A

Mini Project Report

on

" MALICIOUS URL DETECTION"

Submitted in partial fulfilment of requirement for the award of the of

MASTER OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted By

M VISWA SOWRABH REDDY 1005-23-742121

Under the Guidance of

DR. I. GOVARDHANA RAO

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING(AUTONOMOUS)

OSMANIA UNIVERSITY, HYDERABAD

JULY – 2024

UNIVERSITY COLLEGE OF ENGINEERING

(AUTONOMOUS)

OSMANIA UNIVERSITY, HYDERABAD, TELANGANA STATE, INDIA

A University Accredited by NAAC with A+ Grade

Category -I Graded Autonomy by UGC

CERTIFICATE

his is to certify that the thesis entitled "**MALICIOUS URL DETECTIC** ıbmitted by **M VISWA SOWRABH REDDY** Bearing **ROLL 00523742121**, student of Department of computer science engineering, Univer ollege of engineering, Osmania university, Hyderabad, is a record of bona :search work under my supervision and I consider it worthy of consideration ıe award of the degree of Master of Technology of the Institute.

:oject Guide (Dept. of CSE)
sst. Prof. Dr. I. GOVARDHANA
smania University

Head of Department, CSE
Prof. Dr. P.V. Sudha
Osmania University

UNIVERSITY COLLEGE OF ENGINEERING
(AUTONOMOUS)
OSMANIA UNIVERSITY, HYDERABAD, TELANGANA STATE, INDIA
A University Accredited by NAAC with A+ Grade
Category -I Graded Autonomy by UGC

DECLARATION

M VISWA SOWRABH REDDY bearing Roll No. **1005-23-742121**, he
eclare that the results presented in this project is the bonafide work done
arried out by me during the year 2024 in partial fulfilment of the **acade**
equirements for the award of **"Master of Technology"** in Computer Science
ngineering, Osmania University, Hyderabad, TELANGANA(INDIA). This pro
as done under the supervision of **Dr. V.B. Narasimha** Asst. Professor.
urther, I declare the report has not been submitted to any other University for
ward of any other degree.

Date: **Name and Signature of The Student**
Place: Hyderabad **M VISWA SOWRABH REDDY (1005-23-742121)**

ACKNOWLEDGEMENT

he satisfaction that accompanies the successful completion of any task would
ıcomplete the people who made it possible and whose constant guidance
ıcouragement crowns all efforts with success. They have been a guiding
ource of inspiration towards the completion of the project.

is my privilege and pleasure to express profound sense of respect, gratitude
ıdebtedness to my project guide, **Dr. I. GOVARDHANA RAO,** Assis
rofessor, Department of Computer Science and Engineering, **OSMA**
NIVERSITY, who have supported us throughout my project with patience
nowledge and guided me with valuable inputs and suggestions & indefatig
ıspiration, guidance, cogent discussion, constructive criticisms, and encourager
ıroughout this dissertation work.

express my sincere gratitude to Head of the Dept **Prof. Dr. P.V.Sudha**
omputer Science and Engineering, **OSMANIA UNIVERSITY,** for her suggesti
ıotivations and providing excellent infrastructure and a conducive atmosphere

ɔmpleting this project successfully. I convey my heartfelt thanks to the lab ɔr allowing me to use the required equipment whenever needed.

M VISWA SOWRABH REDDY (1005-23-7421

INDEX

ABSTRACT

Malicious URLs are harmful to every aspect of computer users. Detecting of malicious URL is very important. Currently, detection of malicious webpa techniques includes black-list and white-list methodology and machine learn classification algorithms are used. However, the black-list and white-list techno useless if a particular URL is not in list. In this paper, we propose a multi-l model for detecting malicious URL. The filter can directly determine the URL aining the threshold of each layer filter when it reaches the threshold. Otherv ıe filter leaves the URL to next layer. We also used an example to verify that ıodel can improve the accuracy of URL detection.

Keywords: Malicious URL; Black-list and White-list Technology; Mac earning; Multi-layer Filtering Model

