VULNERABILITY DETECTION PROJECT

GAYATRI MALLADI

INTRODUCTION

Vulnerabilities or code smells are the existing flaws or weaknesses if neglected might lead to bugs threatening the architecture of the web applications.

According to OWASP Top Ten-2021, Insecure Design is one of the most serious web app security vulnerabilities which lead to data breaches,denial of service user and system enumeration.

Insecure design due to poor architecture,weak encryption,inadequate access controls and flawed application design was found 79% in majority of web applications.

The proliferation of zero-day threats(ZDTs) to companies' networks has been immensely costly and requires novel methods to scan traffic and behavior at massive scale.

According to OWASP Top Ten-2021, Insecure Design is one of the most serious web app security vulnerabilities which lead to data breaches,denial of service user and system enumeration.

ABSTRACT

To improve scalability and accuracy of prediction Image processing is considered as best approach when compared to graph-based,text-based and line-based methods.

We purpose a framework which is divided into two phases.

The first phase involves an approach to convert source-code into a representation of architecture of source code and populate dataset of source-codes to generate similar source-code snippets using Conditional Generative Adversarial Network(CGAN) for vulnerability detection to uplift the limitation of lack of data.

In the second phase the results generated from the first phase could be utilized for automatic classification from the feature extraction of the programming images.

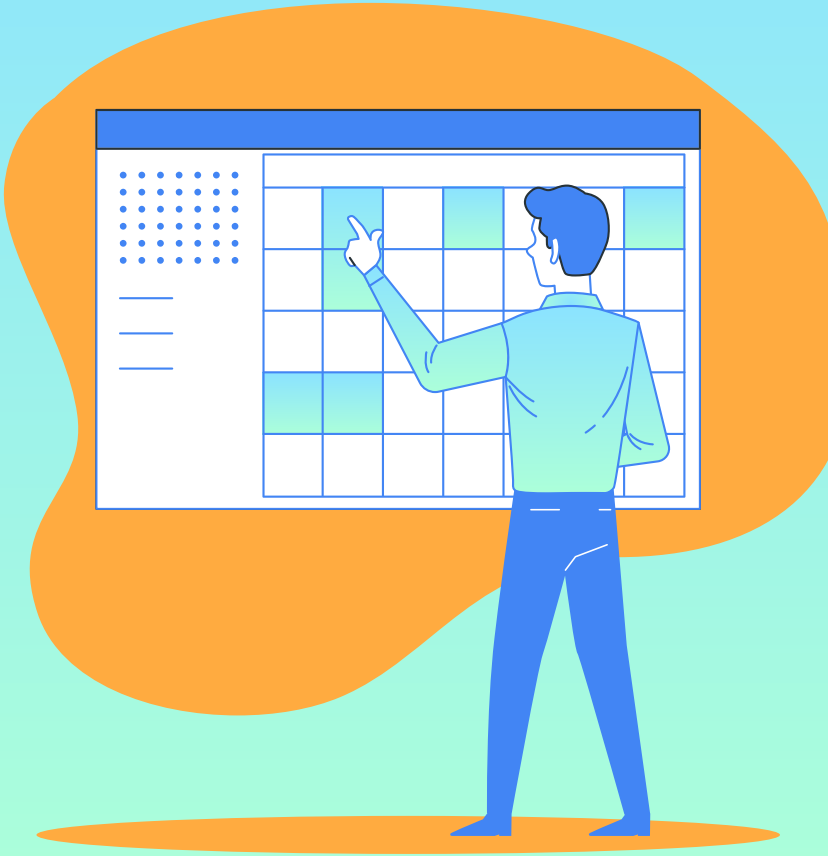
MOTIVATION

To reduce insecure design flaws in web application that leads to data breaches.

To reduce zero-day threats(ZDTs) to protect the software from unknown exploits and malicious enablers.

Integrate deep learning approaches to solve cybersecurity.

INSECURE DESIGN



What is Insecure Design?

Lack of security controls being integrated into the application architecture throughout the development cycle.

TYPES OF INSECURE DESIGN:

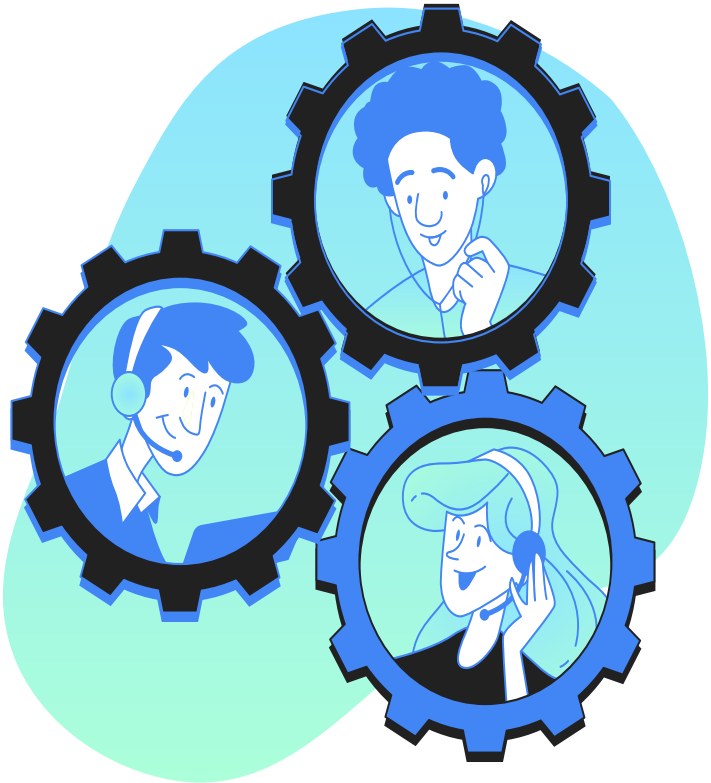
1.Unprotected Storage Of Credentials:(CWE-256)

Plaintext storage of passwords without encryption may compromise system access through unauthorised access to user accounts which leads to exploit legitimate accounts.

2.External control of path to file name:

CSE-73 that allows user input to control or influence paths or filenames that are used in filesystem operations in the following ways:

i. An attacker can specify a path used in an operation on the filesystem.
ii. By specifying the resource, attacker gains a capability by overwriting the specified file or run with a configuration



What is Insecure Design?

3.Storing passwords in a recoverable format:(CSE-257)

Recoverable encryption for passwords has no significant benefit over plaintext passwords since they are subject not only to reuse by malicious attackers through brute force.

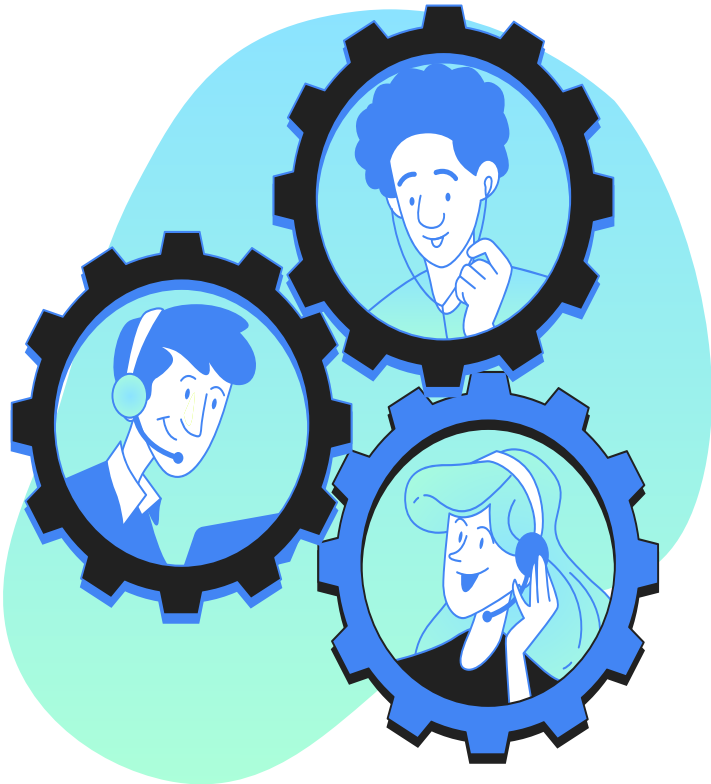
Solution:

i.Username and password information should not be included in a configuration file.

ii.One way hashing algorithm should be used instead of compression algorithm so that attacker can never recover compressed passwords stored in the database.

4.Unrestricted upload of files:(CWE-434)

Occurs when an application fails to validate file extension, eventually allowing an attacker to transfer or upload malicious files.



What is Insecure Design?

5.Inconsistent interpretation of HTTP request:(CWE-444)

When an intermediate agent interprets client requests inconsistently, the flaw allows attackers to smuggle malformed requests to one entity without the other being aware.

Solution:

i. Web Server Firewall must be installed in the system.

6.USE OF A BROWSER CACHE CONTAINING SENSITIVE INFORMATION:(CWE-525)

Client side caching to improve performance by pooling previously processed information locally reduces the time required to initialise process, access information requested by users.

