prevent software misuse. The use of security patterns to secure software applications is not a new phenomenon. Previous research has shown that security patterns have served as excellent security features in various systems. A case in point is the Single Access Point (SAP) security pattern behind the "Login Window" (Alvi & Zulkernine, 2021). Detecting and documenting security patterns play a critical role in enhancing the design and implementation of a highly secure software system. It must be stressed that security patterns are the cornerstone of system security for organizations. This study focuses on the possibility of using machine learning algorithms to detect security patterns in software code.

1.2 Statement of the Problem

The use of big data in scientific and technology fields has increased rapidly in recent years. As a result, big data continues to grow as an area of research interest. According to Lee et al. (2020), the rapid growth of big data in the scientific and technological field is driven by the demand for companies to enhance their competence, innovativeness, and competitiveness in the

global marketplace. For instance, manufacturing companies use machine learning solutions such as a defect tracking system to improve the overall quality and reduce costs (Lee et al., 2020). A detection system founded on a deep learning algorithm can also help a company to automate decision-making and accurately assess business operations.

Although not as widely appreciated as big data, security patterns serve as a fundamental, micro-architectural security element in software code. When software testing and quality assurance are performed, security patterns are thoroughly examined. A security pattern must comply with established standards and comprehensive security patterns documentation requirements (Alvi & Zulkernine, 2021). Detecting security patterns is a prerequisite to measuring the quality of a security pattern in a software code. However, effective methods of detecting security patterns must first be used in order to achieve accurate detection results. The main problem taken up by our study is to investigate the effectiveness, performance, and accuracy of selected machine learning algorithms that have the potential to be used in detecting security patterns in software code.

1.3 Significance of the Thesis

This study aims to examine and analyze the performance and accuracy of Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) algorithms, which have the potential to be used for detecting security patterns. The analysis in this study focuses on the performance and accuracy of CNN and LSTM, which have so far been used to detect software vulnerability. The objective is to identify a potentially effective machine learning algorithm for detecting security patterns in a software code. The present study represents a unique and valuable

contribution to this field in that it employs primary data from selected experts in the fields of machine learning algorithms and security pattern detection.

The findings from this study would enable companies, developers, security experts, and relevant stakeholders to make accurate decisions when choosing a machine learning algorithm for detecting security patterns in software code. The input from the experts who were interviewed reveals the key differences and similarities between CNN and LSTM in terms of their performance and accuracy. The findings from the experts are analyzed, discussed, and compared with those of previous studies in order to provide more insights on the potential usage of CNN and LSTM algorithms in detecting security patterns.

1.4 Research Aim and Objectives

This study investigates the effectiveness, performance, and accuracy of machine learning algorithms that have the potential to be used for detecting security patterns in software code. The objective is to understand the likely performance and effectiveness of the selected machine learning algorithms – CNN and LSTM – if they are used for the purpose of detecting security patterns in software code.

Objectives

- To investigate the performance of CNN and LSTM in detecting software vulnerabilities and, by implication, security patterns.
- To examine the accuracy of CNN and LSTM in detecting software vulnerabilities and, by implication, security patterns.

- To conduct a detailed analysis of the potential performance and accuracy of CNN and LSTM in detecting security patterns.

1.5 Research Questions

The main research questions that guided our study are as follows:

RQ1: How effective can CNN and LSTM machine learning algorithms be in detecting security patterns?

RQ2: How good can the performance of CNN and LSTM algorithms be in detecting security patterns?

RQ3: How accurate can CNN and LSTM algorithms be in detecting security patterns?

RQ4: What is the possible variation in performance and accuracy of CNN and LSTM algorithms in detecting security patterns?

1.6 Assumptions

Li et al. (2020) assert that programming language is a language that facilitates communication between humans and computers. It is supposed that programming languages are founded on natural languages. As such, both programming and natural languages can be converted into syntax trees. The assumption is that words in a similar context have shared meanings. According to Li et al. (2018), most existing solutions for system security detection depend on human experts. In the present study, which investigates machine learning algorithms, a fundamental assumption is that software code can be converted into a syntax tree. Moreover, it

is assumed that software code must follow industry standards and rules in detecting security patterns, including the sequence of operation.

Furthermore, this study assumes that the same software vulnerability in software code have similar characteristics (Li et al., 2020). This assumption makes it easier to categorize the vulnerability identified in software code using machine learning algorithms. Once the detected vulnerabilities are categorized, an organization can effectively implement the best security solutions. Lastly, this study assumes that semantic analysis is the best model for using machine learning algorithms to detect security patterns in software code. I assume that semantic analysis will extract valuable information from the software code and enhance the accuracy of security pattern detection.

1.7 Definitions of Terms

There are numerous important terms in this study. Here is a list of the most salient terms.

- Software design pattern – standard solutions for typical software design and architecture problems (Nazar & Aleti, 2020).
- Security pattern – solutions to recurring security problems in software (Alvi & Zulkernine, 2021).
- CNN – Convolutional Neural Network. It is a deep learning algorithm used to analyze and differentiate visual images. The amount of pre-processing needed is lower compared to other classification algorithms.
- LSTM – Long Short-term Memory. It is an artificial recurrent neural network (RNN) used for classification, processing, and making predictions using time series data.

- Machine learning – a method of data analysis based on artificial intelligence and the concept that a system can learn and identify patterns using data (Sagar, Jhaveri, & Borrego, 2020).
- Vulnerability – a weakness or a fault in a software system that cybercriminals can exploit to execute malicious attacks.
- Software vulnerabilities – the faults in code that attackers can exploit to execute malicious intentions on the software (Chernis & Verma, 2018).
- Vulnerability detection – the process of evaluating a software system using a specific technology, tool, or method to identify faults that attackers can exploit to execute malicious intentions.
- Buffer overflow – the use of data in a buffer that exceeds its length, leading to a split of storage outside the buffer.
- Buffer overflow vulnerability – a security threat posed by data stored outside the intended buffer (Li et al., 2020).

1.8 Chapter Outline

The thesis will consist of five chapters: Introduction, Literature Review, Methodology, Findings, Discussion, and Conclusion. The Introduction outlines the background of the study and introductory elements, including the research aim and objective. The Literature Review section presents the findings of an examination of previous research on similar topics. The Methodology section presents the methodologies used and the justification for selecting the methods. It also outlines the research philosophy, the methods used, the data collection procedures, and the data analysis frameworks. The Findings section presents the study results and the discussion of the

results. The Conclusion summarizes the study's primary findings and suggests directions for future research.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

This section provides a critical review of current literature related to the prospect of using machine learning algorithms for security pattern detection in software code. The criteria for selecting the literature to review focused on current peer-reviewed journals relevant to the study. The keywords and phrases used to search for sources from credible databases included "security pattern detection," "using machine learning algorithms in security pattern detection," and "machine learning algorithms for detecting security patterns in software code." This chapter is organized systematically based on the themes across these sources. The primary themes identified from the critical review of literature related to this study include security pattern detection, machine learning algorithms for security pattern detection, and the effectiveness of machine learning algorithms in security pattern detection.

2.2 Security Pattern Detection

Security pattern detection and vulnerability detection are two different but related terms used in this study. A security pattern is a reusable solution for a specific security problem. Accurate detection of security patterns is necessary to build and maintain secure software (Alvi & Zulkernine, 2021). Vulnerability detection is the process of identifying security flaws in a system, network, or application. Attackers can exploit vulnerabilities to compromise the system and execute malicious actions.

Security pattern detection is one of the key topics explored in the literature review. According to Alvi & Zulkernine (2021), maintaining the security of a system is an active struggle to secure access and prevent misuse of access rights. Security patterns serve as micro-

architectural elements of security in the software development process. However, security patterns must be implemented according to established standards and documentation methods to achieve the best security outcomes.

Cybercriminals exploit flaws or vulnerabilities in a software code to compromise systems and access them illegally. According to Russell et al. (2018), numerous software vulnerabilities are reported annually on the Common Vulnerabilities and Exposures databases, while others are discovered internally in specific software codes. Software vulnerabilities can have disastrous financial and societal consequences if successfully exploited by cybercriminals.

The need to implement effective security strategies and solutions has grown dramatically in recent years. Vulnerabilities are the critical contributors to the increased cyberattacks on various systems and applications. Cybercriminals are keen on exploiting software vulnerabilities to compromise the integrity, confidentiality, and availability of information and information systems. The WannaCry ransomware is an example of a recent attack that exploited Windows server vulnerabilities to target health information systems (Li et al., 2020). The ransomware compromised several health information systems, causing service disruption as well as financial and reputational damage to the affected organizations.

Performing tests and other quality assurance procedures are critical in ensuring that such vulnerabilities are identified and fixed (Alvi & Zulkernine, 2021). Experts can recognize incomplete, imperfect, or misused security patterns in a software code through the tests. Quality assurance processes and tests play a crucial role in ensuring that weaknesses in security patterns are identified and that security patterns are implemented correctly.

Previous studies have utilized various methods to detect software vulnerabilities. A recent study by Li et al. (2020) used an automated and intelligent vulnerability detection method founded on minimum intermediate representation learning. The study transformed the source code to a minimum intermediate representation to eradicate irrelevant items and minimize the dependency length in the sample. Another study by Li et al. (2018) used VulDeePecker, an automated method of detecting vulnerabilities in software code using a deep learning-based system. The researchers chose the approach because deep learning automates the vulnerability detection process, including feature definition. The study enabled the researchers to explore the effectiveness of Vulnerability Deep Pecker (VulDeePecker) from multiple perspectives. For instance, the study examined whether VulDeePecker could deal with various vulnerabilities simultaneously.

Previous studies have employed several methods for detecting vulnerabilities in systems or software codes. Though the methods share the aim of detecting vulnerabilities, they greatly vary in accuracy, effectiveness, and other parameters. A study by Harer et al. (2018) utilized two complementary methods to detect C and C++ code vulnerabilities: build-based feature extraction and source-based feature extraction. The build-based approach takes advantage of features extracted from a compiler that comprehends language structure, while the source-based method leverages language processing and statistical correlations between codes. The way the study is structured and executed is consistent with previous studies. The researchers explored a vital issue: the method's effectiveness in detecting vulnerabilities in software codes. For instance, the study by Li et al. (2018) found that human expertise can play a crucial role in enhancing the effectiveness of VulDeePecker in detecting vulnerabilities in software code. The findings of Harer et al. (2018) and Li et al. (2018) show an element of consistency in the role played by

human expertise in improving the effectiveness of an automated solution for detecting vulnerabilities in a code.

2.3 Security Pattern Detection using Machine Learning Algorithms

Previous studies used machine learning algorithms to detect vulnerabilities in a software. A study by Lee et al. (2020) used the Long Short-Term Memory (LSTM) algorithm based on Rising Time (RT) and Falling Time (FT) to diagnose faults in digital sensors. The researchers found the method nearly 50% more reliable than rule-based fault diagnosis in digital sensors. A more recent study by Li et al. (2021) used Syntax-based, Semantics-based, and Vector-Representations (SySeVR) framework based on deep learning to detect vulnerabilities in C and C++ codes. Their study explored how programs can be represented as vectors that accommodate syntax and semantic information to detect vulnerabilities effectively. The latter research aimed to improve production safety and reduce costs, while the former research focused on detecting vulnerabilities and securing software codes from attacks.

Software vulnerabilities are currently a significant concern in the IT industry. The concern is massive, especially in security-sensitive programs. Attackers can identify software vulnerabilities and capitalize on them to execute various malicious activities on the system. For instance, attackers may exploit software vulnerabilities to perform denial of service (DoS) attacks and remotely control the system operations (Chernis & Verma, 2018). Over the past few years, there has been a rapid increase in machine learning solutions to automate analyzing and examining programs for software vulnerabilities. A key driver for this shift is the availability of large open-source codes for analysis (Russell et al., 2018). Experts can learn about software vulnerabilities and accurately identify them with free access to many codes. Furthermore,

professionals maximize natural language processing (NLP) approaches because of the link between program languages and natural language.

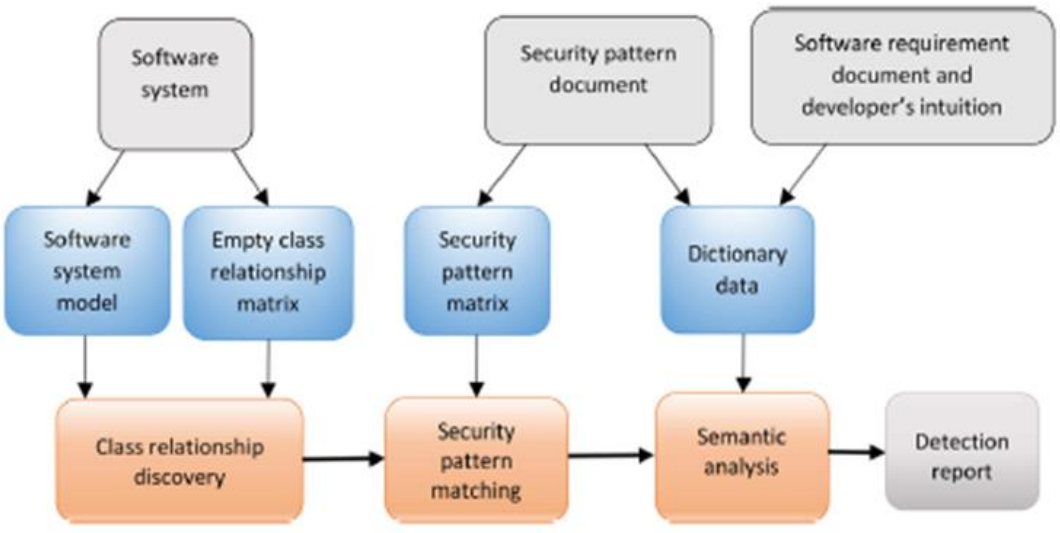
A critical review of existing literature reveals that researchers' research questions and vulnerabilities datasets are consistent, even if there are variations in the machine learning methods employed. A study by Li et al. (2018) focused on three critical objectives: the ability of VulDeePecker to deal with multiple vulnerabilities simultaneously, the effect of human expertise on the effectiveness of VulDeePecker, and the effectiveness of VulDeePecker compared to other approaches for vulnerability detection. Similarly, another study by Li et al. (2021) focused on four research questions: the ability of SySeVR to detect multiple vulnerabilities, the effectiveness of SySeVR, effects of accommodating control-dependency, and the effectiveness of SySeVR compared to other methods. An analysis of the findings of these studies reveals the variations and effectiveness of different machine learning methods used currently in detecting vulnerabilities in software code.