1.INTRODUCTION

ı recent years, the Internet has been playing a bigger and bigger role in peop ork and life. Currently, we observed that not every website is user-friendly rofitable. More and more malicious websites began to appear and these malic ebsites endangering all aspects of the user. This can lead the user tow conomic losses, even some can create confusion over the management of ountry. Detecting and stopping malicious websites has become an impor ontrol measure to avoid the risk of information security [7-8] [10]. Generally, nly entrance to the website is URL, which can be malicious URL. At this leve ıtrance, the identification of URL is the best solution to avoid information loss ıe malicious URL identification has always been a hot area of informa ecurity. Spams, malicious webpages and URLs that redirect or mislead gitimate users to malware, scams, or adult content. It is perhaps correlated ıe use of the internet. To identify malicious URLs, ML-based classifiers c eatures from webpages content (lexical, visual, etc.), URL lexical features, red aths, host-based features, or some combinations of them. Such classifiers usu ct in conjunction with knowledge bases which are usually in browser URL BL om web service providers. If the classifier is fed with URL-based features, ommon to set a URL aggregator as a per-processor before extracting featu Mostly using supervised learning paradigm, Naive Bayes [1][13], Support Ve Machine with different kernels [9], and Logistic Regression are popular Mac earning classifiers for filtering spam and phishing [6]. Meanwhile, GT-b arning to deal with active attackers is also evaluated in spam filtering. In aper we solve this problem using multi-layer filtering model to do lassification which are able to handle their own relatively good data.

1.1.SCOPE OF THE PROJEC

The scope of a project on malicious URL detection typically includes several key components and considerations:

1. **Objective Definition:** Clearly define the goals and objectives of the project. Determine if the focus is on detection, prevention, or

both.

2. **Types of URLs:** Specify what types of URLs the system will detect (e.g., phishing URLs, malware distribution sites, fraudulent websites).
3. **Data Collection:** Decide how to collect URLs for analysis. This could involve using existing datasets, web scraping, or real-time monitoring.
4. **Feature Extraction:** Identify which features of URLs will be used for analysis. This might include domain age, domain registration information, URL structure, etc.
5. **Machine Learning Models:** Choose appropriate machine learning algorithms (e.g., decision trees, random forests, neural networks) for classification of URLs as malicious or benign.
6. **Training and Validation:** Develop a methodology for training the models using labeled datasets and validate their performance using appropriate metrics (e.g., accuracy, precision, recall).
7. **Real-Time Detection:** If applicable, design mechanisms for real-time detection of malicious URLs as they are encountered.
8. **Integration:** Consider how the detection system will be integrated into existing cybersecurity frameworks or applications.
9. **Scalability:** Ensure that the solution is scalable to handle large volumes of URLs efficiently.
10. **Evaluation:** Continuously evaluate and update the models to adapt to new types of malicious URLs and evolving cybersecurity threats.
11. **Ethical Considerations:** Address ethical concerns related to privacy, data security, and potential biases in the data or models.
12. **Documentation and Reporting:** Document the entire process, from data collection to model deployment, and provide clear reporting on the performance and effectiveness of the system.

By defining and addressing these aspects within the scope of the project, you can develop a robust malicious URL detection system that helps mitigate cybersecurity risks effectively.

1.2 OBJECTIVE

The primary objective of malicious URL detection is to identify and mitigate the risks posed by malicious URLs on the internet. This objective can be broken down into several key goals:

1. **Identification of Malicious URLs:** The core goal is to accurately identify URLs that lead to malicious content such as phishing sites, malware distribution sites, fraudulent websites, etc.
2. **Protection Against Cyber Threats:** By detecting malicious URLs, the objective is to protect users, organizations, and systems from various cyber threats that exploit unsuspecting users.
3. **Prevention of Attacks:** Detecting malicious URLs helps in preventing cyber attacks before they can cause harm, thereby enhancing overall cybersecurity posture.
4. **Early Warning System:** Provide an early warning system to users and organizations about potential threats lurking in URLs they may encounter or interact with.
5. **Enhancing Security Posture:** By continuously improving the detection capabilities and integrating them into cybersecurity frameworks, the objective is to enhance overall security defenses against evolving threats.
6. **Improving User Trust and Confidence:** Effective detection and prevention of malicious URLs contribute to building trust and confidence among users who rely on secure internet services and applications.
7. **Reducing Financial and Reputational Risks:** Mitigating the risks associated with falling victim to cyber attacks through malicious URLs can lead to reduced financial losses and protect the reputation of organizations and individuals.

In summary, the objective of malicious URL detection is to proactively identify and neutralize threats posed by malicious URLs, thereby safeguarding users, organizations, and systems from cyber attacks and their associated risks.

1.3 PROBLEM DEFINITION

The problem of malicious URL detection involves identifying URLs that lead to malicious content or activities on the internet. This is crucial due to the increasing sophistication and prevalence of cyber threats, such as phishing attacks, malware distribution, fraud, and other forms of malicious activities that exploit unsuspecting users. Here's a detailed problem definition:

1. **Identification of Malicious Intent:** The primary challenge is to differentiate between benign URLs (which lead to legitimate

content) and malicious URLs (which lead to harmful or deceptive content).