Solution:

i.Use a restrictive caching policy for forms and web pages that contain sensitive information.

ii.Consider using encryption in the cache.

What is Insecure Design?

.INSUFFICIENT COMPARTMENTALIZATION:(CSE-653)

The product does not properly compartmentalize or isolate functionality, processes, or resources that require different privilege levels, rights, or permissions.

Example: Single-sign-on is intended to make it easier for users to access multiple resources without having to authenticate each time.It leads to compromise of user's credentials and attacker can gain access to all resources easily.

8.EXTERNAL CONTROL OF CRITICAL STATE DATA:(CWE-642)

Storing critical state information about software or user sessions in a location that available to attack. Locations where state information can be stored may be a web form field, input argument, database record or a publicly accessible cookie.

Solution:

i.Use Message Authentication Code(MAC) algorithm.

ii.Store state information and sensitive data on the server side only and if information needs to be stored on the client side then do not do it without encryption and integrity checking.

WHY DATA SCIENCE?

1. Data Generation: To uplift the limitations of lack of data in cybersecurity.Using Data Science approaches we can generate synthetic data.
2. Pattern recognition:Using data science approaches we can recognize patterns within vast amount of data.
3. Behavioural Analysis: By analyzing historical data and patterns, predictive models can identify weak points in a system, enabling proactive measures to prevent exploitation.
4. Automation and scale:Data science techniques can automate vulnerability detection, allowing for the quick and efficient analysis of massive amounts of data, something not easily achievable through manual methods.
5. Feature Extraction:Identifying relevant features within the data that contribute to vulnerabilities in creating more robust model.

LITERATURE SURVEY

LITERATURE SURVEY-1

The Effectiveness of Zero-Day Attacks Data Samples Generated via GANs on Deep Learning Classifiers

Problem Statement: Digitization of new software and hardware services has lead to increase in ZDTs that can be exploited by malicious users, posing threats for end-users.In the field of cybersecurity,lack of data especially regarding ZDTs has lead to increase in zero-day attacks.To lift the limitations this paper proposes design and employment of GANs for generating a new and larger synthetic data set of ZDTs,then used to train and evaluate a NN classifier for ZDTs with and without transfer learning features.[2]

Methodology:

i.Feature selection using Univariate Feature selection to extract strongest correlation between the independent features and the selected output feature;Extra Trees classifier to compare the results produced previously and Correlation Matrix with heatmap and correlation scale.[2]

LITERATURE SURVEY-1

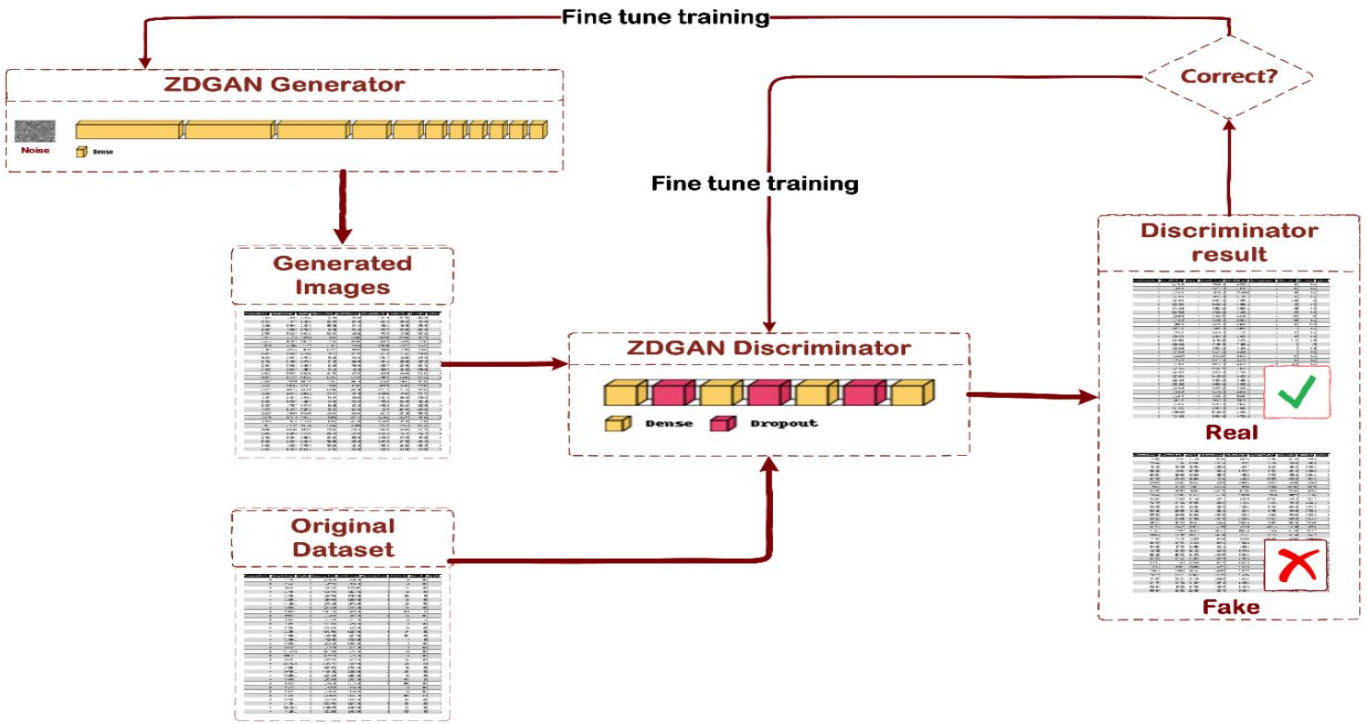


Figure 4. ZDGAN implementation.

LITERATURE SURVEY-1

The Effectiveness of Zero-Day Attacks Data Samples Generated via GANs on Deep Learning Classifiers

Methodology:

ii.The second stage involves synthetic data generation using ZDGAN as shown in the architecture diagram.

iii.The training process was configured for 5000 epochs and a batch training of predefined size is performed on both the generator and discriminator.

iv.The discriminator accepts as input the data samples from the dataset and maximizes probability of assigning correct labels to both original and generated images.

v.After completion of ZDGAN DMT is performed to evaluate the similarities.Then, PCA is performed for linear mapping data features of both datasets.

vii.The last stage involves the classification using NN and comparison of the results produced by training three different zero day datasets:

(i)Original dataset from kaggle (ii) The generated samples dataset (iii) A combination of the two aforementioned datasets.

LITERATURE SURVEY-1

The Effectiveness of Zero-Day Attacks Data Samples Generated via GANs on Deep Learning Classifiers

Limitations:

i.Due to random noise introduced to generate random samples the generator even after getting stabilized still some loss spikes are observed until the end of training process that lead to increase in discriminator loss since that was no longer easy to correctly classify synthetic data.[2]

ii.The overall performance of the original dataset was far better than the generated dataset.

iii.The model only detects possible ZDTs rather than predicting.

iv.The DL model designed is not the optimal one and the study only focuses on effectiveness of data generation via GANs than on the performance of DL model.

Pros:

i. Solves the issue of data availability.

ii.Successfully generated new data and performed way better when trained the combined datasets.

LITERATURE SURVEY-2

SoCodeCNN:Program Source Code for Visual CNN Classification Using Vision Methodology

Problem Statement: The paper explores on automated feature extraction from program source-code by automatically converting source-code to visual images to overcome the conventional methods of feature extraction.[1]

Methodology: The study includes pre-processing of source-code using IR generator to generate optimized code; Code cleanser to get rid of redundant part of IR code and processing source-code IR is done wheresource- code is converted into images using Pixelator[1].

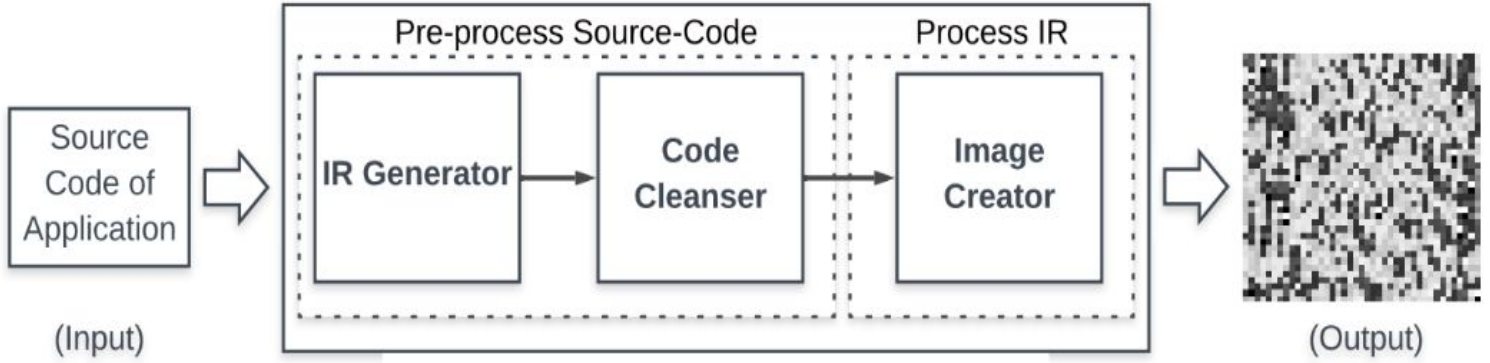


FIGURE 3. Block diagram of *SoCodeCNN*.