Previous studies have also noted that in cases where software problems are recurrent, the solution is to reuse valid solutions. According to Silva-Rodríguez et al. (2019), implementing an effective security solution is a continuous process that covers all software development processes. Security professionals must document all the software requirements to increase the likelihood of successfully identifying security problems in software. However, due to the increased heterogeneity of interactive systems today, companies need to capitalize on innovative models to secure their unique codes. For instance, a specific software code may have different object orientation attributes like an inheritance. Source codes also have unique identifiers such as class names, methods, and construct types (Nazar & Aleti, 2020). Examining these attributes is necessary to identify the design patterns and potentially the security patterns in a specific code.

Understanding such characteristics is likely to help in conducting security pattern detection using machine learning algorithms.

Alvi & Zulkernine's (2021) study proposed a security pattern detection (SPD) framework that provides a platform for data extraction, semantic analysis, and pattern matching. The framework can process security patterns and target software data to detect security patterns through initial, intermediate, and final processing. The framework works similarly to the IDPatternM model proposed in the Silva-Rodríguez et al. (2019) study. The IDPatternM model works by identifying design patterns based on defined software requirements. However, SPD is a unique and more effective framework for security pattern detection because of the ability to discover the class relationship, match patterns, and perform semantic analysis using dictionary data (Alvi & Zulkernine, 2021). An analysis of the experiments' findings using different machine learning methods and algorithms provides precise insights into the effectiveness and challenges of various approaches.

Figure 1: Overview of the Security Pattern Detection (SPD) Framework



(Alvi & Zulkernine, 2021)

Figure 1 provides an overview of the security pattern detection (SPD) framework. The critical elements of the model include software system, security pattern document, and software requirement document. The framework uses components of security pattern and class relationship to discover class relationships, match security patterns, perform semantic analysis, and generate detection reports (Alvi & Zulkernine, 2021). The detection reports generated are different from what other methods offer. For instance, source-based models use natural language processing capabilities to learn statistical correlations in code (Harer et al., 2018). However, the source-based method does not understand the semantics of the code. The challenge is addressed by build-based models of pattern detection, which uses a compiler that comprehends the language structure. An analysis reveals that a combination of different methods enhances the accuracy of detecting vulnerabilities in software code.

2.4 The Effectiveness of Machine Learning Algorithms in Detecting Security Patterns

The effectiveness of machine learning algorithms in detecting security patterns is one of the issues that existing research has not explored. Most studies in this field have only focused on the effectiveness of a specific machine learning algorithm in detecting vulnerabilities in a particular code or programming language. As a result, there is a need to compare the effectiveness of different machine learning algorithms to determine their effectiveness. Notably, a recent study by Ndichu et al. (2019) focused on investigating the effectiveness of the machine learning approach in detecting JavaScript vulnerabilities using Abstract Tree Syntax (AST) and paragraph vectors.

Ndichu's study notes that websites such as online libraries and e-commerce sites attract millions of users because of their convenience. Most of these websites use the JavaScript

programming language to create interactive and dynamic content (Ndichu et al., 2019). However, the software code is highly susceptible to cyberattacks. Attackers can easily exploit such vulnerabilities and insert malicious codes into websites. The gap in the existing literature in terms of exploring the use of machine learning solutions to detect security weaknesses in software codes is further highlighted in several studies. According to Al-Garadi et al. (2020), numerous researchers have studied Internet of Things (IoT) security, but most fail to focus specifically on machine learning applications to enhance IoT security. There is a need to investigate the prospect of using machine learning methods to detect vulnerabilities in different software codes.

One of the critical issues to consider when investigating the effectiveness of machine learning algorithms is how the technology works. According to Nazar & Aleti (2020), machine learning learns automatically from structural and semi-structural data. A text is an example of structural data, whereas the source code is semi-structural. In that regard, machine learning algorithms vary depending on inputs, outputs, and problems they can solve. A second issue to consider is how machine learning algorithms are developed. According to Labonne (2020), creating a machine learning algorithm entails two crucial steps: training and testing. MATLAB, Scikit learn, and PyTorch are some of the frameworks used to manage the design of machine learning algorithms (Labonne, 2020). The models developed can be used to perform various tasks, including regression, classification, and reconstruction in intrusion detection. Understanding how machine learning algorithms are developed provides further insight into how to enhance their effectiveness.

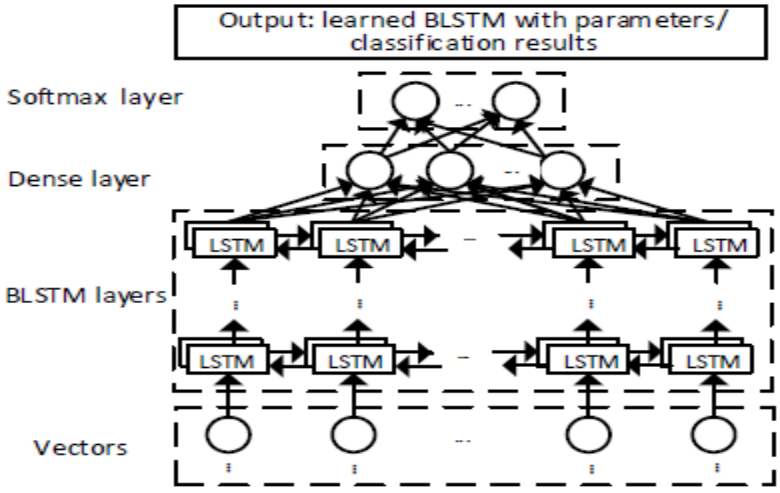
The shift from conventional methods to machine learning methods for detecting vulnerabilities in a system is motivated by various factors. Ullah et al. (2019) argue that though

traditional methods address code obfuscation, visualizations of malware involve high computational costs. The shift is also motivated by the increasing challenge of detecting malware due to the regular updates and manipulations. A recently published article by Wu, Wei, & Feng (2020) acknowledges the considerable role of deep learning solutions in detecting cyberattacks today. The article details the application of deep learning algorithms in intrusion detection, malware detection, and domain generate algorithms. It demonstrates that deep learning and machine learning are applicable in performing various cybersecurity functions.

2.5 Implementation of Security Patterns in Software Development

The use of deep learning-based solutions in detecting vulnerabilities in software code is guided by a set of principles. Li et al. (2020) presents preliminary studies for detecting vulnerabilities using deep learning. The principles focus on answering three critical questions related to the representation of software programs, appropriate granularity, and selection of neural networks. The guiding principle for representing software transforms it into an intermediate representation and later into a vector representation. The transformation preserves semantic relationships of the program's components and makes them ready to feed into neural networks. Another, more comprehensive study by Russell et al. (2018) examines neural networks and representation learning classification. The study acknowledges the commonalities that exist between source code and natural languages. The study thus incorporates methods designed for natural language processing. The findings of that study demonstrate how machine learning can be effectively used to detect vulnerabilities directly from the source code.

Figure 2: Review of the BLSTM Neural Network



(Li et al., 2018)

Figure 2 provides an overview of the Bidirectional Long Short-Term Memory (BLSTM) neural network structure, including the layers. The primary layers are dense and softmax layers, which have forward and backward directions (Li et al., 2018). The layers have unique functions. For instance, the softmax layer receives vectors from the dense layer, represents the classification result, and updates the neural network parameters.

The design, development, and implementation of security features in software code is an issue that has been highlighted in several existing studies. According to Alvi & Zulkernine (2021), the correct implementation and application of security patterns requires that the patterns follow detailed documentation available in the appropriate catalogs. Quality assurance experts check the patterns when inspecting and testing systems for quality issues. However, Alvi & Zulkernine (2021) notes that the inspection processes may be achieved without detecting the patterns in the software. Most similar studies that have been reviewed provide detailed analysis and discussion of how deep learning algorithms are applied in pattern detection. The study by Lee et al. (2020) examines how to detect patterns in data using hybrid Convolutional Neural

networks (CNN) and machine learning algorithms. That study is unique because it responds to the rapidly increasing research in big data in technology and scientific fields. The findings of that study provide a strong foundation for detecting big data patterns in the manufacturing industry.

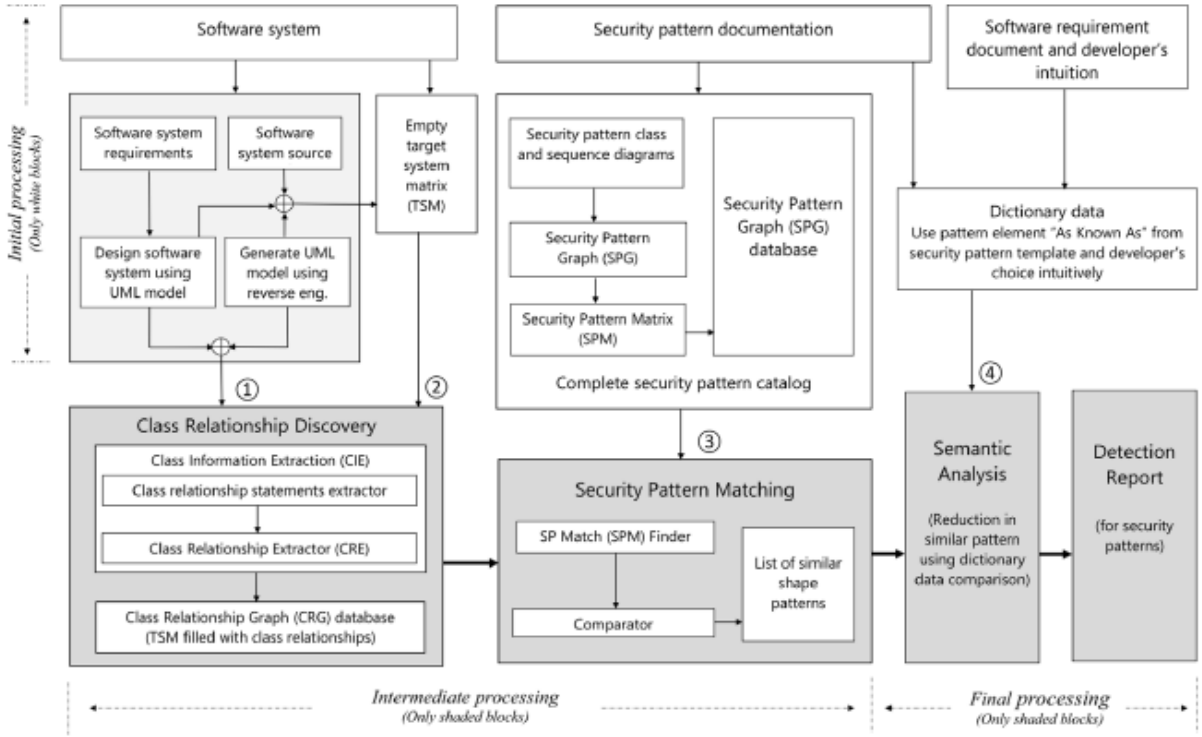
Three primary motivations drive most previous studies in this field: reverse engineering, demonstrating patterns, and finding bad coding practices (VanHilst & Fernandez, 2007). However, most of these studies fail to clarify the differences between Gang-of-Four (GoF) and security patterns. Understanding the differences in the patterns plays a crucial role in pattern detection. Multiple factors cause the vulnerabilities in software. For example, buffer overflow vulnerability is caused by data stored outside the intended buffer (Li et al., 2020). Moreover, a better understanding of the implementation and testing of machine learning algorithms in detecting software vulnerabilities requires using a specific dataset. For instance, to study the effectiveness of SySeVR, Li et al. (2021) used a dataset of 126 unique vulnerabilities collected from the Software Assurance Reference Dataset (SARD) and the National Vulnerability Database (NVD). The study focused only on data of independent value and publicly available data. Using an appropriate dataset size ensures that the findings can be replicated and applied to detect many vulnerabilities in a code.

2.6 Security Patterns and Resilience Against Cyberattacks

Figure 3 shows the fundamental structure of the security pattern detection (PSD) framework. The setup consists of initial processing, intermediate, and final processing. Different operations occur at each processing stage. The initial processing stage is where input file preparation, security pattern matching, and semantic analysis occur. An empty target system matrix (TSM) is prepared using the available list of class names in the source code or software

model diagram. Class relationships are discovered at the intermediate processing stage, and security patterns are matched (Alvi & Zulkernine, 2021). For example, a class relationship extractor (CRE) subunit is used to mine the class-related information. In the final processing stage, semantic analysis is conducted and detection reports are generated. Semantic analysis is performed using dictionary data to reduce false positives.