2. **Diverse Types of Malicious URLs:** Malicious URLs can encompass various types of threats, including phishing sites designed to steal personal information, malware-hosting sites that distribute harmful software, fraudulent websites that deceive users for financial gain, etc.
3. **Dynamic and Evolving Nature:** Malicious actors constantly evolve their tactics, techniques, and procedures (TTPs), making it challenging to keep detection methods effective and up-to-date.
4. **Scale and Volume:** With billions of URLs on the internet, the sheer volume makes it impractical to manually review each URL. Automated detection systems are necessary to handle the scale efficiently.
5. **Speed and Real-Time Detection:** Detection systems need to operate in real-time or near-real-time to prevent users from accessing malicious content as soon as threats are detected.
6. **Feature Extraction and Analysis:** Effective detection often relies on extracting features from URLs (such as domain reputation, URL structure, hosting information, etc.) and analyzing them using machine learning or heuristic-based approaches.
7. **False Positives and False Negatives:** Balancing between minimizing false positives (incorrectly flagging benign URLs as malicious) and false negatives (missing actual malicious URLs) is critical for the effectiveness of the detection system.
8. **Integration with Cybersecurity Frameworks:** The detection system should integrate seamlessly with existing cybersecurity frameworks and tools used by organizations to provide comprehensive protection against cyber threats.
9. **User Education and Awareness:** Addressing the human factor by educating users about the risks associated with clicking on suspicious URLs and promoting safe internet practices.
10. **Privacy and Ethical Considerations:** Ensuring that detection methods respect user privacy and adhere to ethical standards, while also protecting against potential biases in data or algorithms.

In conclusion, the problem of malicious URL detection is multifaceted, requiring advanced technological solutions, continuous adaptation to evolving threats, and a holistic approach to cybersecurity to effectively protect users and organizations from the dangers posed by malicious URLs on the internet.

2.LITERATURE SURVEY

1. **Is this URL Safe: Detection of Malicious URLs Using Global Vector for Word Representation** (2022) by R. Bharadwaj, A. Bhatia, L. D. Chhibbar, K. Tiwari and A. Agrawal [1]. This paper proposes a novel approach for detecting malicious URLs using a global vector for word representation (GVWR). GVWR is a technique that can capture the semantic similarity between words, which is useful for identifying malicious URLs that often contain suspicious keywords or phrases.
2. **Malicious and Benign URL Dataset Generation Using Character-Level LSTM Models** (2022) by S. Vecile, K. Lacroix, K. Grolinger and J. Samarabandu [2]. This paper presents a method for generating a large dataset of malicious and benign URLs using character-level long short-term memory (LSTM) models. LSTM models are a type of recurrent neural network that is well-suited for sequential data, such as URLs. The generated dataset can be used to train machine learning models for malicious URL detection.
3. **Developing and Evaluating an Artificial Intelligence Model for Malicious URL Detection** (2022) by Nora A. A. and Narmatha C [3]. This paper describes the development and evaluation of an artificial intelligence (AI) model for malicious URL detection. The AI model is based on a convolutional neural network (CNN), which is a type of deep learning model that is well-suited for image and text classification. The model was evaluated on a dataset of real-world malicious and benign URLs, and it achieved a high accuracy rate.

5.SYSTEM REQUIREMENTS

3.1. SOFTWARE REQUIREMENT

Operating System : Windows

Programming Language : Python

Environment : Anaconda

Python libraries : • jupyter

- matplotlib
- numpy
- pandas
- scikit-learn
- seaborn
- tensorflow
- pip
- streamlit

3.2. HARDWARE REQUIREMENT

Processor : Intel Core i5

Hard disk : 512 GB

RAM : 8 GB

Device : Laptop

4.SYSTEM ANALYSIS

4.1 PROPOSED SYSTEM

- Malicious URLs are harmful to every aspect of computer users. Detecting of the malicious URL is very important. Currently, detection of malicious webpages techniques includes black-list and white-list methodology and machine learning classification algorithms are used. However, the black-list and white-list technology is useless if a particular URL is not in list. In this paper, we propose a multi-layer model for detecting malicious URL. The filter can directly determine the URL by training the threshold of each layer filter when it reaches the threshold. Otherwise, the filter leaves the URL to next layer. We also used an example to verify that the model can improve the accuracy of URL detection

Proposing algorithms for malicious URL detection involves selecting and implementing techniques that can effectively distinguish between benign and malicious URLs. Here are several algorithms commonly used or proposed for this purpose:

1. Machine Learning Algorithms

Machine learning algorithms are widely used in malicious URL detection due to their ability to learn patterns from labeled datasets and make predictions on unseen data:

- **Decision Trees:**
  - **Description:** Decision trees recursively split data based on features to classify URLs.
  - **Advantages:** Intuitive interpretation, handles categorical and numerical data well.
  - **Considerations:** Prone to overfitting with complex trees.
- **Random Forests:**
  - **Description:** Ensemble of decision trees trained on random subsets of data.
  - **Advantages:** Reduces overfitting, improves generalization.
  - **Considerations:** Requires more computational resources than single decision trees.
- **Support Vector Machines (SVM):**

- **Description:** Constructs hyperplanes in a high-dimensional space to separate classes of URLs.
- **Advantages:** Effective in high-dimensional spaces, versatile with different kernel functions.
- **Considerations:** Requires tuning of kernel parameters, can be sensitive to noise.