LITERATURE SURVEY-2

SoCodeCNN:Program Source Code for Visual CNN Classification Using Vision Methodology

Algorithms: D-CNN-VGG19 is used to convert image into different convolutional layers and pool features and fin-tune features for prediction.[1]

Pixelator Algorithm: defines how code elements are mapped to pixels.

Drawbacks:

i.Program source code must be manually selected and for new applications we need to perform offline training.

Pros:

i.Overcomes the conventional method of feature extraction.

LITERATURE SURVEY-3

Literature review on vulnerability detection using NLP technology

Problem Statement:To overcome the conventional methods of vulnerability detection automated analysis of source code is need using NLP technologies such as CodeBert and DL in NLP.[3]

Methodology: Models like CNN,RNN,LSTM and Embeddings such as BERT,GloVe are used for feature extraction.The source code is segmented and features are extracted in the source code using AST and graph splitting is performed.[3]

The second stage involves model training where input is the output of the first stage using GNN and non-GNN model.Then CodeBert training model is implemented for code generation tasks;code-code,text-code,code-text and text-text using CodeXGLUE.[3]

Drawbacks:

i.Limited to known vulnerabilities as it is based on patterns it has learned from training data.

ii.False positives may occur

iii.Code obstruction to hide security flaws and allow attackers to make their code benign.

Pros:

i.Continuous learning and more powerful feature extraction.

LITERATURE SURVEY-3

Literature review on vulnerability detection using NLP technology

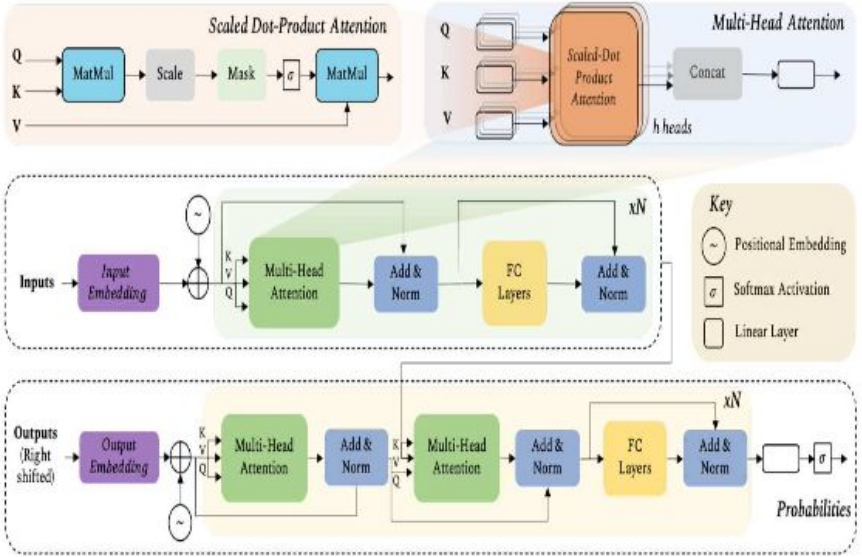


Fig. 5. Architecture of the Transformer Model [63]

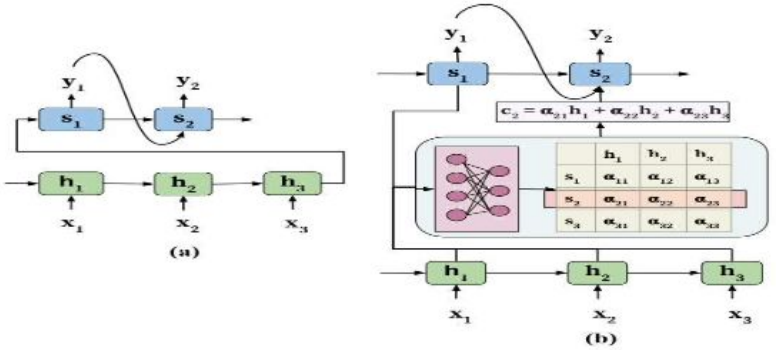


Fig. 3. A comparison between the traditional encoder-decoder architecture (left) and the attention-based architecture (right) [61]

LITERATURE SURVEY-3

Literature review on vulnerability detection using NLP technology

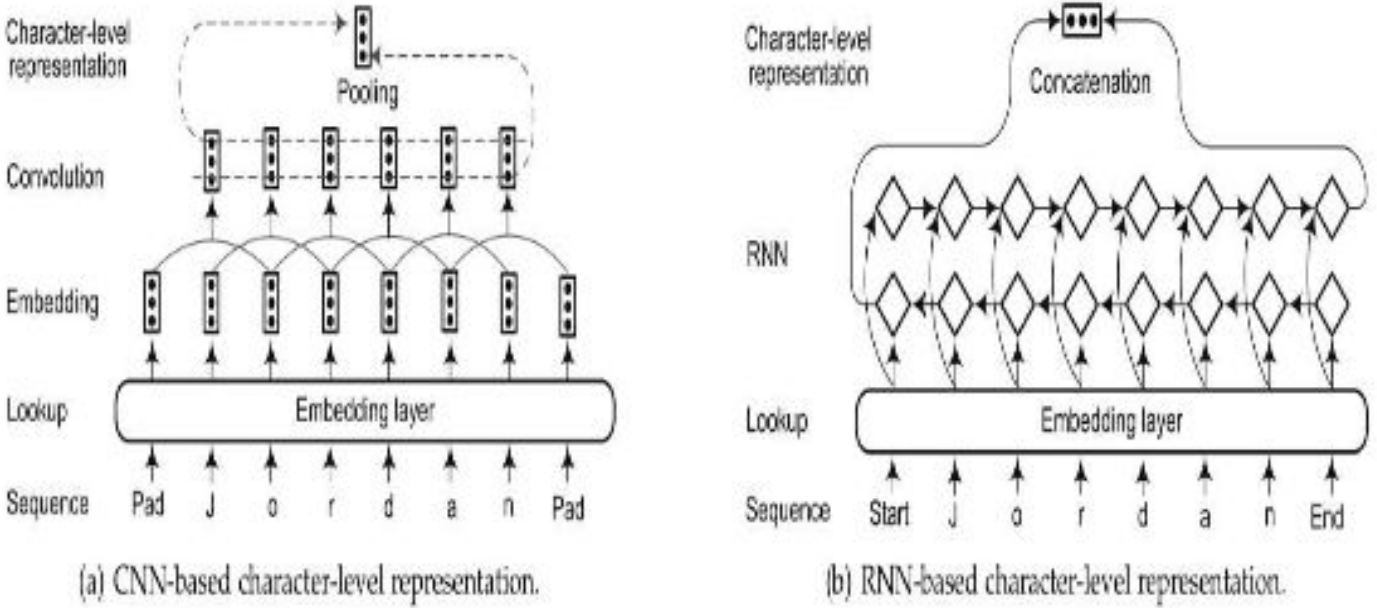
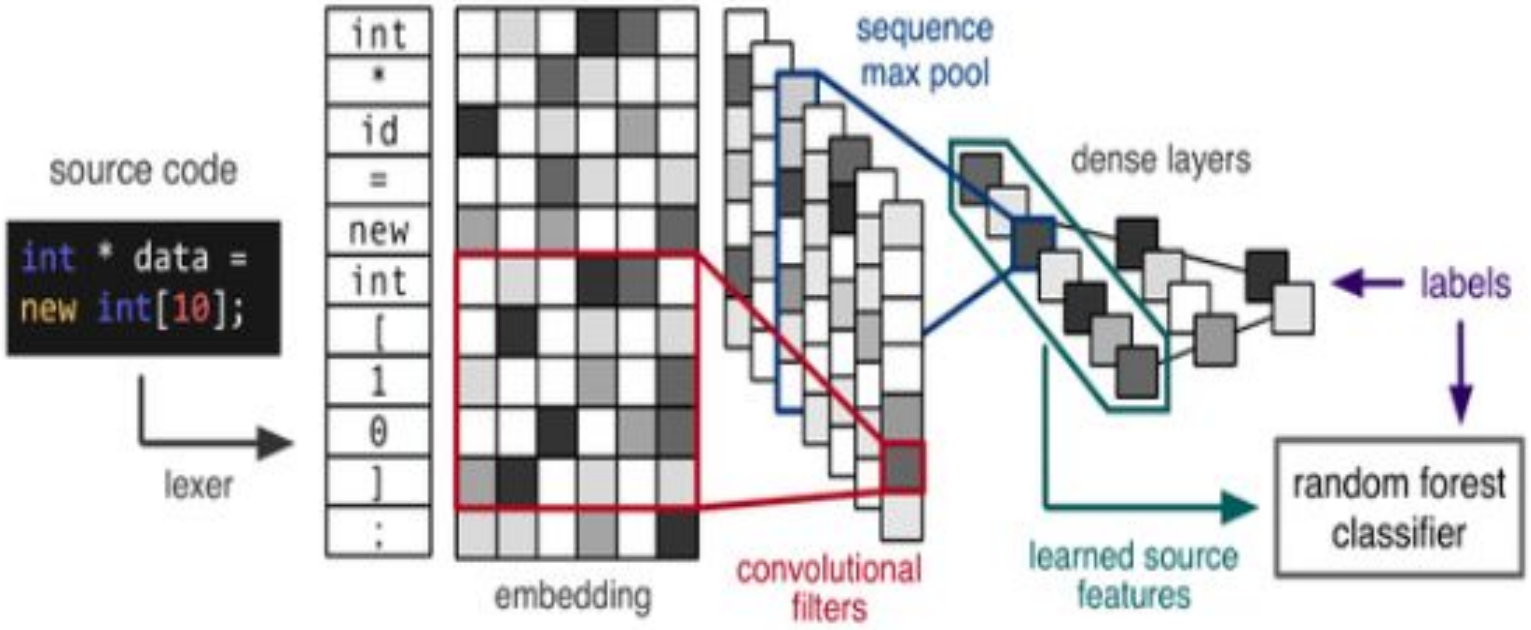