Figure 3: Comprehensive Security Pattern Detection (PSD) Framework

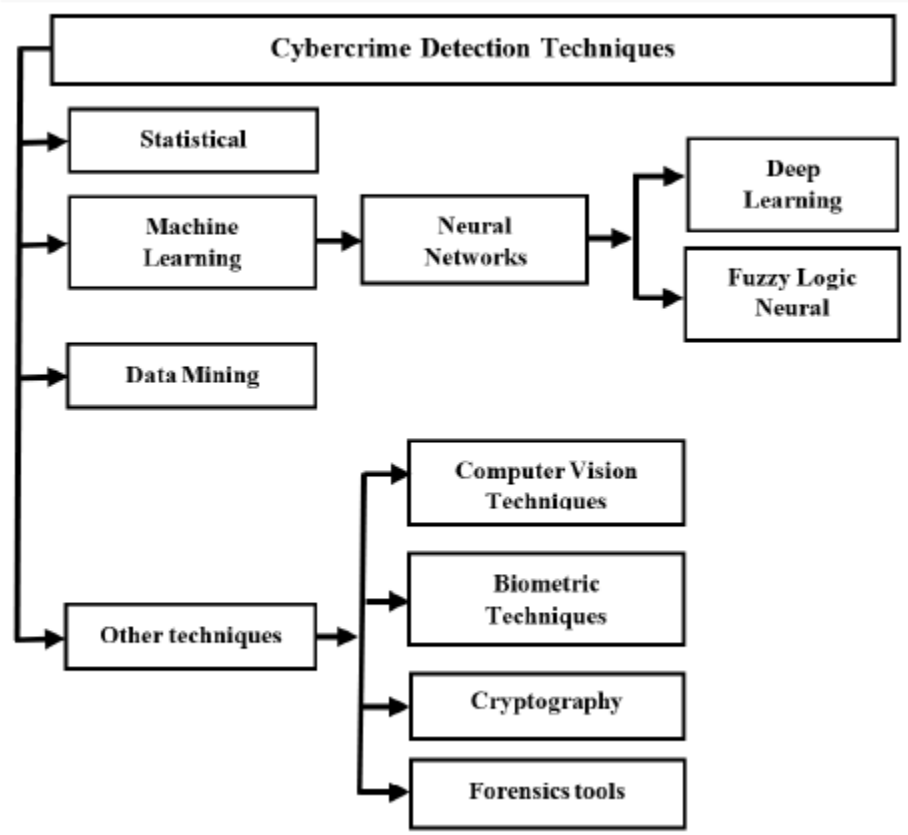


(Alvi & Zulkernine, 2021)

Understanding the different types of cybercrime detection techniques provides significant insights into the effectiveness of machine learning and other methods explored in various studies. Figure 4 below summarizes the categories of cybercrime detection techniques available currently. Al-Khater et al. (2020) acknowledge the recent rapid increase in cybercrimes that render conventional detection methods ineffective in detecting and mitigating them. Their study

highlights various machine learning methods used to detect cybercrimes across different domains. The techniques highlighted include supervised machine learning, the Levenshtein algorithm, naïve Bayes classifier, and decision tree (Al-Khater et al., 2020). For instance, researchers have employed the support vector machine and naïve Bayes to detect cyberbullying in tweets.

Figure 4: Categorization of Cybercrime Detection Techniques



(Al-Khater et al., 2020)

Furthermore, many recent studies emphasize the need to effectively use different machine learning algorithms and methods to detect vulnerabilities in software codes. A recent survey by Arsan (2021) uses statistical analysis and a deep learning system to detect malware in a dataset

of 7,622 applications. Multiple tests were performed by Arsan using machine learning techniques for comparison with the deep learning method.

Figure 5 below summarizes the comparison of the machine learning and deep learning methods that have been used to detect vulnerabilities in previous studies. The figure shows that most of the studies in 2019 and 2020 used artificial intelligence modeling. However, the primary differences in these studies lie in the dataset used and the feature vectors obtained. For instance, some studies only used static property, while others used activity services, intent filters, and API calls (Arslan, 2021). The findings of this study are consistent with the majority of previous studies. For example, deep learning and the latest machine learning methods have been deemed to have a vast potential to transform the detection of software vulnerabilities today. However, deep learning has not achieved the revolutionary breakthroughs hoped for by many in this field (Musser & Garriott, 2021). A key reason for this is the rapid increase in the complexity of cyber threats encountered by organizations. Such observations call for combining different methods to detect and manage the latest software vulnerabilities.

Figure 5: Comparison of Classification Performance of Former and Proposed Methods

Table 4 The comparison of classification performance among former methods and proposed method.

Similar works	Selected features	Num of benign apps	Num of malware apps	Num of neurons or classification method	Precision	Recall	Accuracy	F-measure
ASAEF (*Zhang, Thing & Cheng, 2019*)	Metadata, permissions, intent filter, activity, services	37,224	33,259	N-gram, signature	96.4%	96.1%	97.2%	96.2%
FingerPrinting (*Karbab, Debbabi & Mouheb, 2016*)	Family DNA	100	928	Signature	89%	84%	N/A	85%
DroidChain (*Wang et al., 2016*)	Permissions, API call, behaviour chain	–	1,260	Warshall	91%	92%	93%	N/A
Shhadat (*Shhadat et al., 2020*)	Heuristic strategy, dynamic analysis	172	984	RF	96.4%	87.3%	97.8%	91.2%
DroidDet (*Zhu et al., 2018*)	Permissions, system events, sensitive API and URL	1,065	1,065	SVM	88.16%	88.40%	88.26%	N/A
DL-Droid (*Alzaylaee, Yerima & Sezer, 2020*)	Application attributes, actions, events, permissions	11,505	19,620	300, 100, 300	94.08%	97.78%	94.95%	95.89%
SRBM (*Liu et al., 2021*)	Static and dynamic feature	39,931	40,923	RBM	–	–	0.804	84.3%
Lu (*Lu et al., 2021*)	API calls	1,400	1,400	Correntropy, CNN	95.0%	76.0%	84.25%	84.0%
ProDroid (*Sasidharan & Thomas, 2021*)	API calls	500	1,500	HMM	93.0%	95.0%	94.5%	93.9%
Proposed model DL (376502(300,300,300,300))	Application permissions	961	6,661	300, 300, 300, 300	98.9%	99.1%	98.03%	99.0%

(Arslan, 2021)

The School Semester Scheduling system (SSS) illustrates the procedure for detecting security patterns in a study by Li et al. (2021). The system has six classes linked bi-directionally or in a directed association. The classes include the User, Login, Student, Admin, Teacher, and Semester Schedule. (Refer to Appendix G for the SSS target system class diagram.) Li et al.'s study implements an Ordered Matrix Matching (OMM) and Non-uniform Distributed Matrix Matching (NDMM) techniques to detect vulnerablities in a target system.

Table 1: Effectiveness of VulDeePecker vs. SySeVR-enabled BLSTM

Method	Kind of SyVC	FPR	FNR	A	P	F1	MCC
VulDee-Pecker	FC-kind	5.5	22.5	90.8	79.1	78.3	72.5
SySeVR-BLSTM	FC-kind	2.1	17.5	94.7	91.5	86.8	83.6
	AU-kind	3.8	17.1	92.7	88.3	85.5	80.7
	PU-kind	1.3	19.7	96.9	87.3	83.7	82.1
	AE-kind	1.5	18.3	96.6	87.9	84.7	82.9
	All-kinds	1.7	19.0	96.0	88.0	84.4	82.2

(Li et al., 2021)

Table 1 above highlights the results of an experiment conducted by Li et al. to compare the effectiveness of VulDeePecker and SySeVR-enabled BLSTM for detecting vulnerabilities. The results show that SySeVR-BLSTM leads in the detection of FC-kind and AU-kind, with F1-MEASURE 86.8% and MCC 83.6% (Li et al., 2021). The comparison also shows that SySeVR-BLSTM achieves a slightly lower FPR and FNR than VulDeePecker when used to detect the same vulnerabilities.

Table 2: Results Showing VulDeePecker Can Detect Multiple Vulnerabilities

Dataset	FPR(%)	FNR(%)	TPR(%)	P(%)	F1(%)
BE-ALL	2.9	18.0	82.0	91.7	86.6
RM-ALL	2.8	4.7	95.3	94.6	95.0
HY-ALL	5.1	16.1	83.9	86.9	85.4

(Li et al., 2018)

Table 2 above summarizes the findings showing that VulDeePecker can simultaneously detect multiple types of vulnerabilities. However, the number of API function calls associated with the exposures determines the effectiveness in detection (Li et al., 2018). The experiment shows that the fewer the API function calls, the more effective VulDeePecker is. Refer to Appendix A for a detailed table showing the effectiveness of VulDeePecker in detecting various types of vulnerabilities.

2.7 The Challenges of Using Machine Learning Algorithms in Detecting Security Patterns

Despite the considerable benefits of machine learning methods in detecting software vulnerabilities, most of the models are far from perfect. According to Oseni et al. (2021), most machine models presently are susceptible to problems that threaten the confidentiality, availability, and integrity of systems. These problems can occur in the training and testing phases of the machine learning model. For instance, the dataset can be manipulated by changing the data label or input features at the training stage. Binary code analysis (BCA) is an example of a new vulnerability analysis and detection technique based on machine learning. The primary advantage of BCA is that it enables software engineers to analyze program binaries with limited access to the source code (Xue et al., 2019). However, the technique entails numerous challenges, including the need for cross-platform analysis and high scalability. Though BCA is

an effective machine learning technique for analyzing program binaries with limited access to source code, it has its set of drawbacks.

The previous studies reviewed reveal that researchers used different data collection methods to gather the required information regarding the use of machine learning methods to detect software vulnerabilities. One study approved by Microsoft's Ethics Advisory Board used interviews and surveys to find out about machine learning in software engineering (Amershi et al., 2019). Since machine learning in software engineering is still an emerging topic, the interview method enabled the researchers to use a snowball sampling strategy to select the participants. The sampling strategy enabled the researchers to involve respondents from different company divisions, including software engineering and data and applied sciences. Another study by Khalid et al. (2018) used a dataset to perform experiments. Table 3 shows the total number of web vulnerabilities in the dataset used in this study.

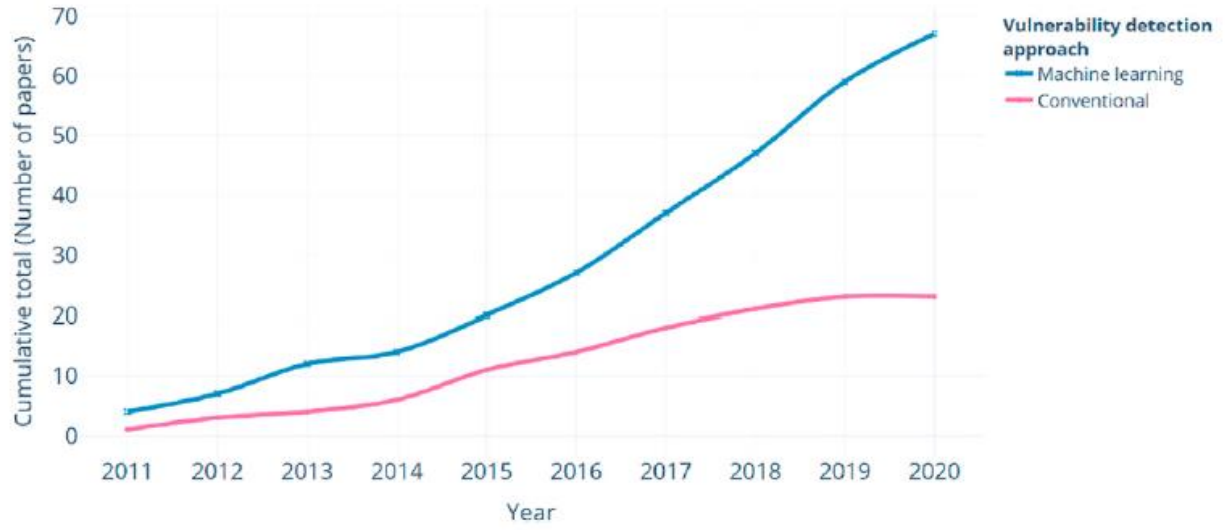
Table 3: Total Number of Web Vulnerabilities in Dataset

Application	Vulnerable files	Total files	P-rate (%)
Drupal	62	202	30.68
PHPMyAdmin	27	322	8.39
Moodle	24	2942	0.82

As presented in the table 3 dataset, out of 233 vulnerabilities, 19 are code injections that allow attackers to manipulate the code randomly. For instance, attackers can change HTTP headers and server-side variables using code injections (Khalid et al., 2018). In this way, attackers can have the chance to execute malicious HTML and hijack sessions.

Recent research by Hanif et al. (2021) shows a consistently increasing trend in the number of studies implementing machine learning approaches over conventional approaches in vulnerability detection. Between 2011 and 2020, 67 papers implemented machine learning methods, including deep learning, supervised, and semi-supervised learning. Figure 6 shows the cumulative total number of papers vs. year (2011-2020).

Figure 6: Categories of Vulnerability Detection Approaches



(Hanif et al., 2021)

Though machine learning approaches for detecting software vulnerabilities have increased in recent years, the detection process remains challenging. According to Hanif et al. (2021), the critical reason for this is that most new projects lack source codes or vulnerabilities history. As a result, machine learning models are compelled to predict zero-day vulnerabilities at the development stage. Oseni et al. (2021) have observed that most attacks on machine learning and deep learning models take the form of manipulation. For instance, the attackers aim to manipulate the inputs of the machine learning approach to misclassify data. However, the

manipulation varies depending on the task category of the algorithm. Software engineers must be conversant with the task categories of machine learning algorithms to understand and effectively deal with the potential challenges that stem from using such algorithms to detect vulnerabilities.

Cloud computing is one of the other areas where machine learning algorithms are presently applied. The shift towards cloud computing has increased rapidly over the years. Organizations and individuals understand the significance of cloud computing infrastructure, especially in solving storage needs. However, despite the benefits, cloud computing is vulnerable to security attacks. The cloud holds vast personal and corporate information and must be protected (Nassif et al., 2021). Machine learning algorithms and techniques are currently being used to detect and prevent cloud security attacks. Moreover, machine learning methods are being used to detect pirated software using the source codes. According to Ullah et al. (2019), deep learning approaches are designed to detect plagiarism in source codes to identify pirated work, suggesting that machine learning and deep learning techniques are applicable across multiple domains.

Similarly, a study by Ullah et al. (2019) highlights the challenges of using deep learning methods in detecting software piracy. A key challenge noted is that the uniqueness of syntax and semantic structures of source codes makes it difficult to detect piracy. The processing of source codes involves deep analysis, stemming, frequency extraction, and other activities. Machine learning techniques are also applied in the detection of website vulnerabilities. Research by Khalid et al. (2018) acknowledges that today, designing a secure website is expensive, time-consuming, and challenging. Website owners must develop and implement effective strategies to secure their websites from cyberattacks. Most website owners and security providers are opting for machine learning models to detect and address web vulnerabilities.

Moreover, a study by Shaukat & Ribeiro (2018) provides a detailed analysis of how to use machine learning in creating a layered defense system against cryptographic ransomware. The study acknowledges the recent increases in expensive and damaging attacks worldwide by cybercriminals using cryptographic ransomware. The attacker encrypts user files and demands a huge ransom to provide decryption keys. The researchers analyzed an extensive dataset of ransomware families in RansomWall using a static and dynamic analysis hybrid to characterize the ransomware behavior (Shaukat & Ribeiro, 2018). According to the study, machine learning methods provide a more effective strategy for detecting zero-day attacks than antivirus programs. The findings of this study are consistent with most of the previous studies that were reviewed. For instance, the results are consistent with Choi et al.'s (2020) findings, which acknowledge the agility of deep learning methods in detecting security vulnerabilities. The challenges are associated with different factors, including unbalanced and hierarchical data structures in program analysis. Future studies need to focus more on creating hybrid machine learning and deep learning methods to effectively detect and address software vulnerabilities.