- **Neural Networks:**
  - **Description:** Deep learning models that learn complex patterns from data.
  - **Advantages:** Can capture intricate relationships in data, suitable for large datasets.
  - **Considerations:** Requires significant computational resources for training, complex to interpret.

2. Clustering Algorithms

Clustering algorithms group URLs into clusters based on similarity measures, which can aid in identifying outliers or anomalies (potentially malicious URLs):

- **K-Means:**
  - **Description:** Divides data into K clusters based on feature similarity.
  - **Advantages:** Simple and efficient, works well with large datasets.
  - **Considerations:** Requires specifying the number of clusters (K).
- **DBSCAN (Density-Based Spatial Clustering of Applications with Noise):**
  - **Description:** Clusters URLs based on density of points in feature space.
  - **Advantages:** Automatically detects outliers (noise), handles irregularly shaped clusters.
  - **Considerations:** Sensitivity to parameters (epsilon, minPts) affecting cluster quality.

3. Anomaly Detection Algorithms

Anomaly detection algorithms identify URLs that deviate from normal behavior, which can indicate malicious activity:

- **Isolation Forest:**

    - **Description:** Constructs isolation trees to isolate anomalies (malicious URLs) in feature space.
    - **Advantages:** Efficient for high-dimensional data, scales well with large datasets.
    - **Considerations:** Parameter tuning for tree depth and number of trees.
- **One-Class SVM:**
    - **Description:** Learns a boundary around normal data points to identify outliers (malicious URLs).
    - **Advantages:** Effective for novelty detection, suitable for datasets with few anomalies.
    - **Considerations:** Sensitive to choice of kernel and parameters.

4. Hybrid Approaches

Hybrid approaches combine multiple algorithms or techniques to leverage their strengths and improve detection accuracy:

- **Feature-Based + Behavioral Analysis:**
    - **Description:** Combine feature extraction from URLs with dynamic monitoring of URL behavior.
    - **Advantages:** Captures both static and dynamic aspects of URLs, enhances detection robustness.
    - **Considerations:** Integration complexity, computational overhead.
- **Machine Learning + Heuristic Rules:**
    - **Description:** Integrate machine learning models with heuristic rules (e.g., blacklists, patterns) for enhanced detection.
    - **Advantages:** Utilizes both data-driven and rule-based approaches, improves accuracy and reliability.
    - **Considerations:** Maintenance of rules and updating with evolving threats.

4.2 EXISTING SYSTEM

- In view of the situation of mass URL examination and filtering in the power information network, this paper proposes a large-scale URL pattern string matching algorithm based on the Wu-Manber algorithm. The proposed algorithm improves the matching of the algorithm from the angle of reducing the hash conflict and reducing the number of accurate checksums. Match performance. Tests on real data sets show that the algorithm has lower memory consumption. The algorithm proposed in this paper improves the performance of large scale URL matching, and the detection method can be applied to many network filtering occasions. .

Machine learning (ML) has become a powerful tool for detecting malicious URLs. Here's an overview of some commonly used algorithms for this task:

Supervised Learning Algorithms:

These algorithms learn from labeled data (URLs classified as malicious or benign) to predict the category of new URLs.

- **Support Vector Machines (SVMs):** Efficiently classify URLs by finding a hyperplane that best separates malicious and benign examples in the feature space.
- **Random Forests:** Create an ensemble of decision trees, each trained on a random subset of features and data points. This reduces overfitting and improves generalization.
- **Logistic Regression:** Estimates the probability of a URL being malicious based on its features. Simple to interpret but might not be suitable for complex patterns.
- **Gradient Boosting:** Combines multiple weak decision trees sequentially, where each tree learns from the errors of the previous one. This approach can achieve high accuracy for complex classification problems.

Unsupervised Learning Algorithms:

These algorithms identify anomalies or outliers in unlabeled data (URLs without pre-defined classification). They can be useful for detecting new and unseen malicious patterns.

- **K-Means Clustering:** Groups URLs with similar features into clusters. Malicious URLs might form distinct clusters separate from benign ones.
- **Local Outlier Factor (LOF):** Identifies data points that significantly deviate from their local density. This can help detect outlier URLs that might be malicious.
- **Isolation Forest:** Isolates anomalous data points by randomly partitioning the feature space. The number of partitions required to isolate a point can indicate its anomaly score.