Fig. 2. CNN & RNN for extracting character-level representation for a word [9]

LITERATURE SURVEY-3

Literature review on vulnerability detection using NLP technology



LITERATURE SURVEY-4

An Adaptable Deep Learning-Based Intrusion Detection System to Zero-Day Attacks

Problem Statement:IDS distinguishes malicious traffic from the benign one and determines attack types targeting the assets but the main challenge is is detection of ZDTs.To overcome this problem DI-based IDSes are implemented.[4]

Methodology:The first phase distinguishes new attacks from older ones.The second phase gathers enough evidence and instances of same new attack fro deeper analysis and the third phase determines types of unknown traffic into categories:malicious traffic,new attack,unseen benign and temporary anomaly traffic.[4]

Drawbacks:

i.An adaptable IDS may primarily rely on anomaly detection methods which are effective at detecting unknown threats but they can be prone to FP and may struggle with know patterns.

ii.IDS does not specifically focus on vulnerability detection but rather on identifying unauthorized access or potential security breaches that might exploit vulnerabilities.

LITERATURE SURVEY-4

An Adaptable Deep Learning-Based Intrusion Detection System to Zero-Day Attacks

Pros:

ii.Overcomes the limitations of traditional ISDs which fail to detect ZDTs

ii.Adaptable DL-based IDss have shown potential to identify novel attacks strategies and anomalies.

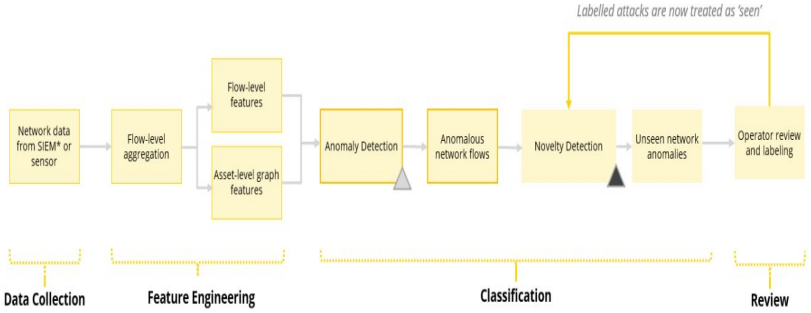
iv.Reduced false positives

LITERATURE SURVEY-5

Zero Day Threat Detection Using Autoencoders

Problem Statement: To overcome the flaws of dual-encoder approach to identify ZDTs in network flow and asset-level graph features,metric learning to tain second autoencoder or labeled attack data is implemented.[7]

Methodology: The study involves data collection;feature engineering involving flow-level and asser level features;anomaly detection classifies as Anomaly detector trained on benign network traffic designed to identify any malicious activity and Novelty detector trained on known attack types to identify previously unseen attack types.[7]



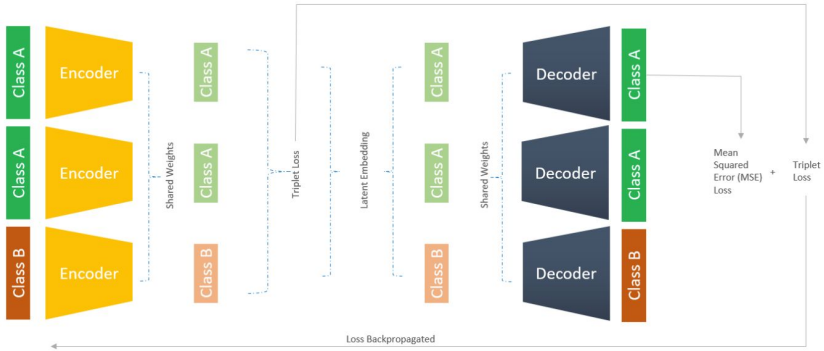


Fig. 2: Novelty Detector Triplet Training Architecture

LITERATURE SURVEY-5

Zero Day Threat Detection Using Autoencoders

Drawbacks:

i.The field of known attacks is more diverse than the labeled data available to us.Many attacks may be falsely labelled as ZDTs because they are unknown to the network in raining.

ii.Overfitting happens when the latent space collapses into smaller dimension destroying necessary information for the decoder.

iii.Anomaly detection only labels the anomaly inputs.

Pros:

i.Segregates novel threats from those seen during training despite low support and high attack variety and stronger performance than other ZDTs.

ii.Novel approach to attack type attribution and model interpretability that enables operators to identify the closest attack type to ZDT for easy threat hunting.

LITERATURE SURVEY-6

A Transformer-based Line-level Vulnerability Prediction

Problem Statement: The IV detect is limited to project specific dataset which are unable to capture the most accurate relationship between tokens and surroundings.The sub-graph interpretation of IV deetct is still coarse grained and not effective to capture the meaning of full long-term dependencies. To overcome the problems face a transformer based line-level vulnerability prediction is implemented.[5]

Methodology:[5]

i. Data Splitting where 80% of training data and 10% of validation data of testing is used.

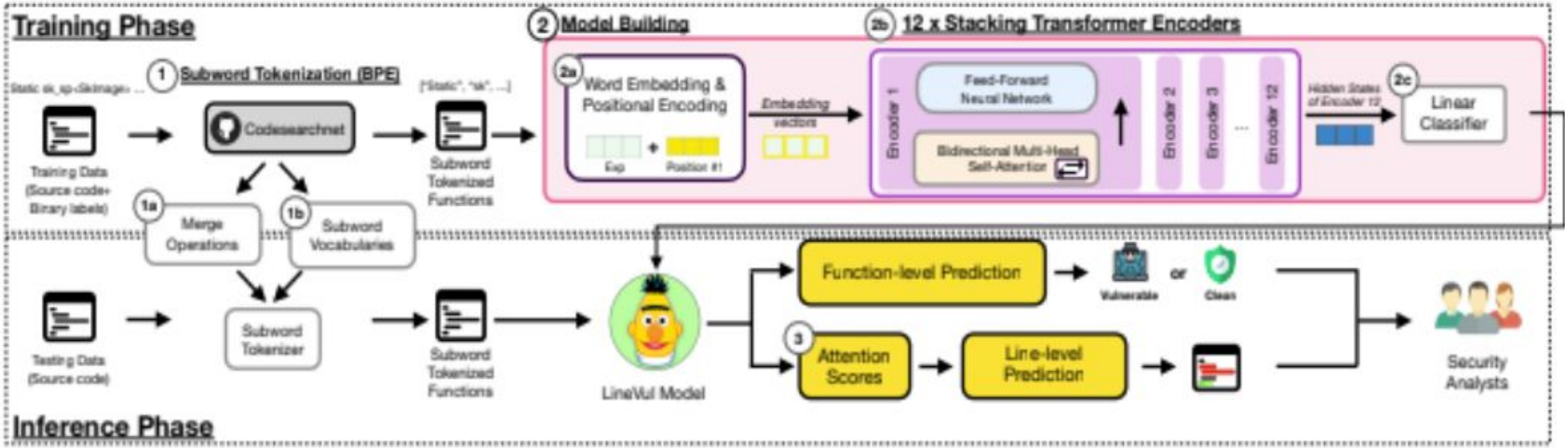
ii.Function level-model implementation: To implement, we need 2 python libraries i.e. Transformers and Pytorch.CodeBert tokenizer and CodeBert model is pre-trained.

iii.Line-level model implementation:By using attention matrix returned by fine-tuned model we integrate tokens into line score.The higher line score represents high vulnerabilities.

Iv. Hyper-parameter settings and Fine-tuning

LITERATURE SURVEY-6

A Transformer-based Line-level Vulnerability Prediction



LITERATURE SURVEY-6

A Transformer-based Line-level Vulnerability Prediction

Drawbacks:

i.High false positives

ii.Lack of zero day vulnerabilities detection

iii.Limited in addressing insecure design flaws.

iv.Unable to detect code duplication.