Previous studies have also explored various software engineering challenges at the intersection of machine learning and software engineering. The primary issues noted include development challenges, limited transparency, troubleshooting, and resource limitations (Arpteg, Brinne, Crnkovic-Friis, & Bosch, 2018). The development challenges vary depending on the requirements of each method. Machine learning methods are developed to automatically program a system using data and not to write source codes. As a result, creating a machine learning model requires multiple experiments to identify the optimal model. The challenges are also linked with how the machine learning method works. A study by Kumar & Lim (2019) proposed a machine learning algorithm that detects malware activity by scanning traffic patterns. The machine

learning algorithm used was Early Detection of IoT Malware Network Activity (EDIMA). Though the algorithm is effective, it checks for traffic patterns over several months to detect and isolate vulnerabilities. Cybercriminals can utilize the latest software to exploit software vulnerabilities and execute various attacks. (Refer to Appendix C for a modular structure of EDIMA algorithm.)

The use of machine learning methods is gaining momentum in terms of recognizing binary codes in software. According to Wang, Wang, & Wu (2017), machine learning techniques enable security experts to detect and construct code patterns. However, that study notes various challenges with the existing literature. A fundamental problem highlighted is that most current research uses syntax features in model training (Wang, Wang, & Wu, 2017). The researchers, therefore, proposed a semantics-based approach called FID. The method identifies functions in stripped binaries. However, the researchers fail to clarify the meaning of FID used in binary code. Their study would have been more cogent if it included definitions of key terminologies, including FID. (Refer to Appendix D for the workflow of FID.)

Most of the studies reviewed did well in presenting their findings using various tools and techniques. For instance, a study by Li et al. (2020) uses tables to compare the different results of the experiments conducted. Table 3 and table 4 outline the comparison of representation learning models and classification algorithms. (Refer to Appendix E and F for tables 3 and 4, respectively.) In addition, Sarker (2021) has done exemplary work using tables, figures, graphs, and other tools to organize and present findings. The study understands the role of artificial intelligence in developing intelligent and automated applications today, specifically machine learning. Sarker (2021) also focused on examining the application of machine learning algorithms in improving the intelligence and capabilities of data-driven applications. The

presentation of data and findings using graphical and other effective means makes it easy for the readers to understand and apply the content.

2.8 Summary of Literature Review

In brief, the literature review section provided a critical analysis of numerous sources related to the topic under investigation. The primary topics treated in the previous studies include security pattern detection, vulnerability detection using machine learning algorithms, and the effectiveness of machine learning algorithms in detecting software vulnerabilities. For instance, the effectiveness of SySeVR-BLSTM is compared to VulDeePecker (Li et al., 2021). The section also provided a critical review of previous research methods, findings, and instruments. Notably, some prior research uses surveys to investigate the effectiveness of machine learning algorithms. For instance, a study by Shaukat et al. (2020) used surveys to investigate the effectiveness of machine learning methods, techniques, and tools for cybersecurity. The surveys show that machine learning techniques and tools are effective in spam detection, intrusion detection, and other cybersecurity applications (Shaukat et al., 2020). The other significant topics explored include implementing security patterns in software development, resilience against cyberattacks, and the challenges and opportunities of using machine learning algorithms in vulnerability detection.

CHAPTER 3: METHODOLOGY

3.1 Introduction

This chapter explains the methods used in this study and how the data was collected and analyzed. The chapter outlines the research design and how experts in machine learning and pattern detection were chosen for participation in the virtual interviews. The purposive sampling technique was used to select experts to participate in the voluntary virtual interviews. Their responses to the interview questions were noted and analyzed. The chapter also includes a discussion of reliability, validity, ethical considerations, and methodological limitations. Refer to Appendix H for the interview questions.

3.2 Research Design, Approach, and Strategy

The study aimed to investigate the possible use of machine learning in detecting security patterns in software code. According to Basias & Pollalis (2018), qualitative research consists of research methodologies that investigate a phenomenon by analyzing a specific sample population's behaviors, experiences, and opinions. Qualitative research answers what, how, where, and when questions (Basias & Pollalis, 2018). A qualitative research design was used to collect primary data from a sample of experts of security patterns and machine learning algorithms.

Qualitative research is an effective method for understanding a specific phenomenon (Rutberg & Bouikidis, 2018). Using a qualitative research method facilitates understanding the

effectiveness and performance of various machine learning algorithms in detecting software vulnerabilities. Qualitative research is an umbrella approach used to explore a specific phenomenon and gain a deeper understanding (Aspers & Corte, 2019). Qualitative research is appropriate for investigating various performance aspects of machine learning algorithms that might be used for security pattern detection.

The research strategy outlines how the researcher plans to collect the data to answer the research questions or test the formulated hypotheses (Saunders, Lewis, & Thornhill, 2009). In this qualitative study, virtual interviews were used to collect primary data from a sample of experts in machine learning and security pattern detection.

3.3 Data Collection

Data collection is a vital part of a research project. Considering the complexity of conducting experiments using machine learning algorithms, the data collected in this study were categorized into primary and secondary data. The primary data was collected using semi-structured interviews that were administered virtually. The secondary data was collected from peer-reviewed articles, published literature, and credible sources.

This study focused on analyzing the effectiveness of machine learning algorithms in detecting software vulnerabilities. This study compares the relative performance and accuracy of CNN and LSTM algorithms in vulnerability detection. According to Bukhari (2011), a comparative analysis examines, analyzes, and compares two or more subjects or ideas. The analysis explores the similarities or differences between the two subjects. Comparative analysis is an effective approach to determining and quantifying the selected variables (Bukhari, 2011).

The CNN and LSTM variables analyzed in this study are performance and accuracy. A specific criterion for analyzing the accuracy and performance of each algorithm in detecting software vulnerabilities was followed. The analysis was conducted primarily to differentiate the effectiveness of CNN and LSTM in detecting software vulnerabilities. The findings based on the virtual interviews would enable developers, security experts, and other interested parties to make more accurate decisions when choosing algorithms for security pattern detection.

3.3.1 Sampling Technique

The purposive sampling technique was used to select a sample of experts to participate in this study. The primary target participants were experts in security pattern detection and machine learning algorithms. A key condition was that an expert must have at least a B.S. degree in Computer Science or a related field to qualify to participate in the study. Two professors from an IT internship program were selected for the interview. These professors also recommended their former students who qualified for this study. Virtual interviews were preferred because of the COVID-19 pandemic.

3.3.2 Primary Data

The collection of primary data was guided by the qualitative research method. A sample of purposively selected experts in machine learning and security pattern detection was interviewed virtually. The two professors who were selected to participate in this study were from an IT internship program. The professors are in their 50s, with a Ph.D. degree and 15+ years of research experience in the IT field. The professors also recommended their former students, who had at least a B.S. degree in Computer Science or a related field, for participation in the interviews. Those experts who were eventually selected have worked for at least five years as computer security professionals. Two of the former students were in their 30s, and one in the

40s with an M.S. degree in Computer Science. One of these former students works at a small company and mainly does security consulting as a member of a team. Another works in an entertainment firm and mainly manages the company systems. The third expert, who works in a technology firm offering cybersecurity services, develops security solutions, performs risk assessment, and offers cybersecurity consultation services.

All of the experts were presented with semi-structured interviews designed to answer the research questions and provide more insights into the research topic. The semi-structured interviews provided valuable insights from the experts selected to participate in the study.

3.3.3 Secondary Data

A large amount of secondary data was used to enrich the findings from the virtual interviews conducted in this study. The secondary data was collected from multiple credible sources, including peer-reviewed articles, case studies, and recently published literature. The credible sources were also retrieved from library and trusted online databases, including EBSCO Host and Google Scholar. The keywords used during the search include “security pattern,” “security pattern detection,” “machine learning algorithms,” and “software code.” The secondary data includes results from recently published experiments conducted by reputable researchers and research institutions.

3.4 Interviews

The primary objective of purposive sampling was to ensure that experts with good knowledge of machine learning and security pattern detection were involved in this study. The expertise, experience, and knowledge of the experts are vital to enhancing the credibility of the

findings. The interviews consisted of 10 open-ended questions (Refer to Appendix H for the interview questions). Online interview was preferred because it is a flexible and cost-effective method of collecting data (Krouwel, Jolly, & Greenfield, 2019). It was also substantially easier for the experts to participate in the brief interviews.

Gray et al. (2020) state that qualitative data collection methods such as online video conferencing enhance accessibility to participants by eliminating the need to travel to a specific location. During the COVID-19 pandemic, various restrictions were introduced to curb the spread of the virus. Using virtual interviews in the study played a vital role in enhancing access to the selected experts. However, Gray et al. (2020) note that one challenge with online video conferencing is that it requires access to a stable and reliable internet connection. Online interviews may limit access to a more diverse sample to participate in a study.

The experts who agreed to participate in the virtual interviews were briefed on the purpose of the study. They were informed that the purpose of the study was to investigate the potential use of machine learning algorithms in detecting security patterns in software code. The experts were also informed that details about them would not be included in the final report because the study complies with research ethics. The semi-structured interviews were preferred because the open-ended questions enabled the experts to provide their unique responses to the specific issues under investigation. In fact, the experts responded differently about the feasibility of machine learning algorithms in detecting security patterns. Refer to Appendix H for the complete interview guide and questions.

3.5 Data Analysis

The primary data collected during the virtual interviews with the experts were transcribed for analysis. Thematic analysis is the analytical method used in this study. Thematic analysis is an effective method of analyzing interview findings to discover a deeper understanding and meaning of the data collected (Friis-Liby & Cressy, 2019). The interview transcripts containing the responses of each expert were analyzed to identify key themes. The details of the interview findings are presented in chapter 4.

3.6 Ethical Considerations

3.6.1 Consent

Consent is a crucial ethical consideration when conducting research involving human subjects. During this study, individual consent was sought before administering the online interviews. The experts were briefed about the purpose of the study and asked to participate. They were also informed that their participation in the study was voluntary. The experts who agreed to participate in the study voluntarily were interviewed.

3.6.2 Confidentiality

Confidentiality is also essential when carrying out research involving human subjects. The experts approached for the online interview were informed that none of their personal information would be included in the final report. No personal information like names and contacts was collected. The aim was to enhance the confidentiality of the research participants.

3.6.3 Data Protection

The experts who agreed to participate in this study were informed that their data would be protected in compliance with research ethics involving human subjects. The participants were informed that their personal information and responses would not be used outside this research project. During the online interviews, the participants were informed that the research complies with applicable data protection laws.

3.7 Reliability and Validity

Data collection and analytical research methods are prone to having reliability and validity issues. A researcher must identify the potential reliability and validity issues in research and implement the most effective strategies to address them (Saunders, Lewis, & Thornhill, 2009). In this study about the potential use of machine learning algorithms for security pattern detection, the key issues considered are participant bias and participant error. Participant bias and error can significantly affect the credibility and accuracy of the research findings. The confidentiality and privacy of the study participants were enhanced to minimize the risk of participant bias.

Furthermore, Harrell & Bradley (2009) assert that the interview is an effective method for gathering primary data based on the opinions and experiences of the selected participants. Interviews are also used to tap into the knowledge and expertise of a specific individual in a particular topic under investigation (Harrell & Bradley, 2009). Interviewing the selected experts in the fields of security patterns and machine learning algorithms minimized response errors in

this study. Focusing on the input of the experts chosen played a vital role in enhancing the reliability and validity of the study.

3.8 Summary

In brief, Chapter 3 outlines the methodology and the techniques used in the study. A purposively selected machine learning and pattern detection sample were interviewed, thus yielding important primary data. Chapter 4 of this thesis presents the data collected from the interviews. Previous studies investigating the use of machine learning algorithms in vulnerability detection are also highlighted to enrich this study.

CHAPTER 4: FINDINGS

4.1 Introduction

Chapter 4 presents the findings of the study. The responses of the selected experts in machine learning and security pattern detection are presented. The response rate and a descriptive analysis of the interview responses are all presented in this chapter. A comparative analysis reveals the relative performance and accuracy of CNN and LSTM in detecting software vulnerabilities. These results make this research meaningful and contribute to this field by providing more insights regarding the possibility of conducting security pattern detection using machine learning algorithms.

4.2 Response Rate

Out of the 15 experts who were contacted for the interview, only five agreed to participate in the study. This represents a response rate of 33.3%. The experts who agreed to participate in the virtual interviews were asked to respond to 10 questions related to security pattern detection in software code using machine learning algorithms. Refer to Appendix H for the complete interview questions.

4.3 Descriptive Analysis of Interview Responses

Table 4 below summarizes the expert responses to the interview questions.