Deep Learning Algorithms:

These algorithms learn complex patterns from large datasets and can be particularly effective for feature extraction and classification tasks.

- **Convolutional Neural Networks (CNNs):** Can automatically learn relevant features from the URL string itself, potentially capturing hidden patterns indicative of malicious intent.
- **Recurrent Neural Networks (RNNs):** Can handle sequential data like URLs, taking into account the order of characters or words within the URL for better analysis.

Other Techniques:

- **Ensemble Methods:** Combining predictions from multiple algorithms (supervised, unsupervised, or deep learning) can often improve overall accuracy and robustness.
- **Transfer Learning:** Leveraging pre-trained models on similar tasks (e.g., text classification) can be a good starting point for building custom URL detection models, especially when dealing with limited data.

.3 PROJECT MODULES

A malicious URL detection system can be broken down into several key modules responsible for specific functionalities:

1. **URL Ingestion:**
    - This module receives URLs from various sources:
        - Web browser extensions
        - Email attachments
        - Security scans
        - Network traffic analysis (optional)
    - It pre-processes the URLs by removing irrelevant characters, normalizing formats, and potentially extracting basic features (length, presence of special characters).
2. **Blacklisting:**
    - This module maintains a frequently updated list of known malicious URLs.
    - Incoming URLs are checked against the blacklist.
    - **Match:** The URL is flagged as malicious and blocked (access denied).
    - **No Match:** The URL proceeds to further analysis.
3. **Feature Engineering:**
    - This module extracts informative features from the URL itself, potentially its content (if accessible), and historical data.
    - Features can include:
        - **URL-based:** Length, presence of suspicious characters, keywords, subdomain analysis.
        - **Host-based:** Domain age, registration details, website popularity.
        - **Content-based (if accessible):** Presence of malware signatures, phishing indicators, suspicious code.
        - **Historical data:** Past user reports, website reputation data.
4. **Machine Learning Model:**
    - This module utilizes a trained machine learning model to analyze the extracted features and predict the probability of a URL being malicious.
    - The model can be:
        - **Supervised learning:** Trained on labeled data (malicious vs. benign URLs) for core classification.

    - **Unsupervised learning:** Used for anomaly detection to identify outlier URLs with unusual characteristics.
- The model outputs a score or probability indicating the likelihood of a URL being malicious.

5. **Decision Engine:**
    - This module utilizes a threshold or rule-based system to make the final decision based on the model's output and potentially other factors:
        - **Confidence Score:** How confident the model is in its prediction.
        - **Threat Landscape:** Current trends in malicious URL patterns.
    - **Action:**
        - **High Probability:** The URL is flagged as malicious and potentially further investigated.
        - **Low Probability:** The URL is considered safe for now, but the system might keep track of user interactions with it for future analysis.
6. **Alerting and Reporting:**
    - This module generates alerts for flagged URLs, notifying administrators or security applications for further action.
    - It can also generate reports:
        - Track trends in malicious URL attempts
        - Analyze attack patterns
        - Evaluate system performance
7. **User Feedback Integration (Optional):**
    - This module allows users to report encountered suspicious URLs or verify encountered ones as benign.
    - This data can be used for:
        - **Model Retraining:** Continuously improve the machine learning model's accuracy.
        - **Blacklist Enrichment:** Update the blacklist with user-reported malicious URLs.
8. **Explainable AI (XAI) Integration (Optional):**

- This module incorporates techniques to provide insights into the machine learning model's decision-making process.
- This helps:
    - **Identify Feature Importance:** Understand which features contribute most to the model's classification.
    - **Debug Biases:** Detect and address potential biases in the training data that can impact results.