Pros:

i.High level automation and scalability achieved when compared to IV detect

ii.Non-intrusive

ii.Programming language Agnostic and cost-effective.

LITERATURE SURVEY-7

A Context-Aware Neural Embedding for Function-Level Vulnerability Detection

Problem Statement: The application of DL for code analysis requires transformation of software code to vector representations which is challenging to prevent the semantics and syntax of software.The paper explores the data driven approaches to ignore the similarities between words and the relationship by implementing different word embedding techniques and then design deep neural network.

Methodology:The methodology involves:

i.Code context analysis

ii.Contextual Semantic learning

iii.Handling out-of vocabulary words.

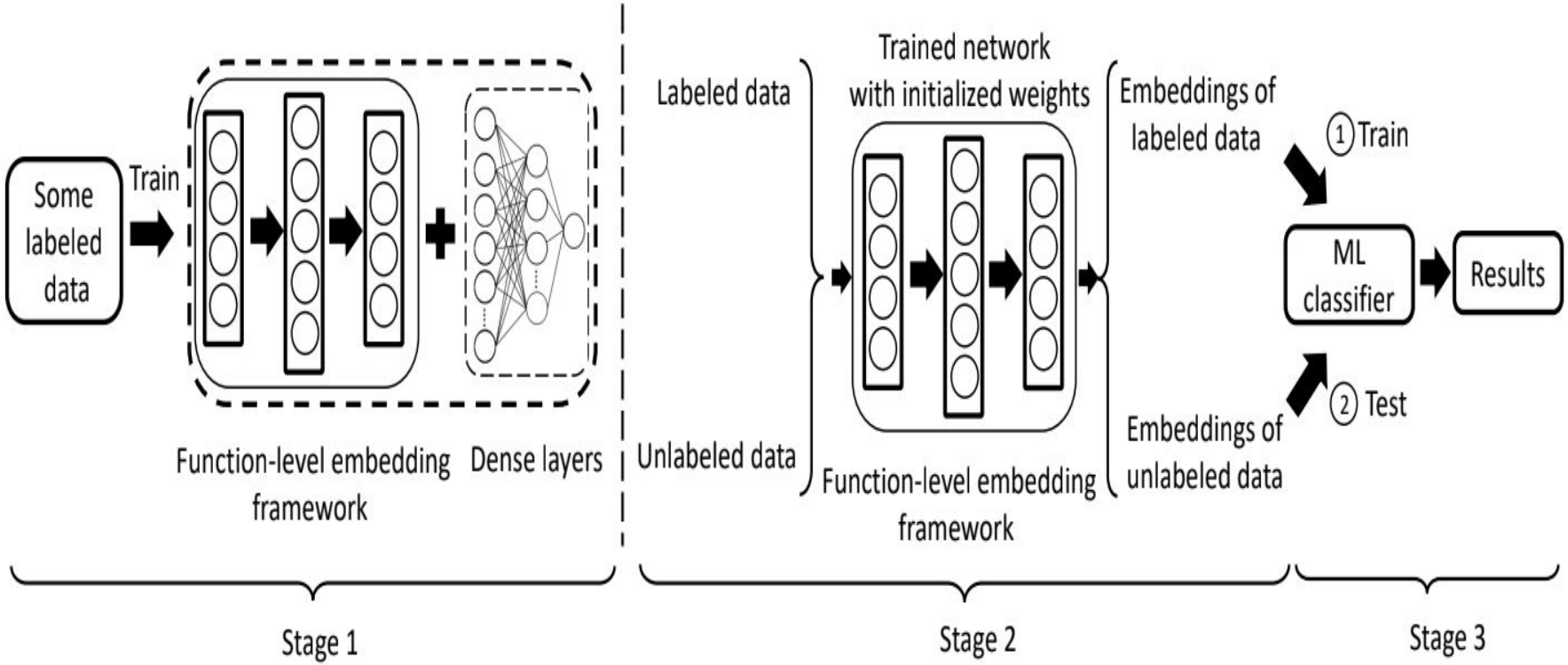
iv.Long-term dependency learning using LSTM

v.Input Preparation

vi.Function level embedding generation.

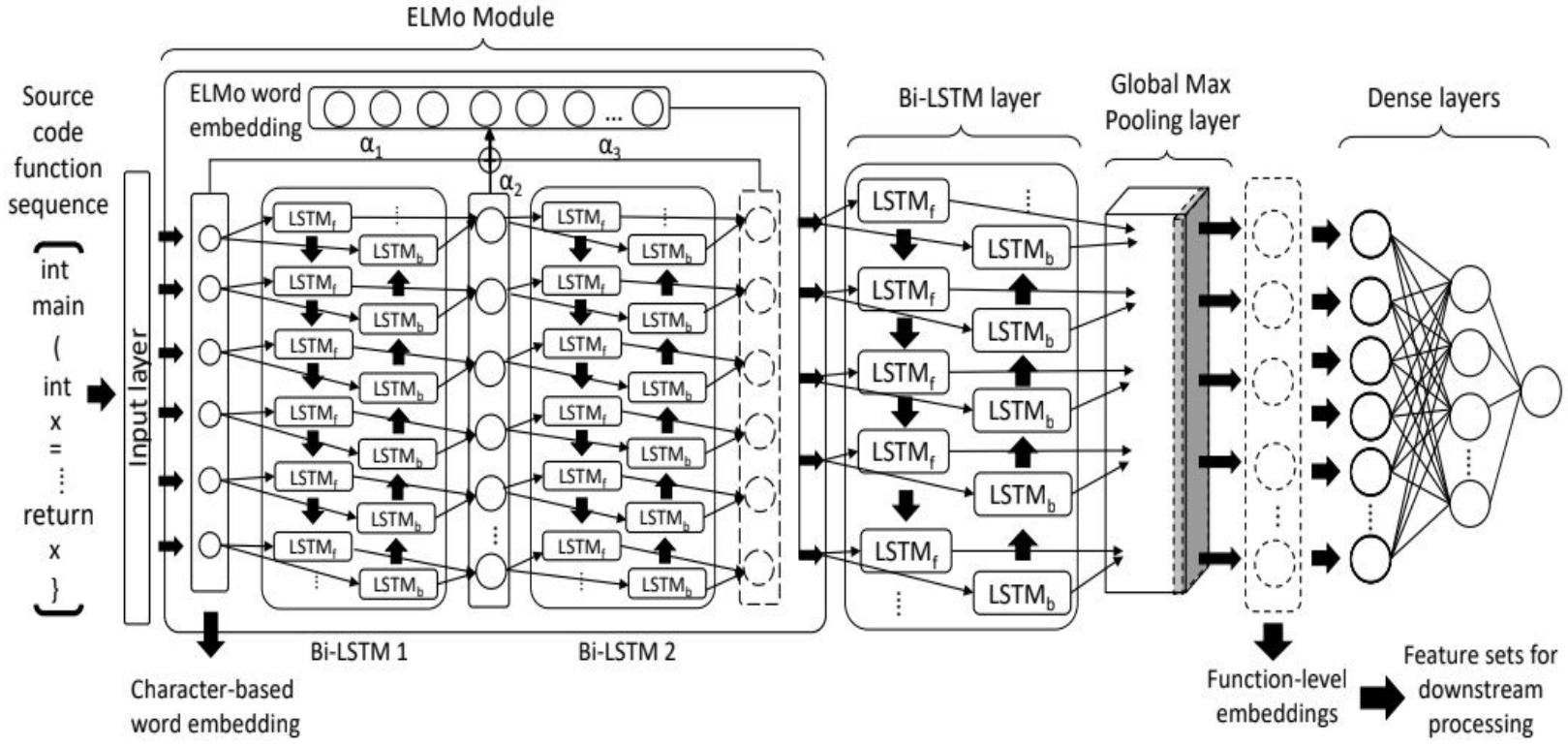
LITERATURE SURVEY-7

A Context-Aware Neural Embedding for Function-Level Vulnerability Detection



LITERATURE SURVEY-7

A Context-Aware Neural Embedding for Function-Level Vulnerability Detection



LITERATURE SURVEY-7

A Context-Aware Neural Embedding for Function-Level Vulnerability Detection

Drawbacks:

i.In terms of measuring effectiveness of generating code representations for vulnerability detection,ELMo with W2V,FastText,GloVe and BERT neglecting methods such as Code2Vec and code semantic generation.

ii.Truncating long sequences to a unified length of 100,it is possible to truncate the actual vulnerable code parts.

Iii. Performance of LSTM degrades when sequence length increases due to hidden state bottleneck.

iv.Not able to provide interpretations for detection results and no validation.

Pros:

i.Successfully learnt the hidden code semantics in the code.

LITERATURE SURVEY-8

Vulnerability detection in source code using deep representation learning

Problem Statement: Both static and dynamic analyzers are rule-based tools and this limited to their hand-engineered rules and not able to guarantee full test coverage of codebases.To supplement existing labeled vulnerability datasets they compiled vast datasets and labeled from 3 static analyzers that indicate exploits.