Table 4: Interview Responses

Question	Yes	No	Unknown
1) Have you or your organization used Machine Learning algorithms? For what purposes?	5	0	0
2) Please take your time to respond to the following questions: i. How effective can CNN and LSTM machine learning algorithms be in detecting security patterns in software code? ii. How good can the performance of Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) algorithms be in detecting security patterns in software code? iii. How accurate a can CNN and LSTM algorithms be in detecting security patterns in software code? iv. What would the variation in performance and accuracy of CNN and LSTM algorithms be in detecting security patterns in software code?	Check findings section		
3) Can your organization use machine learning algorithms to detect security patterns in software code?	0	3	2

4) Based on your answer to question 3, what's the possibility of machine learning algorithms being effective in detecting security patterns in software code?	**1**	**2**	**2**
5) In your opinion, can your organization correctly implement security patterns?	**1**	**2**	**2**
6) Can your organization perform software assurance and testing? If yes, how frequently? Also, can you check security patterns during this process?	**4**	**0**	**1**
7) Can you compare the effectiveness of CNN vs LSTM algorithms in terms of performance, accuracy, reliability, and other factors?	**CNN 2**	**LSTM 1**	**Unknown 2**
8) Can your organization be a cyber-attack victim or target?	**5**	**0**	**0**
9) How did the attack occur? Were there any software vulnerabilities?	**4**	**0**	**1**
10) Can the implementation of correct software security patterns guarantee resilience against cyberattacks?	**2**	**2**	**1**

Figure 7: Software Assurance and Testing

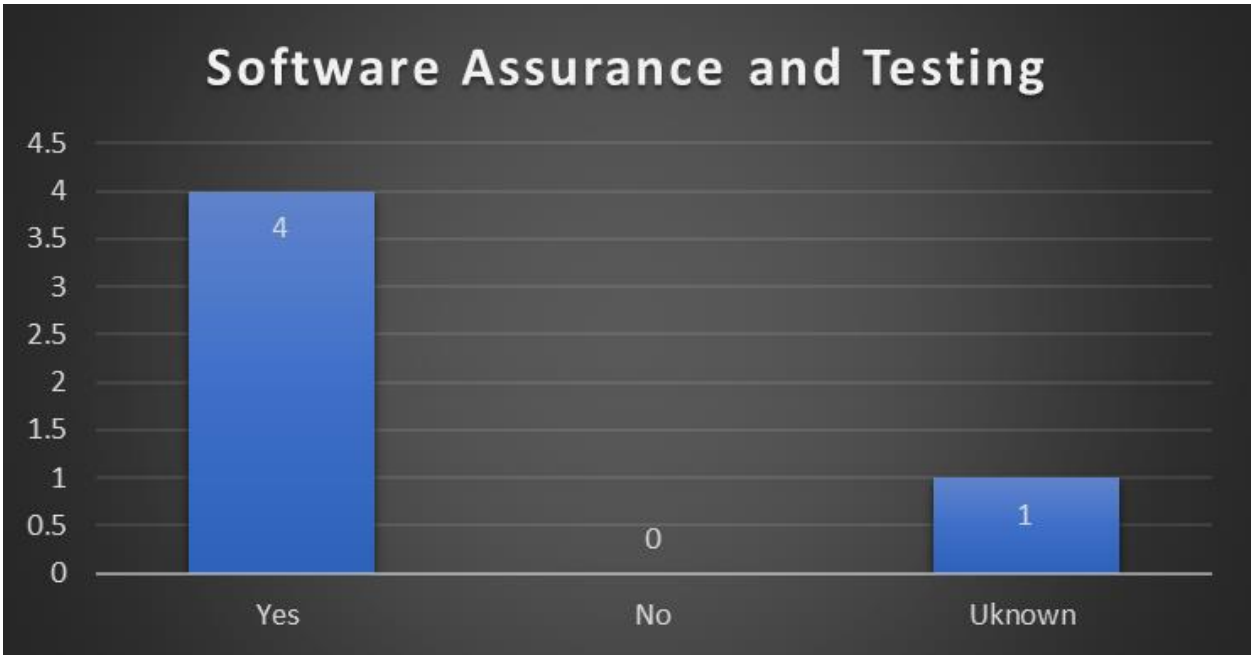


Figure 7 shows how the experts interviewed responded to the software assurance and testing question. Four experts responded that their organizations conduct software assurance and testing to enhance the security of their systems. However, one expert did not know whether their company does software assurance and testing.

Figure 8: Perceived Effectiveness of Machine Learning (ML) Algorithms in Detecting Security Patterns

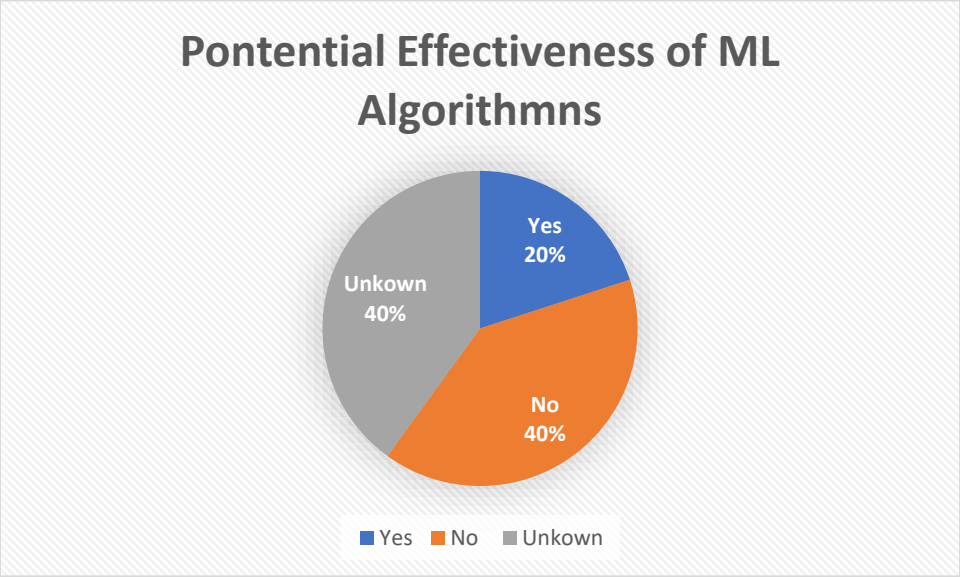


Figure 8 shows the perceived potential of machine learning algorithms in detecting security patterns in software code. One expert affirmed that machine learning algorithms would effectively detect security patterns in software code. Two experts said no, and two others did not know about the potential effectiveness of machine learning algorithms in detecting security patterns.

Figure 9: Correct Implementation of Security Patterns

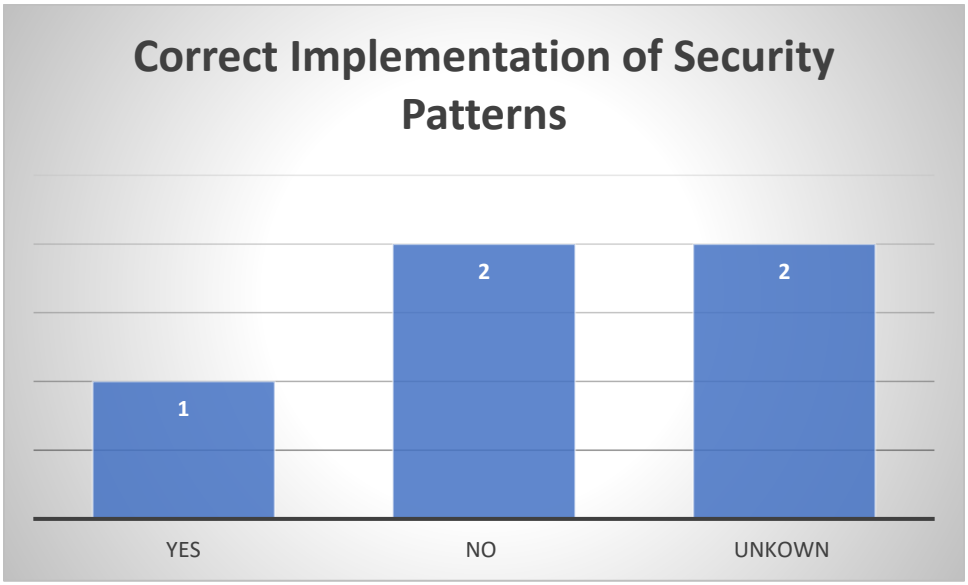


Figure 9 shows the experts' responses regarding the correct implementation of security patterns in organizations. The findings show that one expert agrees that the organization implemented security patterns correctly. Two experts said no, while two other experts did not know whether their organizations correctly implemented security patterns.

Figure 10: Software Vulnerabilities

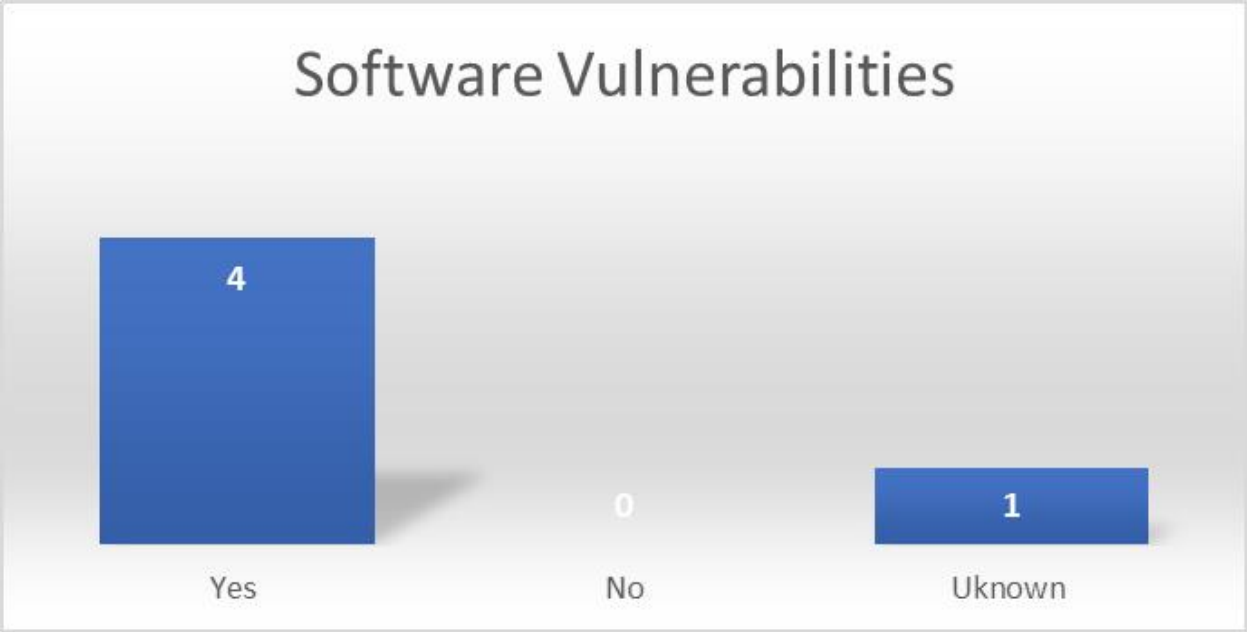


Figure 10 shows how the experts who were interviewed responded to the question about the frequency of cyberattacks. Four of the experts interviewed stated that their organizations are recent victims of cyberattacks. The attacks highlighted by the experts include hacking, ransomware, and phishing. However, one expert was unaware of recent cyberattacks on their company.

Figure 11: Software Assurance and Testing

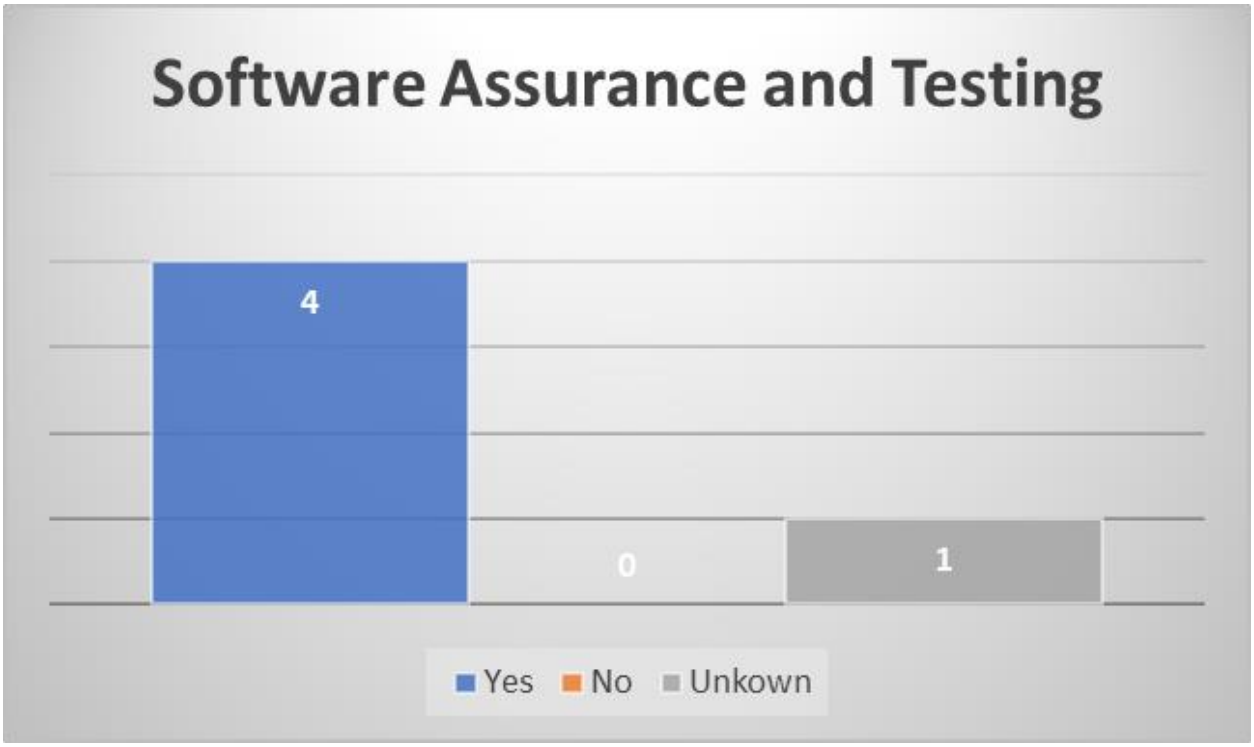


Figure 11 shows that most organizations perform software assurance and testing. Out of the five experts, four stated that their organizations perform software assurance and testing.