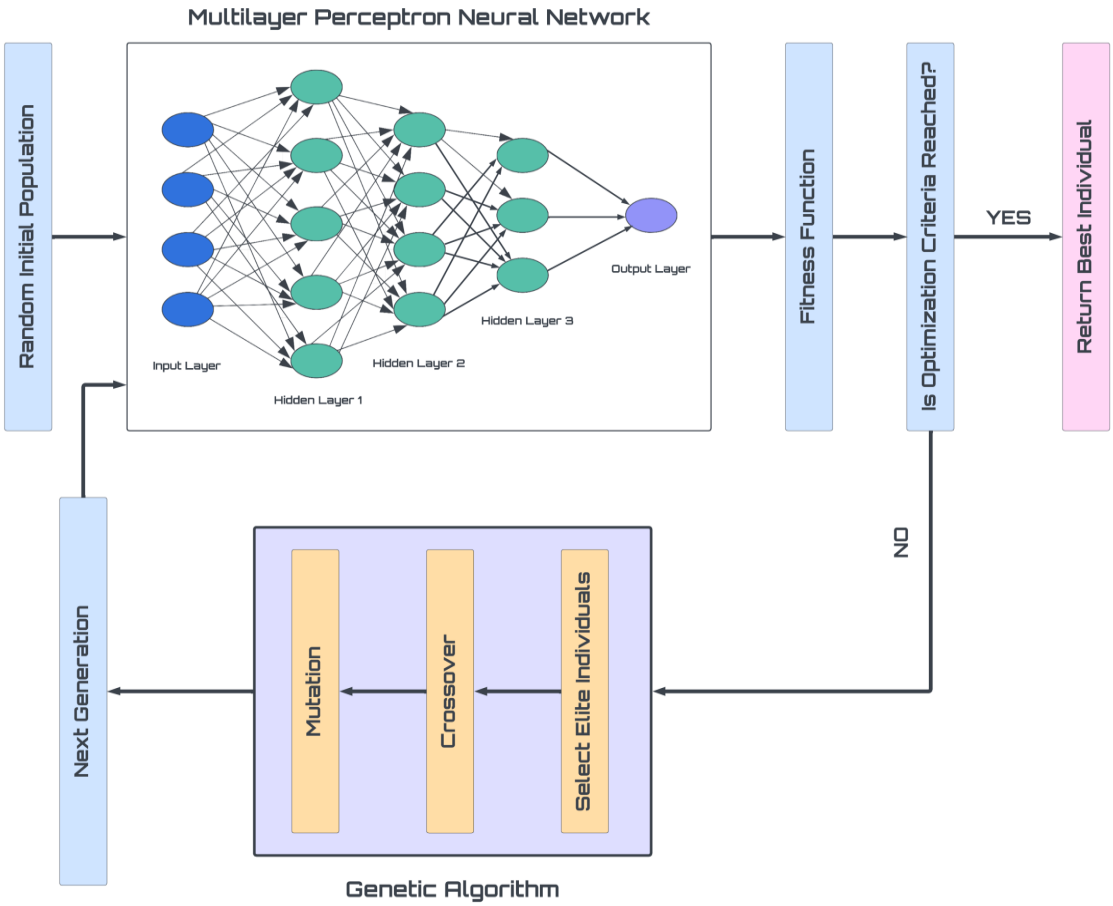
By combining these modules, you can create a comprehensive and adaptable system for detecting malicious URLs and protecting users from online threats. The specific modules and functionalities might vary depending on the chosen approach and desired system complexity.

5.SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

A multilayer perception is a feedforward artificial neural network that consist multiple layers of interconnected nodes also known as neurons.

- It is characterized by several layers of input nodes connected as a directed graph between the input and output layers.
- It also utilizes backpropagation for training the model.
- Input Layer (16 features)
- ↓
- Hidden Layer (32 neurons, ReLU)
- ↓
- Hidden Layer (16 neurons, ReLU)
- ↓
- Hidden Layer (8 neurons, ReLU)
- ↓
- Output Layer (1 neuron, Sigmoid)