Methodology:

i.Dataset curation and labelling

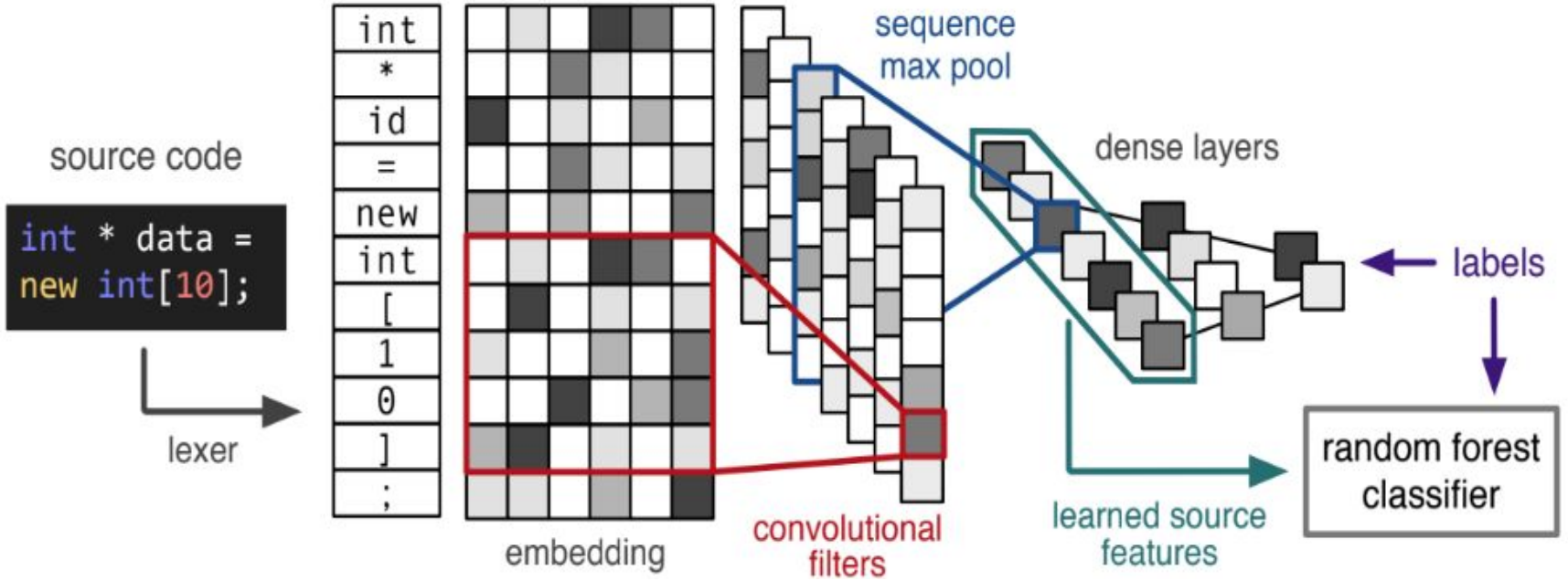
ii.Neural network classification and learning

iii.Recurrent feature extraction

iv.Ensemble learning:CNN and RNN

v.Training a BOW

LITERATURE SURVEY-8



LITERATURE SURVEY-8

Vulnerability detection in source code using deep representation learning

Pros:

i.ML methods have additional advantages over static analysis tools.

Drawbacks:

i.Improved labels such as those from dynamic analysis tools or mined from security patches are lacking.

LITERATURE SURVEY-9

Vulnerability Detection in PHP Web Application using Lexical Analysis approach with Machine learning.

Problem Statement:To overcome the static analysis method which needs manual inspection he paper introduces Vulnerability detection technique using lexical analysis with machine learning for classification.PHP token and AST are used as features for extraction of node type tokens using BFS.

Methodology:

i.Dataset collection from open-source.

ii.Feature extraction using TF-IDF.

ii.Balancing the dataset using SMOTE(Synthetic Majority Oversampling Technique)

iv.Pruning and using GBN for vulnerability detection.

Drawbacks:

i.Error in detection using AST feature occurs due to the pruning process and studied source code is too complex for analysis.

ii.Error in PHP token occurs as the framework only analyses vulnerable files.

LITERATURE SURVEY-9

A Transformer-based Line-level Vulnerability Prediction

Pros:

i.GNB is the best algorithm for vulnerability detection regardless of the features.

ii.Improvement in learning process

iii.Overcomes the limitations of manual static code analysis

Feature	Experiment
PHP Token	Classification with SVM, Decision Tree, GNB
	Classification with (SVM, Decision Tree, GNB) + SMOTE
	Classification with (SVM, Decision Tree, GNB) + Cluster Centroid
AST Token	Classification with SVM, Decision Tree, GNB
	Classification with (SVM, Decision Tree, GNB) + SMOTE
	Classification with (SVM, Decision Tree, GNB) + Cluster Centroid
AST Token + Pruning	Classification with SVM, Decision Tree, GNB
	Classification with (SVM, Decision Tree, GNB) + SMOTE
	Classification with (SVM, Decision Tree, GNB) + Cluster Centroid
Modify PHP Token	Classification with SVM, Decision Tree, GNB
	Classification with (SVM, Decision Tree, GNB) + SMOTE
	Classification with (SVM, Decision Tree, GNB) + Cluster Centroid

LITERATURE SURVEY-10

Discovering software vulnerabilities using data-flow analysis and machine learning.

Problem Statement: To overcome the limitations that vulnerabilities are covered only by the rule set we use data flow analysis for SQL injections and XSS. the Drawbacks:An adaptable IDS may primarily rely on anomaly detection methods which are effective at detecting unknown threats but they can be prone to FP and may struggle with know patterns.[6]

Methodology:[6]

i.Extracting features using reaching definition analysis,taint analysis and reaching constant analysis.

ii.Reaching definition analysis:Reaching definitions analysis is a data-flow analysis technique which statically determines which definitions may reach a given point in the code.

ii.Tainted data is data that can be modified (either directly or indirectly) by potentially malicious users, and thus can cause security problems at vulnerable points in the program.

iv.Tainted data can be sanitised, which removes harmful properties and hence transforms tainted data into untainted data. Vulnerabilities such as SQLi and XSS can be avoided by proper input sanitisation.

LITERATURE SURVEY-10

Discovering software vulnerabilities using data-flow analysis and machine learning.

Problem Statement: To overcome the limitations that vulnerabilities are covered only by the rule set we use data flow analysis for SQL injections and XSS. the Drawbacks:An adaptable IDS may primarily rely on anomaly detection methods which are effective at detecting unknown threats but they can be prone to FP and may struggle with know patterns.

Methodology:

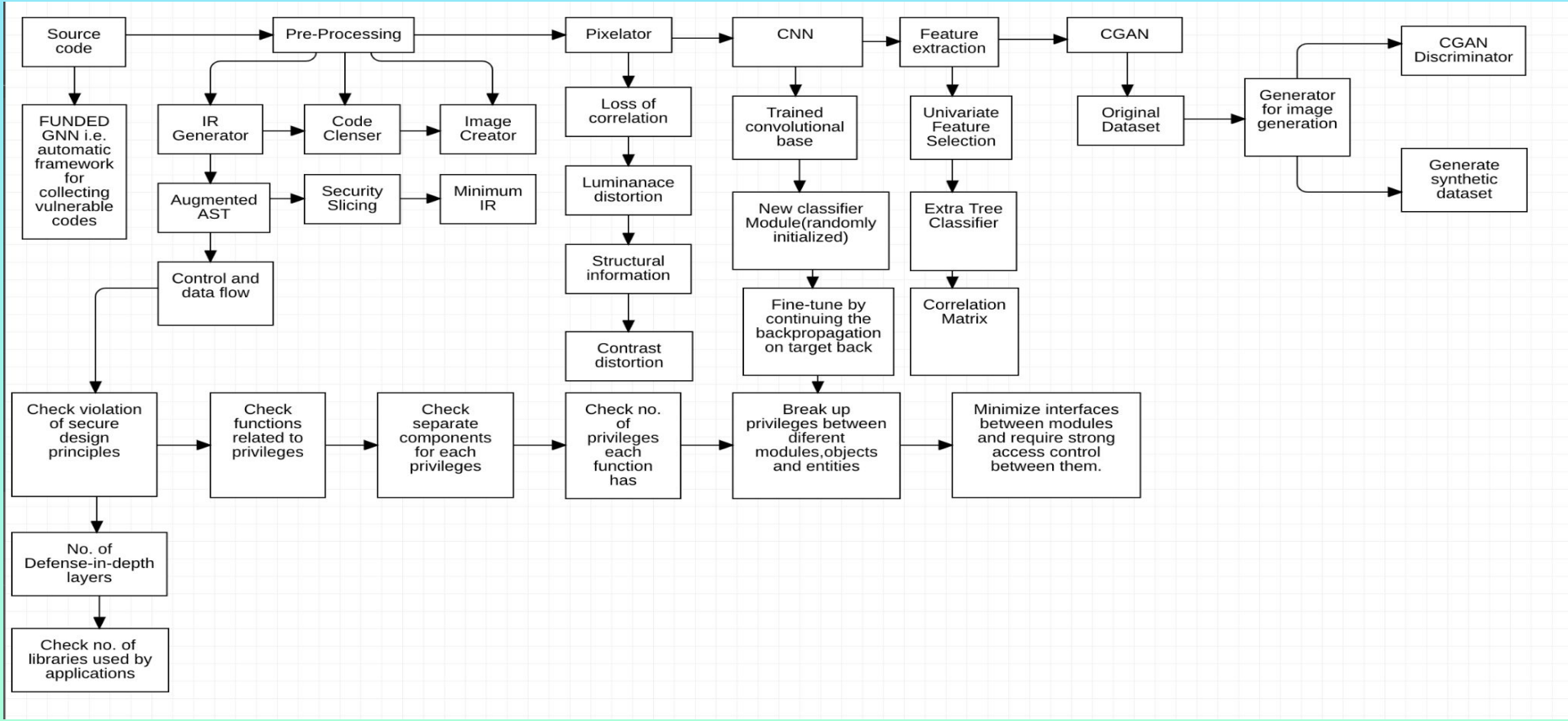
i.Extracting features using reaching definition analysis,taint analysis and reaching constant analysis.

ii.Reaching definition analysis:Reaching definitions analysis is a data-flow analysis technique which statically determines which definitions may reach a given point in the code.