Figure 12: Comparison of CNN and LSTM (Performance, Accuracy, Reliability)

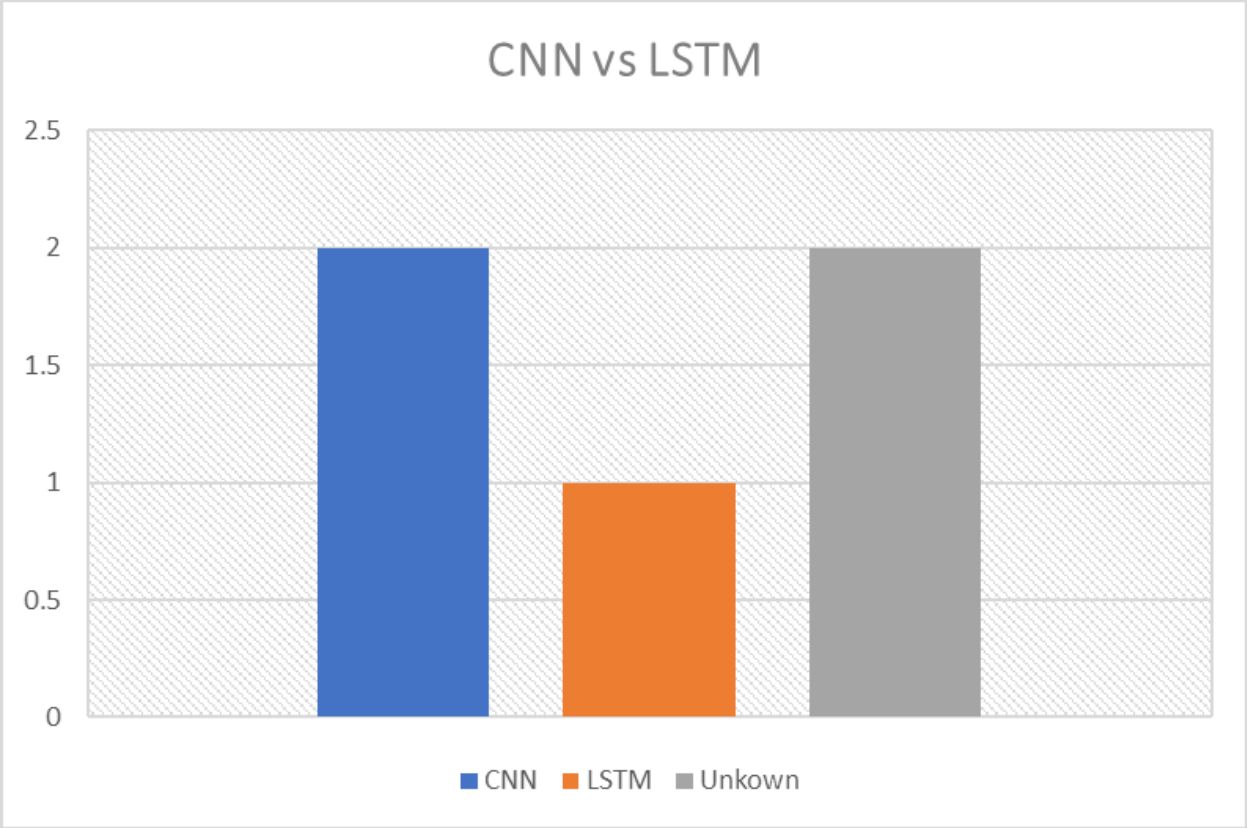


Figure 12 highlights how the experts interviewed compare CNN and LSTM performance, accuracy, and reliability. Two of the five experts indicated that CNN performs better than LSTM. They also viewed CNN to be more accurate and reliable than LSTM. One expert noted that LSTM is more accurate, reliable, and performs better than CNN. Meanwhile, two experts

did not know how LSTM compares with CNN performance, accuracy, and reliability. These results provide more data comparing CNN and LSTM algorithms in vulnerability detection.

4.4 Summary

The findings of this interview play a crucial role in answering the research questions of this study. The results bear on the perceived feasibility of using machine learning algorithms to detect security patterns in software code. The findings also indicate the higher effectiveness of CNN compared to LSTM in terms of performance, accuracy, and reliability. A comparative analysis shows the performance and accuracy of each algorithm as well as the effects of combining the two in detecting software vulnerabilities. A detailed analysis and discussion of the findings of this interview will be presented in the final chapter.

CHAPTER 5: DISCUSSION, CONCLUSION, AND RECOMMENDATIONS

5.1 Introduction

This chapter presents a comprehensive discussion of the findings of this study. The discussion focuses on the results of the comparison of Convolutional Neural Network (CNN) and Long Short-Term Algorithm (LSTM) in detecting software vulnerabilities. The valuable input of the interviewed experts is also discussed in detail and compared with previous studies on this topic. The comprehensive discussion covers the uniqueness of CNN and LSTM algorithms regarding performance, accuracy, and related metrics. The study's limitations, conclusion, and recommendations for future research in security pattern detection are also discussed.

5.2 Discussion of Interview Findings

The responses of the interviewed experts were analyzed using thematic analysis. The descriptive analysis in Chapter 4 summarizes the responses of the experts. The findings of the interviews play a crucial role in answering the study's research questions. The first interview question was about how organizations use machine learning algorithms. Three interviewed experts stated that their organizations are not considering using machine learning algorithms to detect security patterns. Additionally, no expert said they use machine learning algorithms in their organization, while two did not know. This suggests that many organizations do not currently use machine learning algorithms, and even fewer are considering using them to detect security patterns in software code.

Moreover, the interview findings diverged on the perceived potential of using machine learning algorithms for detecting security patterns in software code. Most experts did not know whether machine learning algorithms can be used to detect security patterns. One expert did note that machine learning algorithms can be used effectively for that purpose. However, some experts expressed concerns. Expert #2 stated, "*Yes, the algorithms can be effective. However, more tests need to be done to ascertain the effectiveness of different algorithms*." Expert #4, on the other hand, noted, "*In my opinion, machine learning algorithms are not effective*."

The results of the interview do show that many organizations are vulnerable to cyberattacks. The security loopholes are linked to non-standard security patterns, failure to perform thorough software assurance and testing, and related issues. Some organizations perform a thorough check of security patterns. For example, expert #2 noted, "*Yes, we thoroughly check the security patterns. We aim to ensure that any loopholes and vulnerabilities are identified.*" However, expert #5 stated, "*I cannot tell whether the team goes into such details*." This indicates that some experts are not aware of security pattern detection methods and procedures used by their organizations.

The interviewed experts gave different responses about the performance, accuracy, and reliability metrics of CNN and LSTM. For instance, expert #2 stated, "*We have used CNN and LSTM, but LSTM is better regarding reliability, performance, and accuracy*." Expert #4 also noted, "*I think LSTM performs better than CNN based on my previous experiments. LSTM is also more accurate and reliable compared to CNN algorithm.*" Expert #3, however, noted, "*CNN is more effective compared to LSTM. I think it is better in terms of performance, accuracy, and reliability*."

It is important to note that some experts did not know how the two algorithms compared in terms of their performance, accuracy, and reliability. For example, expert #1 noted, "*I think the best way to compare the effectiveness of CNN and LSTM is to do experiments. In my opinion, the algorithms' performance, accuracy, and reliability do not differ much*."

The variation in performance, accuracy, and reliability of CNN and LSTM in detecting software vulnerabilities is linked to several factors. The algorithm's structure is one of the main factors determining the algorithm's performance. For instance, CNN consists of four layers: convolutional, input, pooling, and fully connected (Alkahtani & Aldhyani, 2021). Each layer has a specific function. For example, the role of the convolutional layer is to explore, size, and filter the training sample. On the other hand, LSTM has three primary gates; input, forget, and output gate. Each gate has a specific role to play, which affects the effectiveness and performance of the algorithm (Alkahtani & Aldhyani, 2021; Gu et al., 2018). The function of the input gate is to store training data in long-term memory.

5.3 CNN vs. LSTM Analysis

This study's findings highlight some of the critical differences in effectiveness, performance, accuracy, and reliability of CNN and LSTM algorithms. This study focused on comparing the effectiveness of the two algorithms in vulnerability detection. However, CNN and LSTM algorithms have been deployed in various fields. Analyzing the effectiveness of CNN and LSTM in other fields provides more comprehensive details on how the algorithms differ. A study by Rasheed et al. (2020) investigated the efficacy of one-dimensional Convolutional Neural Network (1D-CNN) and the Long Short-Term Memory (LSTM) in stock price modeling and prediction. The deep learning-based methods were used to enhance the accuracy of stock

prediction. The findings revealed that both 1D-CNN and LSTM accurately predicted the stock value based on the datasets provided (Rasheed et al., 2020). However, a combination of LSTM with Principal Component Analysis (PCA) led to more accurate predictions.

Gu (2019) conducted a study to investigate the effectiveness and accuracy of the CNN algorithm in security code detection. The study aimed to examine the accuracy of CNN algorithms in using CAPTCHA or security code to differentiate computers and humans. According to that study, artificial intelligence and AI developments have played a vital role in improving security code detection. New machine learning algorithms focus mainly on addressing the challenges of the traditional methods of security code detection. For instance, conventional methods are inefficient, inaccurate, and have fewer learning abilities (Gu, 2019). The study found that the CNN algorithm is more effective in code recognition because it has a strong learning ability and extracts features automatically. The CNN algorithm used in that study achieved an accuracy of 80.5% (Gu, 2019). To achieve more accurate results in code recognition, experts ought to use machine learning algorithms, including CNN.

CNN and LSTM algorithms have also been used recently in the medical field. A study by Islam, Islam, & Asraf (2020) explored the effectiveness of combining CNN and LSTM to help autodetection and diagnosis of COVID-19 using X-ray images. Islam, Islam, & Asraf's (2020) study used CNN for feature extraction and LSTM to classify COVID-19 features. The findings show that CNN is effective for feature extraction, while LSTM is well-suited for classification. Notably, CNN-LSTM perfectly classified the COVID-19 images. However, the accuracy of CNN and LSTM varied. For instance, CNN misclassified four images while LSTM misclassified two images. The researchers proposed to arrive at better and more accurate results from the CNN-LSTM network (Islam, Islam, & Asraf, 2020). The study concluded that combining CNN

and LSTM significantly improved the detection of COVID-19 using automatically extracted features. The conclusion is consistent with Rasheed et al.'s (2020) finding that combining LSTM with PCA led to more accurate predictions. The results are also consistent with the finding of Demir & Taşcı (2021) that combining LSTM with other algorithms improves accuracy. To achieve more precise security pattern detection results, CNN and LSTM algorithms should probably be combined with other models and frameworks.

A hybrid of CNN and LSTM has also been used for anomaly detection systems in software-defined networks (SDN). Elsayed et al. (2021) conducted a study to compare the effectiveness of the CNN-LSTM hybrid in detecting anomalies in SDNs. CNN and LSTM were combined with learning the spatial and temporal features of the traffic input in the network. According to that study, CNN is an effective supervised learning algorithm mainly used in computer vision. On the other hand, LSTM is a special type of recurrent neural network (RNN) used primarily for processing time-series data. The study found that the performance of a standard CNN algorithm is lower than other deep learning models. For instance, the average accuracy of CNN was observed to be 90.79% (Elsayed et al., 2021). However, when regularization methods were introduced, the accuracy of CNN improved to 93.83%. The study also found that LSTM usually performs better than CNN, but worse when CNN is regularized. This means that the performance and accuracy of LSTM are between 90.79% and 93.83%. Figure 13 shows how CNN standard, LSTM, CNN regularized, and CNN-LSTM compared in terms of their precision, recall, and F-1 scores.

Figure 13: Precision, Recall, and F1-Scores

Model	Precision (%)		Recall (%)		F1-Score (%)	
	Normal	Attack	Normal	Attack	Normal	Attack
CNN Standard	76.69	98.86	97.47	88.11	85.84	93.18
LSTM	84.53	98.31	96.02	92.95	89.91	95.55
CNN (*L2 Reg.*)	84.24	**98.56**	**96.62**	92.75	90.00	95.56
CNN-LSTM	**93.18**	97.60	94.04	**97.24**	**93.61**	**97.42**

(Elsayed et al., 2021)

Figure 13 shows that combining CNN and LSTM leads to the best accuracy and performance compared to other algorithms. A hybrid of CNN and the LSTM model seems to enhance the accuracy. However, the accuracy will differ depending on the dataset used.

CNN and LSTM have unique pros and cons in performance, accuracy, reliability, effectiveness, and other parameters. In most of the critically analyzed studies, experts combined CNN and LSTM to leverage their unique benefits and address the challenges of each algorithm. Jang et al. (2020) conducted a study to increase accuracy in text classification by using a hybrid of Bi-LSTM and CNN. In that study, CNN was used to extract sentence features, while LSTM was used to extract contextual information from features obtained by CNN. The results show the accuracy of CNN to be 0.8874 and that of LSTM to be 0.8940. The accuracy of a proposed hybrid of CNN and LSTM was 0.9141 (Jang et al., 2020). The findings of that study are consistent with the results of Elsayed et al.'s (2021) study showing that a combination of CNN and LSTM increases accuracy.

5.4 Discussion of Secondary Findings

RQ1: How effective can CNN and LSTM machine learning algorithms be in detecting security patterns in software code?

One of the experts interviewed noted that LSTM could be more effective than CNN in detecting security patterns in the software code. Most of the experts did not know how these algorithms compared regarding their potential effectiveness. Security pattern detection enables organizations or developers to identify security patterns in a specific software code. Accurate detection also allows the developer and interested parties to comprehend the structure and location of security patterns in a software code. According to Alvi & Zulkernine (2021), security and design pattern detections are not explicitly meant to detect security loopholes in a software code. Many security patterns are needed to enhance security in a system's access control architecture (Alvi & Zulkernine, 2021). Various methods have been formulated to detect security patterns in software code. The accuracy, procedures, models, and frameworks used in these methods vary considerably (Alvi & Zulkernine, 2021). This study's findings play a crucial role in showcasing the perceived effectiveness of particular machine learning algorithms in detecting security patterns.

This study shows that machine learning algorithms are perceived to have the potential to be used for detecting security patterns in software code. However, the accuracy of the machine learning algorithms would depend on various factors. For example, the accuracy of the SPD framework used in this study relies on the quality of the inputs. The reliability of the tools also used significantly impacts the accuracy of security pattern detection. According to Alvi & Zulkernine (2021), one of the best solutions to ensure that the inputs are high quality is using

reliable datasets sources. In addition, the quality of the inputs can be assessed by checking the source code manually. The accuracy of the security pattern detection methods can also be improved by using various methods. For instance, security patterns can be detected using the reverse engineering method. Reverse engineering tools may enable the developer to detect more security patterns that other methods may not reliably detect.

RQ2: How good can the performance of Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) algorithms be in detecting security patterns in software code?

Two experts interviewed stated that LSTM would perform better than CNN in detecting security patterns in software code. A crucial way to ensure that the machine learning algorithms and other security pattern detection methods are effective is to follow the recommended guidelines of security pattern detection. For instance, when extracting syntax characteristics, the focus should be on known vulnerable lines of code. Once the line of code is extracted, the next process is to analyze it (Li et al., 2021). The process is time-consuming; therefore, developers and interested parties should leverage available technologies for security pattern detection.