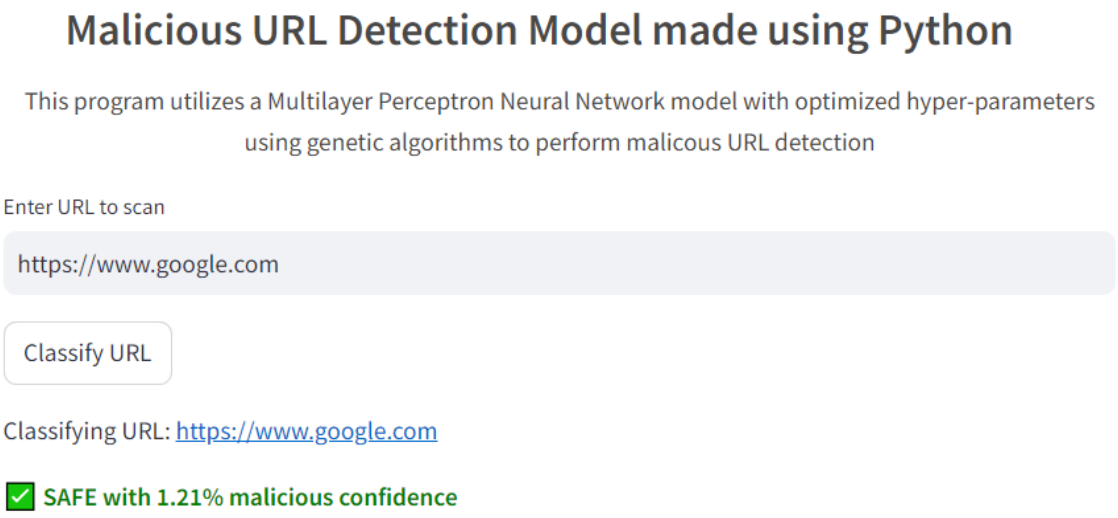

Multilayer Perceptron Neural Network
Random Initial Population
Input Layer
Hidden Layer 1
Hidden Layer 2
Hidden Layer 3
Output Layer
Fitness Function
Is Optimization Criteria Reached?
YES
Return Best Individual
NO
Next Generation
Mutation
Crossover
Select Elite Individuals
Genetic Algorithm

```python
# Example usage
whitelist = ["example.com", "safe-site.org"]
patterns = ["bad-site.com", "malicious.com"]
ml_model = DummyModel()  # Placeholder for an actual trained ML model

filter = MultiLayerFilter(whitelist, patterns, ml_model)
urls = ["example.com", "bad-site.com/path", "newmaliciousssite.net"]

for url in urls:
    result = filter.filter_url(url)
    print(f"URL: {url}, Result: {result}")
```

```
URL: example.com, Result: Safe
URL: bad-site.com/path, Result: Malicious
URL: newmaliciousssite.net, Result: Malicious
```



6.IMPLEMENTATION

6.1. DEVELOPMENT ENVIRONMENT

Python: Used for implementing machine learning algorithms, data preprocess and
integrating various components of the system.

NumPy and Pandas: Essential for data manipulation and preprocessing tasks.

Matplotlib: Tool for visualizing data distributions, model performance met and
results.

TensorFlow: Deep learning frameworks employed for building, training, and evaluating CNN models.

Streamlit: Framework for building interactive web applications for m prediction
and evaluation.

Integrated Development Environment (IDE):

Ananconda or Visual Studio Code (VS Code): Providing advanced code editir features, debugging capabilities, and integration with version control systems Git.

6.3 ALGORITHM

- **Multi-Layer filter Model(4)**
  - **Stratified filter**
    - **Black and White list Filter**
    - **Naïve Bayesian filter**
  - **Alpha N-bayes threshold training**
  - **CART decision tree filter**
  - **SVM filter**
- **Multi-Layer Perceptron(Genetic Algorithm)**

- Black and white list filter: The model's first-level filter is a black-and-white list filter which will be validated by recognizing the normal URLs and Malicious URLs. The normal URL addresses are stored in the white list file, while the malicious URL addresses are stored in the blacklist file. To detect the URL, we traverse the list of black and white to determine whether the URL in the black list or in the white list.
- Naive Bayesian filter: The second layer filter in this model is a naive Bayesian filter that trains the model by dividing into two main steps: By training the URL samples, we use two-dimensional arrays C1 and C2 to store the probability of each value of malicious website and normal website, such as formula

- Black and white list filter: The model's first-level filter is a black-and-white list filter which will be validated by recognizing the normal URLs and Malicious URLs. The normal URL addresses are stored in the white list file, while the malicious URL addresses are stored in the blacklist file. To detect the URL, we traverse the list of black and white to determine **whether the URL in the black list or in the white list.**
- Naive Bayesian filter: The second layer filter in this model is a naive Bayesian filter that trains the model by dividing into two main steps: By training the URL samples, we use two-dimensional arrays C1 and C2 to store the probability of each value of malicious website and normal website, such as formula

- The model's third-level filter is CART Decision Tree Filter. The model is further split into two steps:
- 1) model training: The CART tree is constructed by training samples with URLs, and the leaf nodes are decision nodes, and store the CART tree in the file system.

2) nbayes cart threshold max(n/ m,m/ n) training: Let (n represents the number of malicious URL leaf nodes, m represents the number of normal URL leaf nodes), so the size of a cart can characterize the type of occupation that a leaf node decides. Similar to naive Bayesian filter, this threshold can neither be set too small nor too large. So, to train the threshold, through the same training data group, the specific method is discussed in Section 3. If nbayes nbayes , then we think that URL is CART Decision tree model has a good at data, otherwise, record the classification results, and the URL is filtered to the next filter.

The this malicious URL dataset in experiment is downloaded from the malicious website lab (Http://www.mwsl.org.cn/), the normal URL dataset is collected from first category directory (http://www.dir001.com/). 10000 samples are taken from each dataset i.e. 10000 from malicious URLs and 10000 from normal URLs. This article uses features extraction and data modeling. Python language is used as the implementation programming. Windows 10 64bit as an operating system and Core i5 with 16GB RAM was used as personal computer.