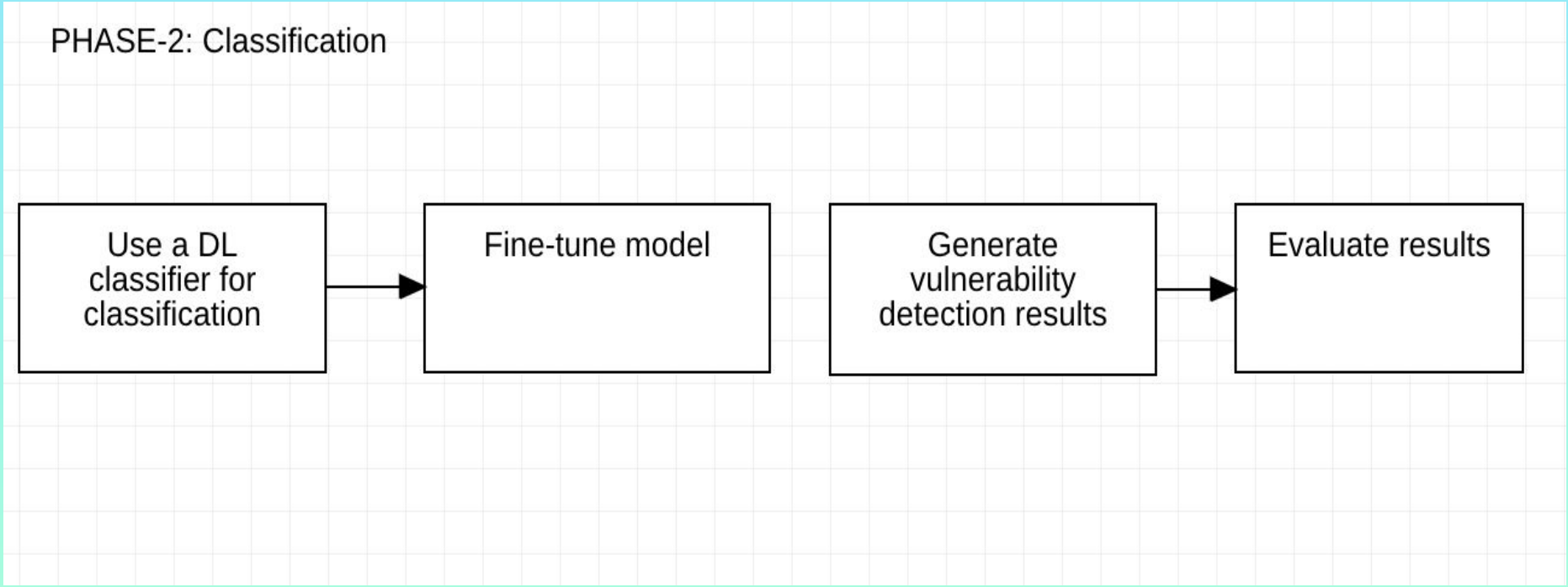
ii.Tainted data is data that can be modified (either directly or indirectly) by potentially malicious users, and thus can cause security problems at vulnerable points in the program.

iv.Tainted data can be sanitised, which removes harmful properties and hence transforms tainted data into untainted data. Vulnerabilities such as SQLi and XSS can be avoided by proper input sanitisation.