The study by Li et al. (2020) provides more comprehensive results on automated technologies to detect patterns and vulnerabilities in source code. That study shows that sequential CNN consists of three sequentially connected networks. Each convolutional neural network consists of 128 filters (Li et al., 2020). However, Sequential LSTM consists of two long short-term memory neural networks (LSTM) with 128-dimensional output.

RQ3: How accurate can CNN and LSTM algorithms be in detecting security patterns in software code?

Two experts noted that LSTM would be more accurate than CNN in detecting software code security patterns. However, most experts have not considered using CNN and LSTM to detect security patterns. This study shows that organizations use different enterprise software to automate various business processes and activities. Lee et al. (2020) argue that software in the manufacturing industry has played a crucial role in enhancing efficiency and productivity. The use of technology also plays a vital role in reducing costs and improving the quality of services. Organizations must know that their software is vulnerable to attacks despite these benefits. The findings enable organizations to understand the potential performance and accuracy of CNN and LSTM algorithms if they are used to detect security patterns. Organizations and developers need to understand which algorithm would be more accurate and perform better in detecting security patterns. Properly detecting security patterns, of course, will enable them to build and maintain secure software.

Organizations must capitalize on the available machine learning methods and algorithms that can be used to detect security patterns in software code. Accurate detection of security patterns enables organizations to effectively implement the best solutions to ensure that the software is secure. Lee et al. (2020) recommend using hybrid neural network algorithms combined with CNN and LSTM to enhance pattern detection accuracy. However, it must be stressed that the effectiveness of the combined machine learning algorithms depends on various factors. Furthermore, the appropriate procedures must be followed to ensure that the algorithms

work effectively. For example, before using the LSTM algorithm, two-dimensional data should be transformed into one-dimensional data (Lee et al., 2020). If it becomes possible to use machine learning algorithms such as CNN and LSTM to detect security patterns in software code, organizations must follow the recommended procedures and guidelines to achieve accurate results.

RQ4: What would the variation in performance and accuracy of CNN and LSTM algorithms be in detecting security patterns in software code?

This study also examined the anticipated variation in performance and accuracy of CNN and LSTM in detecting security patterns in software code. The findings show that LSTM would be more accurate and perform better than CNN. This study also explored whether implementing the correct security patterns in software code would guarantee resilience against cyberattacks. The findings show that security patterns play a crucial role in securing a system from software vulnerabilities. Indeed, experts are looking for more effective methods to detect security patterns and vulnerabilities in different software codes. Machine learning algorithms are being tested to allow for the detection of security patterns and to enable organizations to fix software vulnerabilities.

Additionally, using automated technology to detect security patterns and vulnerabilities in software code would enable organizations to be more resilient to cyberattacks. Using automated technologies may play a crucial role in enhancing the effectiveness of security pattern detection. According to Li et al. (2020), one of the best methods to improve the effectiveness of security pattern detection methods is to conduct a comparative experiment with unique techniques. The procedures for detecting security patterns are unique for each method. The

security pattern detection (SPD) method, for instance, involves targeting systems to detect specific security patterns (Alvi & Zulkernine, 2021). The procedure includes initial, intermediate, and final processing to detect security patterns. Semantic analysis and report generation are done during the final processing when using the SPD tool. The generated report shows whether security patterns exist and their specific location.

5.5 Limitations of the Study

The first limitation of our study is the low sample size involved in the interviews. The number of experts selected to participate in the virtual interviews is low. The low sample size limits the ability to generalize the findings of this study. Semi-structured interviews were chosen to enable the selected experts to answer the question more openly (Friis-Liby & Cressy, 2019). However, the limitation is that the quality of the responses relies on the verbal skills of the interviewee. Also, unclear responses to the interview questions can adversely affect the quality of the study findings. The limitation was minimized by eradicating distractions that could interfere with the virtual interviews. The experts were asked to listen keenly to the questions and respond clearly.

Secondly, the secondary findings are derived from literature review and documented experiments that have been conducted to detect software vulnerabilities. Very few secondary literature provided any details on the procedures for or possible outcomes of using machine learning algorithms to detect security patterns. However, the analysis of this study will capitalize on the available literature regardless of its limitations.

Likewise, the potential effectiveness of methods and tools for detecting security patterns depends on the quality of inputs. The inputs used in this study to test the potential effectiveness of machine learning algorithms in security pattern detection were derived from SARD and NVD. Though the databases are reputable, the quality of the available datasets is not guaranteed. Assessing the quality of data and selecting the relevant source code to use was very time-consuming (Alvi & Zulkernine, 2021).

Moreover, this study relied on recently completed studies on machine learning algorithms in vulnerability detection. The use of machine learning algorithms for the novel purpose of security pattern detection is still gaining momentum. Several recent studies provide comprehensive steps, algorithms, and frameworks for detecting security patterns automatically (Lee et al., 2020). A study by Alvi & Zulkernine (2021) explains how the SPD framework can detect security patterns through initial, intermediate, and final processing. Lee et al.'s (2020) study shows how CNN and LSTM algorithms can detect security patterns through power data analysis. Several studies revealed how organizations could leverage the performance and accuracy of CNN and LSTM algorithms to detect security patterns. Most of the reviewed studies used unique methods to detect security patterns, including machine learning, deep learning, and SPD framework.

Furthermore, due to time and resource constraints, previous experiments' findings were analyzed to enrich the primary data collected through the interviews. Performing experiments to compare various machine learning algorithms' effectiveness, performance, and accuracy is expensive. These experiments also have reliability and validity issues. The reliability and validity issues could impact the quality and accuracy of the findings and conclusions of this study. Recommendations on how future research can effectively address such limitations are provided.

For example, using a larger sample in future studies will enhance the reliability and generalizability of the results.

Finally, studying the potential use of machine learning algorithms for detecting security patterns in specific software code is time-consuming and requires many resources. Our study proceeds under a tight schedule using the available resources. Our objective is to deliver accurate results within the existing scope, schedule, and budget.

5.6 Conclusion

The study explored the potential use of machine learning in security pattern detection in software code. The findings showed the feasibility of using machine learning algorithms in detecting security patterns. Machine learning algorithms – namely, convolutional neural network (CNN) and long short-term memory (LSTM) – combined with appropriate tools and methods, can probably be used effectively to detect security patterns in a software code. The findings also highlight the significance of implementing security patterns correctly and fixing vulnerabilities that cybercriminals can exploit. Though the findings are promising, more in-depth research is needed to precisely assess the effectiveness of machine learning algorithms and methods in detecting security patterns in software code.

5.7 Recommendations for Future Research

The primary objective of this study was to contribute meaningfully to its field of study: machine learning algorithms and their potential use in security pattern detection. The study's

findings provide fresh insights on how machine learning algorithms can be employed to detect security patterns. Future research needs to be more in-depth and examine various machine learning algorithms, frameworks, deep learning algorithms, and more advanced tools (Li et al., 2021). Focusing on advanced machine learning algorithms, comprehensive datasets, and automated technologies will enhance the quality of the results of future studies.

Future research in this field also needs to address the study's limitations. It is highly recommended for future research to address the threats of validity encountered in the study. According to the study by Alvi & Zulkernine (2021), the process of detecting security patterns is affected by the variations in the security pattern matrix and size. Future research needs to address this challenge by using more innovative methods to detect security patterns.

Furthermore, future research should focus more on ensuring that the findings improve specific areas in this field. Presently, organizations use different technologies to enhance productivity, performance, and efficiency. Organizations also use software to automate key business processes, leading to a reduction of costs and increased profitability. However, most of the software they use is vulnerable to security attacks. Future studies need to focus on testing the effectiveness of using machine learning algorithms to detect security patterns in a specific software (Lee et al., 2020). The sample size of the dataset used for experiments should also be improved. Using a larger sample size will lead to more accurate and detailed findings.

Finally, considering the complexity of machine learning and pattern detection experiments, researchers should have better access to advanced machine learning tools, software, datasets, training sets, and other such materials to carry out various experiments. Future research should also investigate the likely performance, effectiveness, accuracy, and reliability of various

machine learning algorithms other than CNN and LSTM in detecting security patterns in software code.

References

Apruzzese, G., Colajanni, M., Ferretti, L., Guido, A., & Marchetti, M. (2018). On the effectiveness of machine and deep learning for cybersecurity. In *2018 10th International Conference on Cyber-Conflict (CyCon)* (pp. 371-390). IEEE.

Alvi, A. K., & Zulkernine, M. (2021). A security pattern detection framework for building more secure software. *Journal of Systems and Software*, *171*, 110838.

Al-Garadi, M. A., Mohamed, A., Al-Ali, A. K., Du, X., Ali, I., & Guizani, M. (2020). A survey of machine and deep learning methods for internet of things (IoT) security. *IEEE Communications Surveys & Tutorials*, *22*(3), 1646-1685.

Al-Khater, W. A., Al-Maadeed, S., Ahmed, A. A., Sadiq, A. S., & Khan, M. K. (2020). Comprehensive review of cybercrime detection techniques. *IEEE Access*, *8*, 137293-137311.

Alkahtani, H., & Aldhyani, T. H. (2021). Botnet attack detection by using CNN-LSTM model for internet of things applications. *Security and Communication Networks*, *2021*.

Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., ... & Zimmermann, T. (2019). Software engineering for machine learning: A case study. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)* (pp. 291-300). IEEE.

Arpteg, A., Brinne, B., Crnkovic-Friis, L., & Bosch, J. (2018). Software engineering challenges of deep learning. In *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (pp. 50-59). IEEE.

Arslan, R. S. (2021). AndroAnalyzer: android malicious software detection based on deep learning. *PeerJ Computer Science*, *7*, e533.

Berman, D. S., Buczak, A. L., Chavis, J. S., & Corbett, C. L. (2019). A survey of deep learning methods for cyber security. *Information*, *10*(4), 122.

Chernis, B., & Verma, R. (2018). Machine learning methods for software vulnerability detection. In *Proceedings of the Fourth ACM International Workshop on Security and Privacy Analytics* (pp. 31-39).

Choi, Y. H., Liu, P., Shang, Z., Wang, H., Wang, Z., Zhang, L., ... & Zou, Q. (2020). Using deep learning to solve computer security challenges: A survey. *Cybersecurity*, *3*(1), 1-32.

Demir, F., & Taşcı, B. (2021). An effective and robust approach based on R-CNN+ LSTM model and NCAR feature selection for ophthalmological disease detection from fundus images. *Journal of Personalized Medicine*, *11*(12), 1276.

Elsayed, M. S., Le-Khac, N. A., Jahromi, H. Z., & Jurcut, A. D. (2021). A hybrid CNN-LSTM based approach for anomaly detection systems in SDNs. In *The 16th International Conference on Availability, Reliability and Security (ARES 2021)*.

Friis-Liby, P., & Cressy, J. (2019). Organizational aspects to consider in order to implement machine learning and create business value: A qualitative study on a technology industry leader. *University of Boras*.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., ... & Chen, T. (2018). Recent advances in convolutional neural networks. *Pattern Recognition*, *77*, 354-377.

Gu, J. (2019). The application of convolutional neural network in security code recognition. In *Journal of Physics: Conference Series* (Vol. 1187, No. 4, p. 042064). IOP Publishing.

Guo, J., Wang, Z., Li, H., & Xue, Y. (2021). Detecting Vulnerability in Source Code Using CNN and LSTM Network. *School of Computer Science and Engineering, Xi'an Technological University.*

Hanif, H., Nasir, M. H. N. M., Ab Razak, M. F., Firdaus, A., & Anuar, N. B. (2021). The rise of software vulnerability: Taxonomy of software vulnerabilities detection and machine learning approaches. *Journal of Network and Computer Applications*, 103009.

Harrington, D. (2020). Data security: Importance, types, and solutions – Varonis. Retrieved from https://www.varonis.com/blog/data-security/

Harer, J. A., Kim, L. Y., Russell, R. L., Ozdemir, O., Kosta, L. R., Rangamani, A., ... & Lazovich, T. (2018). Automated software vulnerability detection with machine learning. *arXiv preprint arXiv:1803.04497*.

Hwang, R. H., Peng, M. C., Nguyen, V. L., & Chang, Y. L. (2019). An LSTM-based deep learning approach for classifying malicious traffic at the packet level. *Applied Sciences*, *9*(16), 3414.

Islam, M. Z., Islam, M. M., & Asraf, A. (2020). A combined deep CNN-LSTM network for the detection of novel coronavirus (COVID-19) using X-ray images. *Informatics in Medicine Unlocked*, *20*, 100412.

Jang, B., Kim, M., Harerimana, G., Kang, S. U., & Kim, J. W. (2020). Bi-LSTM model to increase accuracy in text classification: Combining Word2vec CNN and attention mechanism. *Applied Sciences*, *10*(17), 5841.

Khalid, M. N., Farooq, H., Iqbal, M., Alam, M. T., & Rasheed, K. (2018). Predicting web vulnerabilities in web applications based on machine learning. In *International Conference on Intelligent Technologies and Applications* (pp. 473-484). Springer, Singapore.

Kumar, A., & Lim, T. J. (2019). EDIMA: Early detection of IoT malware network activity using machine learning techniques. In *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)* (pp. 289-294). IEEE.

Krouwel, M., Jolly, K., & Greenfield, S. (2019). Comparing Skype (video calling) and in-person qualitative interview modes in a study of people with irritable bowel syndrome–an exploratory comparative analysis. *BMC medical research methodology*, *19*(1), 1-9.

Lee, J. H., Kang, J., Shim, W., Chung, H. S., & Sung, T. E. (2020). Pattern detection model using a deep learning algorithm for power data analysis in abnormal conditions. *Electronics*, *9*(7), 1140.

Li, Z., Zou, D., Xu, S., Jin, H., Zhu, Y., & Chen, Z. (2021). SySeVR: A framework for using deep learning to detect software vulnerabilities. *IEEE Transactions on Dependable and Secure Computing*.

Li, Z., Zou, D., Xu, S., Ou, X., Jin, H., Wang, S., ... & Zhong, Y. (2018). VulDeePecker: A deep learning-based system for vulnerability detection. *arXiv preprint arXiv:1801.01681*.