Table 1 feature vector extraction rules

No	Features
F1	The domain names contained more than 4 consecutive numbers
F2	The domain name contains special characters (#, $, @, ~, _, -)
F3	Top Five domain name (com, en, net, org, cc)
F4	The number of "." in domain name
F5	domain name total length
F6	The Length of longest domain name segment
F7	Meaningful coefficients in primary domain names

8.ANALYSIS AND TESTING

Analyzing the results of a malicious URL detection system involves interpreting the system's output and taking appropriate actions. Here's a breakdown of key steps:

1. **Classifications:**
    - **Blocked URL:** The system identified the URL as malicious and blocked access.
        - **Review:** Investigate further to confirm if the block is legitimate (true positive) or a mistake (false positive).
        - **Action:** If a false positive, adjust the system or whitelist the URL. If a true positive, take further action based on the threat type (e.g., isolate infected devices, report phishing attempts).
    - **Allowed URL:** The system classified the URL as benign.
        - **Context Matters:** Consider the context in which the URL appeared (e.g., user browsing trusted sites vs. unsolicited email).

    - **Potential False Negative:** If there's high suspicion, manually investigate the URL or rescan it with updated threat intelligence.
2. **Model Performance Metrics:**
    - **Accuracy:** Overall percentage of correctly classified URLs (malicious and benign).
    - **False Positive Rate (FPR):** Percentage of benign URLs incorrectly blocked.
        - **High FPR:** Investigate potential issues with the model or training data. Consider adjusting thresholds or retraining the model.
    - **False Negative Rate (FNR):** Percentage of malicious URLs that slipped through undetected.
        - **High FNR:** Indicates the model might miss new or sophisticated threats. Update the model with fresh data or investigate alternative detection techniques.
3. **Alert and Report Analysis:**
    - **Alerts:** Analyze triggered alerts for flagged URLs.
        - **Prioritize:** Focus on alerts with high confidence scores or those involving critical systems.
        - **Investigate:** Research the flagged URL and threat type to determine the appropriate response.
    - **Reports:** Regularly review reports generated by the system.
        - **Identify Trends:** Look for patterns or spikes in malicious URL detections, potentially indicating targeted attacks.
        - **Evaluate Effectiveness:** Track changes in FPR, FNR, and overall accuracy over time to assess the system's effectiveness and identify areas for improvement.
4. **Continuous Improvement:**
    - **Feedback Loop:** Integrate user feedback into the system.
        - **False Positives:** Allow users to report incorrectly blocked URLs to refine the model.
        - **Suspicious URLs:** Enable users to report encountered URLs that seem suspicious, potentially feeding them into the system for further analysis.
    - **Retraining:** Regularly retrain the machine learning model with fresh data to adapt to evolving threats and improve detection

capabilities.

By analyzing the results effectively, you can gain valuable insights into the system's performance, identify potential security risks, and ensure the system remains effective in protecting against malicious URLs.

7.2 SYSTEM TESTING

Testing a malicious URL detection system is crucial to ensure its effectiveness and reliability. Here's a breakdown of key testing approaches:

- **Functional Testing:**
  - Verify blacklist matching accuracy.
  - Assess machine learning model performance (accuracy, false positives/negatives, edge cases).
  - Test alerting and reporting functionality.
- **Non-Functional Testing:**
  - Evaluate performance (response time, resource usage) under load.
  - Assess scalability (handling increased URL/user volume).
  - Test usability (UI design, ease of use).
- **Security Testing:**
  - Simulate attacks (penetration testing) to identify vulnerabilities.
  - Verify secure system configuration.
- **Integration Testing:**
  - Test communication and functionality with other security systems.
  - Evaluate impact on overall security posture.
- **Additional Considerations:**
  - Diverse test data (malicious, benign, user-reported).
  - Continuous testing (regularly adapt to evolving threats).
  - Test automation for efficiency.
  - Document testing process, results, and identified issues.

CONCLUSION

- In this paper, black and white list technology and machine learning algorithms were used and formed multi layer filtering model for detection of malicious URLs. The model was trained for each machine learning algorithm i.e. naive Bayesian classification and decision tree classifier threshold and this threshold is used to refer to guide two classifiers for filtering URL. We combined the Naive Bayesian classifier, Decision Tree classifier and SVM classifiers in one multi-layer model to improve the malicious URL detection system in terms of accuracy. We observed from the real examples that multi-layer filtering models does effective detection of malicious URLs.

REFERENCES

1. **Is this URL Safe: Detection of Malicious URLs Using Glob Vector for Word Representation** (2022) by R. Bharadwaj, A. Bhatia, L. D. Chhibbar, K. Tiwari and A. Agrawal
2. **Malicious and Benign URL Dataset Generation Using Character-Level LSTM Models** (2022) by S. Vecile, K. Lacr K. Grolinger and J. Samarabandu
3. **Developing and Evaluating an Artificial Intelligence Model f Malicious URL Detection** (2022) by Nora A. A. and Narmatl C
4. **A Review on Malicious URLs Detection Using Machine Learning Methods** (2023) by Tasfia Tabassum, Md. Mahbubu Alam, Md. Sabbir Ejaz, and Mohammad Kamrul Hasan