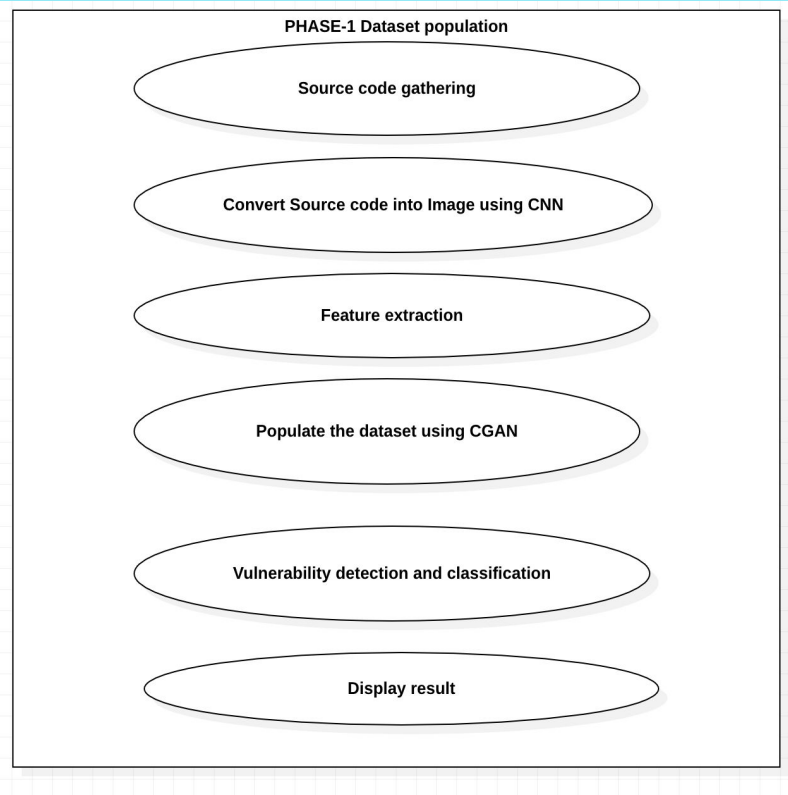
FLOW DIAGRAM

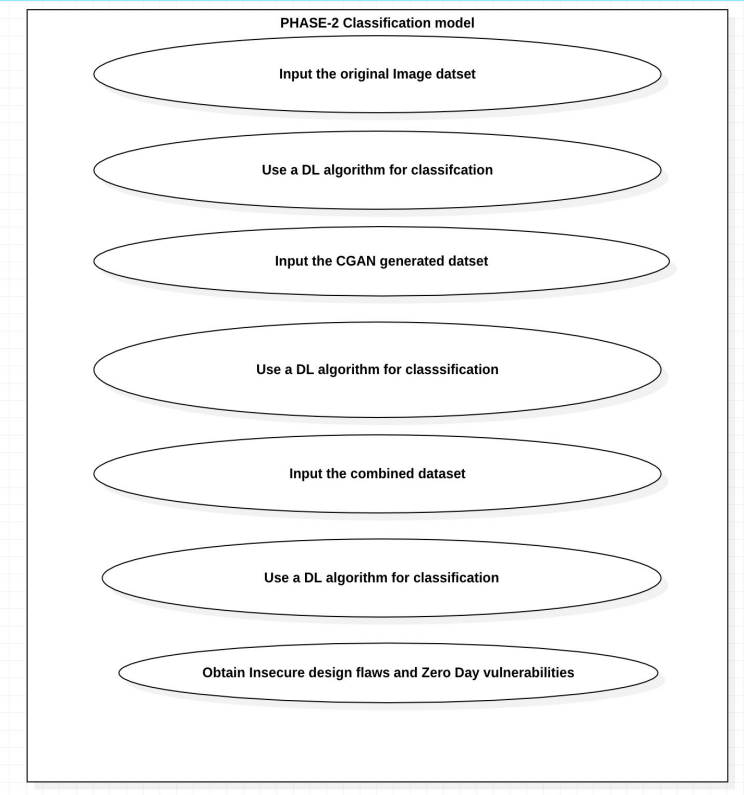


FLOW DIAGRAM

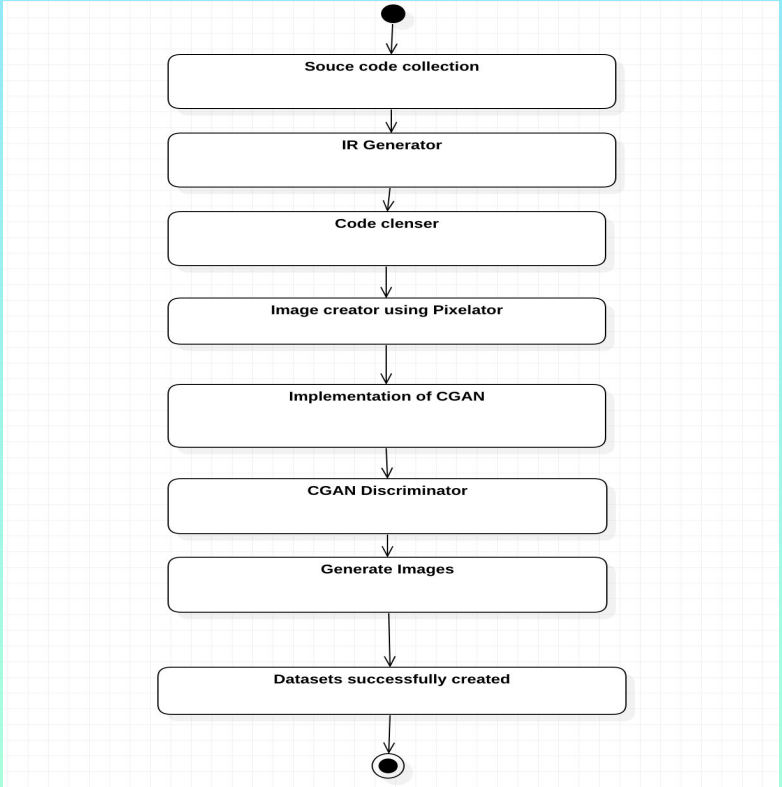


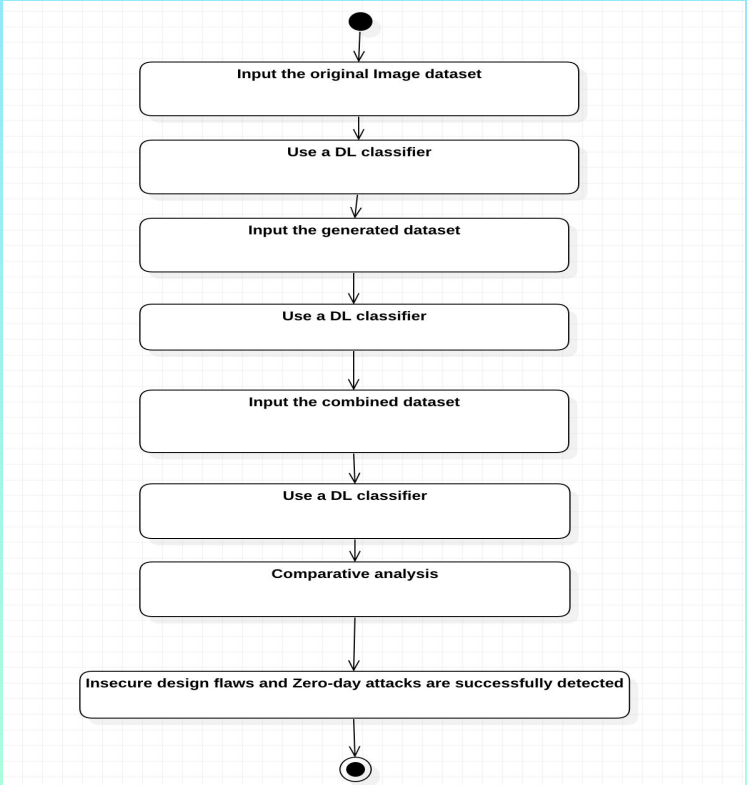
USE CASE DIAGRAM





STATE CHART DIAGRAM





PROBLEM STATEMENT

Despite the increasing importance of cybersecurity in today's digital landscape, our software/system design has demonstrated significant vulnerabilities and weaknesses that expose it to potential security breaches, data leaks, and unauthorized access. The current architecture and coding practices of our software lack adequate compartmentalization, leading to the mixing of critical state information with less sensitive data. This design flaw poses a significant security risk, as it increases the potential for unauthorized access and unintended data exposure. Furthermore, it complicates the task of maintaining and auditing the security of our software. To address this issue, we must establish a robust compartmentalization strategy and restructure our codebase to ensure the proper segregation of sensitive and non-sensitive data, reducing the risk of security breaches and enhancing the overall resilience of our software.

Currently existing tools like STRIDE have limited automation,incomplete coverage as they follow structural approach that do not cover all possible threats or unknown ZDTs , dependency on Accurate data and lack of data flow analysis.

Therefore to overcome the above mentioned drawbacks, data science approaches are used to introduce complementary and often more proactive ways to understand the flow of data.

PROBLEM STATEMENT

Human being's ability to learn from its surrounding visually it is evident that humans learn and interact with their surroundings based on their visual perception and eyes playing an important role in the process and have grown to be one of the complex sensory organs.So we took an inspiration to introduce Image processing to implement the ideology of learning through visual representation by converting source-code into image to learn the patterns and to overcome the tradition feature extraction methods.

Another purpose of our study is to lift the limitations due to lack of data i.e. there is only limited data available which create a huge trouble in detecting new unknown vulnerabilities.To overcome this problem we purpose a DL approach i.r Conditional Generative Adversarial Networks(CGAN) considered to reason about program structures to identify potential vulnerabilities at source code compared to other techniques as DL has an advantage of not requiring expert involvement to tune representations for program structures manually;instead it automatically generates synthetic data.

REFERENCES

[1] Somip Dey, Amit Kumar Singh, Dilip Kumar Prasad and Klaus Dieter Mcdonald-Maier, “SoCodeCNN: Program Source Code for Visual CNN Classification Using Computer Vision Methodology”, Oct. 2019,in IEEE Access, https://creativecommons.org/licenses/by/4.0/.

[2]Nikolaos Peppes, Theodoros Alexakis, Evgenia Adamopoulou, Konstantinos Demestichas, “The Effectiveness of Zero-Day Attacks Data Samples Generated via GANs on Deep Learning Classifiers.” Sensors 2023, 23, 900. https:// doi.org/10.3390/s23020900.

[3] M. S. Abdallah, G. H. Samaan, A. R. Wadie, F. Makhmudov, and Y.-I. Cho, “Literature review on vulnerability detection using NLP technology,” 23 Apr 2021, https://doi.org/10.48550/arXiv.2104.11230.

[4] Mahdi Soltani, Behzad Ousat, Mahdi Jafri Siavoshani and Amir Hossein Jahangir, "An Adaptable Deep Learning-Based Intrusion Detection System to Zero-Day Attacks," *20 Aug 2021, https://doi.org/10.48550/arXiv.2108.09199*.

[5] Michael Fu and Chakkrit Tantithamthavorn, “LineVul: A Transformer-based line-level Vulnerability Prediction”, 26 Mach 2022, DOI:10.1145/3524842.3528452.

REFERENCES

[6] Jorrit Kronjee, Arjen Hommersom,Harald Vranken Louis Kim, Lei H. Hamilton, Tomo Lazovich, Jacob A. Harer1, Onur Ozdemir, Paul M. Ellingwood, Marc W. McConley,Language Translation," "Automated Vulnerability Detection in Source Code Using Deep Representation Learning", August 2018, ARES '18: Proceedings of the 13th International Conference on Availability, Reliability and Security, https://doi.org/10.1145/3230833.3230856.

[7] Dhruv Nandakumara , Robert Schillera , Christopher Redino , Kevin Choia , Abdul Rahmana , Edward Bowena , Marc Vucovicha, Joe Nehilaa , Matthew Weeks, Aaron Shaha., "Zero Day Threat Detection Using Metric Learning Autoencoders",1 Nov 2022, https://doi.org/10.48550/arXiv.2211.00441.

[8] Huanting Wang , Guixin Ye , Zhanyong Tang , Shin Hwei Tan, Songfang Huang, Dingyi Fang, Yansong Feng, Lizhong Bian, and Zheng Wang, "Combining Graph-Based Learning With Automated Data Collection for Code Vulnerability," IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 16, 2021, https://ieeexplore.ieee.org/document/9293321.

REFERENCES

[9]Xin Li 1,2,3, Lu Wang 1 , Yang Xin 1,2,3,*, Yixian Yang 1,2,3 and Yuling Chen, “Automated Vulnerability Detection in Source Code Using Minimum Intermediate Representation,”2 March 2020, Applied Sciences MDPI, https://doi.org/10.3390/app10051692.

[10] Hongwei Wei 1 , Guanjun Lin , Lin Li and Heming Jia, “Context-Aware Neural Embedding for Function-Level Vulnerability Detection”,Mach 2021, Applied Sciences MDPI, https://doi.org/ 10.3390/a14110335

[11] Kim, J.-Y.; Bu, S.-J.; Cho, S.-B. Zero-Day Malware Detection Using Transferred Generative Adversarial Networks Based on Deep Autoencoders. Inf. Sci. 2018, 460–461, 83–102. [CrossRef]

[12] Won, D.-O.; Jang, Y.-N.; Lee, S.-W. PlausMal,”GAN: Plausible Malware Training Based on Generative Adversarial Networks for Analogous Zero-Day Malware Detection.”, IEEE Trans. Emerg. Top. Comput. 2022. [CrossRef]