Li, X., Wang, L., Xin, Y., Yang, Y., & Chen, Y. (2020). Automated vulnerability detection in source code using minimum intermediate representation learning. *Applied Sciences*, *10*(5), 1692.

Musser, M., & Garriott, A. (2021). Machine learning and cybersecurity - Hype and reality. *Center for Security and Emerging Technology.*

Nazar, N., & Aleti, A. (2020). Feature-based software design pattern detection. *arXiv preprint arXiv:2012.01708*.

Nassif, A. B., Talib, M. A., Nassir, Q., Albadani, H., & Albab, F. D. (2021). Machine learning for cloud security: A systematic review. *IEEE Access*.

Ndichu, S., Kim, S., Ozawa, S., Misu, T., & Makishima, K. (2019). A machine learning approach to detection of JavaScript-based attacks using AST features and paragraph vectors. *Applied Soft Computing*, *84*, 105721.

Oseni, A., Moustafa, N., Janicke, H., Liu, P., Tari, Z., & Vasilakos, A. (2021). Security and privacy for Artificial Intelligence: Opportunities and challenges. *arXiv preprint arXiv:2102.04661*.

Park, D., Bahrudin, F., & Han, J. (2020). Circular reasoning for the evolution of research through a strategic construction of research methodologies. *International Journal of Quantitative and Qualitative Research Methods*.

Rasheed, J., Jamil, A., Hameed, A. A., Ilyas, M., Özyavaş, A., & Ajlouni, N. (2020). Improving stock prediction accuracy using CNN and LSTM. In *2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI)* (pp. 1-5). IEEE.

Russell, R., Kim, L., Hamilton, L., Lazovich, T., Harer, J., Ozdemir, O., ... & McConley, M. (2018). Automated vulnerability detection in source code using deep representation learning. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 757-762). IEEE.

Rutberg, S., & Bouikidis, C. D. (2018). Focusing on the fundamentals: A simplistic differentiation between qualitative and quantitative research. *Nephrology Nursing Journal*, *45*(2), 209-213.

Sagar, R., Jhaveri, R., & Borrego, C. (2020). Applications in security and evasions in machine learning: A survey. *Electronics*, *9*(1), 97.

Saunders, M., Lewis, P., & Thornhill, A. (2009). *Research methods for business students*. Pearson Education.

Shaukat, S. K., & Ribeiro, V. J. (2018). RansomWall: A layered defense system against cryptographic ransomware attacks using machine learning. In *2018 10th International Conference on Communication Systems & Networks (COMSNETS)* (pp. 356-363). IEEE.

Shaukat, K., Luo, S., Varadharajan, V., Hameed, I. A., & Xu, M. (2020). A survey on machine learning techniques for cyber security in the last decade. *IEEE Access*, *8*, 222310-222354.

Sarker, I. H. (2021). Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*, 2(3), 1-21.

Silva-Rodríguez, V., Nava-Muñoz, S. E., Castro, L. A., Martínez-Pérez, F. E., Pérez-González, H. G., & Torres-Reyes, F. (2019). Machine learning methods for inferring interaction design patterns from textual requirements. In *Multidisciplinary Digital Publishing Institute Proceedings* (Vol. 31, No. 1, p. 26).

Ullah, F., Naeem, H., Jabbar, S., Khalid, S., Latif, M. A., Al-Turjman, F., & Mostarda, L. (2019). Cyber security threats detection in internet of things using deep learning approach. *IEEE Access*, *7*, 124379-124389.

VanHilst, M., & Fernandez, E. B. (2007). Reverse engineering to detect security patterns in code. In *Proceedings of the International Workshop on Software Patterns and Quality. Information Processing Society of Japan* (pp. 25-30).

Wang, S., Wang, P., & Wu, D. (2017). Semantics-aware machine learning for function recognition in binary code. In *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)* (pp. 388-398). IEEE.

Wu, Y., Wei, D., & Feng, J. (2020). Network attacks detection methods based on deep learning techniques: A survey. *Security and Communication Networks*, *2020*.

Xue, H., Sun, S., Venkataramani, G., & Lan, T. (2019). Machine learning-based analysis of program binaries: A comprehensive study. *IEEE Access*, *7*, 65889-65912.

Ziems, N., & Wu, S. (2021). Security vulnerability detection using deep learning natural language processing. *arXiv preprint arXiv:2105.02388*.

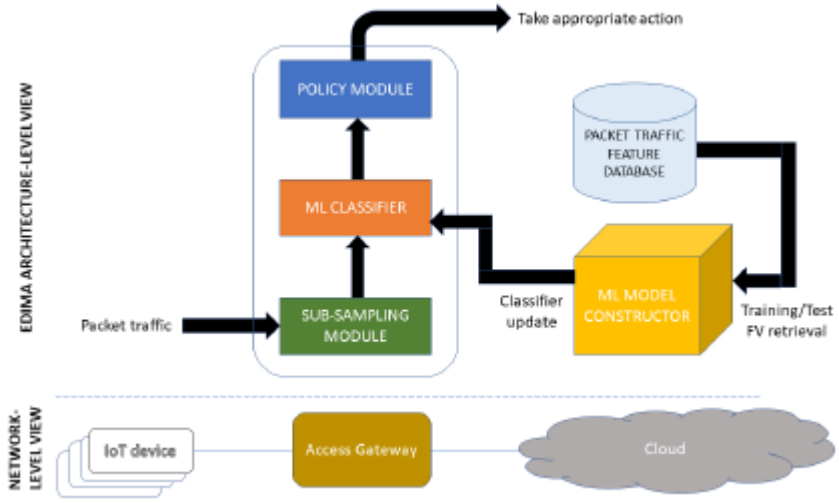
Appendices

Appendix A: Results of VulDeePecker Effectiveness Tests

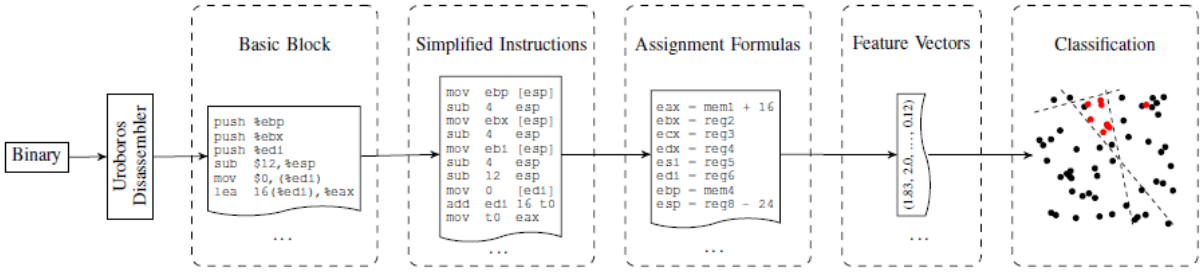
System	Dataset	FPR (%)	FNR (%)	TPR (%)	P (%)	F1 (%)
VulDeePecker vs. Other pattern-based vulnerability detection systems						
Flawfinder	BE-SEL	44.7	69.0	31.0	25.0	27.7
RATS	BE-SEL	42.2	78.9	21.1	19.4	20.2
Checkmarx	BE-SEL	43.1	41.1	58.9	39.6	47.3
VulDeePecker	BE-SEL	**5.7**	**7.0**	93.0	88.1	90.5
VulDeePecker vs. Code similarity-based vulnerability detection systems						
VUDDY	BE-SEL-NVD	0	95.1	4.9	100	9.3
VulPecker	BE-SEL-NVD	1.9	89.8	10.2	84.3	18.2
VulDeePecker	BE-SEL-NVD	**22.9**	**16.9**	83.1	78.6	80.8
VUDDY	BE-SEL-SARD	N/C	N/C	N/C	N/C	N/C
VulPecker	BE-SEL-SARD	N/C	N/C	N/C	N/C	N/C
VulDeePecker	BE-SEL-SARD	**3.4**	**5.1**	94.9	92.0	93.4

(Li et al., 2018)

Appendix B: EDIMA Architecture
(Kumar & Lim, 2019)



Appendix C: The Workflow of FID
(Wang, Wang, & Wu, 2017)



Appendix D: Comparison of Different Representation Learning Models

Table 3. Comparison of different representation learning models.

Model	FPR (%)	FNR (%)	P (%)	R (%)	F1 (%)
Sequential CNN	6.3	22.0	85.9	68.0	75.9
Sequential LSTM	17.6	30.8	69.0	69.2	69.1
BiLSTM	29.4	7.3	64.1	92.3	75.7
CNN+BiLSTM	13.1	10.4	80.6	89.6	84.9
CNN +BiLSTM+Attention	5.5	33.9	87.1	66.1	75.1
Concatenated CNN	1.8	15.0	94.4	85.0	89.5

(Li et al., 2020)

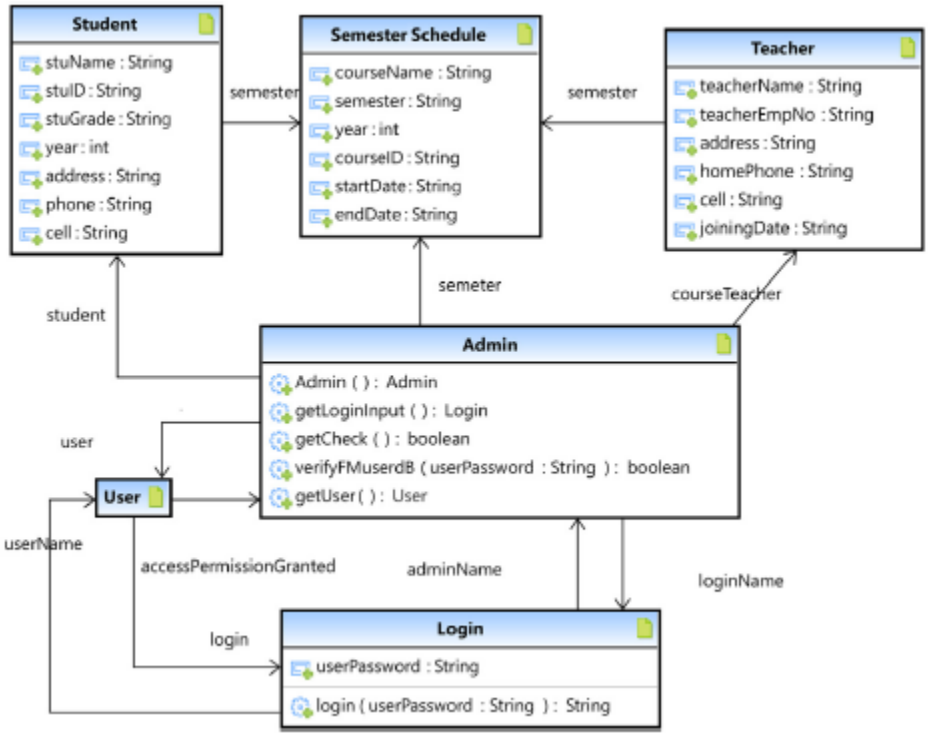
Appendix E: Comparison of Different Classification Algorithms

(Li et al., 2020)

Table 4. Comparison of different classification algorithms.

Algorithm	FPR (%)	FNR (%)	P (%)	R (%)	F1 (%)
LR	2.4	12.0	93.0	88.0	90.5
NB	18.4	13.1	62.6	86.9	72.8
SVM	1.2	10.9	96.4	89.1	92.6
MLP	2.8	12.6	91.8	87.4	89.5
GBDT	1.3	11.3	96.0	88.7	92.2
RF	1.5	9.6	95.7	90.4	93.0

Appendix F: The SSS Target System Class Diagram
(Alvi & Zulkernine, 2021)



Appendix G: Matrix Representation of the SSS Target System

(b)

	A	B	C	D	E	F	Notation	Classes
A	**0**	**1**	**1**	0	0	0	A	User
B	**1**	**0**	**1**	0	0	0	B	Login
C	**1**	**1**	**0**	1	1	1	C	Admin
D	0	0	0	0	0	0	D	Semester Schedule
E	0	0	0	1	0	0	E	Student
F	0	0	0	1	0	0	F	Teacher

(Alvi & Zulkernine, 2021)

Appendix H: Interview Questions

Security Pattern Detection in Software Code Using Machine Learning Algorithms

Interview

Introduction

Dear Sir / Madam

You are invited to participate in an interview on Security Pattern Detection in Software Code using Machine Learning Algorithms.

The interview targets experts in security pattern detection and machine learning.

If it applies to you, I would be grateful if you agreed to participate.

The interview won't take more than 30 minutes.

The interview responses are anonymous, and personal information (name and organization) will not be included in the final report.

By participating in this interview, you agree:

- Are aged above 18 years
- Understand the purpose of the study
- Give consent for your responses to this research
- Understand security pattern detection or Machine learning algorithms

1. Would you please confirm that you have read and understood the text above?
   ☐ Yes
   ☐ No
2. I understand that participation in the interview is voluntary.
   ☐ Yes
   ☐ No
3. I understand that the interview complies with research ethics requirements.
   ☐ Yes
   ☐ No
4. I have been given a copy of this consent form.
   ☐ Yes
   ☐ No

Interview Questions

Interviewee:

1) Have you or your organization used Machine learning algorithms? For what purposes?

2) Please take your time to respond to the following questions:

    i. How effective can CNN and LSTM machine learning algorithms be in detecting security patterns in software code?
    ii. How good can the performance of Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) algorithms be in detecting security patterns in software code?
    iii. How accurate can CNN and LSTM algorithms be in detecting security patterns in software code?
    iv. What would the variation in performance and accuracy of CNN and LSTM algorithms be in detecting security patterns in software code?

3) Can your organization use machine learning algorithms to detect security patterns in software code?

4) Based on your answer to question 3, what's the possibility of machine learning algorithms being effective in detecting security patterns in software code?

5) In your opinion, can your organization correctly implement security patterns?

6) Can your organization perform software assurance and testing? If yes, how frequent? Also, can you check security patterns during this process?

7) Can you compare the effectiveness of CNN vs LSTM algorithms in terms of performance, accuracy, reliability, and other factors?

8) Can your organization be a cyber-attack victim or target?

9) How did the attack occur? Were there any software vulnerabilities?

10) Can the implementation of correct software security patterns guarantee resilience against cyberattacks?

11) Please feel free to provide additional comments about the use of machine learning algorithms to detect security patterns in software code.

Thank you for taking the time to participate in the interview.