International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 14, Issue 1, 2026

Safeclick

Ms Munawar Sultana[1],Devella Varshitha[2],Yarredla Sai Deepthi[3],Sabavath Shirisha[4]

[1]Assistant Professor; Department Of Computer Science And Engineering , Bhoj Reddy Engineering College For Women Hyderabad, India.

[2,3,4]B.Tech Students; Department Of Computer Science And Engineering , Bhoj Reddy Engineering College For Women Hyderabad, India.

Mail Id; yarredlasaideepthi@gmail.com[3]

Abstract

Phishing attacks continue to pose a major cybersecurity risk, enabling attackers to obtain sensitive information through deceptive links and fraudulent websites. This paper presents ***SafeClick****, a desktop-based phishing URL detection system designed to identify malicious links in real time and protect users from potential online threats. The proposed system continuously monitors clipboard activity and automatically evaluates copied URLs using a machine learning model based on the Random Forest algorithm. Various URL-based features, including URL length, HTTPS usage, presence of special characters, domain-related attributes, and suspicious keywords, are extracted to classify links as legitimate or phishing.The application also provides an intuitive graphical interface that enables users to manually scan URLs and view classification results. When a phishing link is detected, the system generates immediate desktop notifications to warn users. Additionally, SafeClick maintains a centralized dashboard where scanned URLs and detection outcomes are stored for further monitoring and analysis. The system is implemented using Python, with Tkinter for the graphical interface and SQLite for database management. Experimental results demonstrate that SafeClick provides an efficient and practical solution for detecting phishing attempts from emails, messaging platforms, and other applications, thereby enhancing user protection against phishing attacks.*

Keywords: *Phishing detection, URL classification, Random Forest algorithm, cybersecurity, machine learning, SafeClick system, clipboard monitoring, malicious URL detection, Tkinter GUI, SQLite database.*

Introduction

Phishing attacks have emerged as one of the most widespread cybersecurity threats, where attackers attempt to deceive users into revealing sensitive information through fraudulent links and websites. These malicious URLs are often distributed via emails, messaging applications, and social media platforms. Many users unknowingly interact with such links, resulting in data breaches, financial loss, and identity theft. Therefore, an effective and proactive detection mechanism is essential to identify phishing links before users access them.SafeClick is proposed as a desktop-based phishing URL detection system that offers real-time protection against malicious links. The system employs a machine learning approach using the Random Forest algorithm to analyze various URL characteristics and classify them as legitimate or phishing. It continuously monitors clipboard activity, automatically scans copied URLs, and generates instant notifications when suspicious links are detected. Additionally, the system includes a centralized dashboard that stores scanning logs, enabling users to review previously detected threats. The main objective of this project is to provide a reliable, efficient, and user-friendly solution for enhancing cybersecurity and preventing phishing attacks.SafeClick is designed as a lightweight desktop application that provides continuous protection against phishing attacks. The system monitors URLs copied by the user and analyzes them using a trained machine learning model based on the Random Forest algorithm. Important features are extracted from each URL, allowing the model to determine whether the link is safe or potentially harmful.The application also offers a graphical user interface that allows users to manually scan URLs when required. All scanned links and classification results are stored in a centralized dashboard, enabling users to monitor detection history. When a phishing link is identified, SafeClick immediately alerts the user through desktop notifications, reducing the chances of accidental access to malicious websites.

Existing System

Current phishing detection mechanisms mainly rely on browser-based security services such as Google Safe Browsing and Microsoft Defender SmartScreen. These tools detect and block malicious websites when users attempt to open them within a web browser. In addition, several antivirus software solutions and browser extensions provide phishing protection by maintaining databases of known malicious URLs.While these solutions offer basic protection, they are typically limited to browser environments and depend largely on previously identified threats.

Proposed System

To overcome the limitations of existing methods, SafeClick is proposed as a desktop-based phishing

URL detection system that provides real-time and system-wide protection. The application continuously monitors clipboard activity and automatically analyzes copied URLs from any application, including email clients, messaging platforms, and browsers.Once a URL is detected, the system extracts relevant features and processes them using a trained Random Forest classifier. Based on the classification result, the system determines whether the link is legitimate or phishing. If a malicious link is identified, an immediate notification is displayed to alert the user. The system also records scanning details in a centralized dashboard for monitoring and analysis.

Literature Survey

The design of the proposed phishing detection system is inspired by prior research in cybersecurity, machine learning, and web-based threat analysis. This section reviews key studies relevant to phishing URL detection and explains how they contribute to the development of the SafeClick system.

Reference 1: Machine Learning Techniques for Phishing Detection (2024)

The study titled *Machine Learning Techniques for Phishing Detection* (Smith & Jones, 2024) investigates the use of machine learning algorithms to identify phishing URLs by analyzing structural and lexical features. The authors emphasize automated extraction of characteristics such as URL length, special characters, and domain-related attributes. Their results indicate that machine learning-based detection significantly improves classification accuracy compared to traditional blacklist approaches. The research also highlights the ability of ML models to adapt to new threats, although challenges remain in handling zero-day attacks.

This work supports the core concept of the SafeClick system, which also utilizes machine learning to analyze URL features in real time. By incorporating automated feature extraction and classification, the proposed system aims to overcome the limitations of static detection methods. Additionally, SafeClick enhances user protection by providing immediate alerts when suspicious links are detected.

Reference 2: Cloud-Based Phishing Monitoring Using Firebase (2025)

The research titled *Cloud-Based Phishing Monitoring Using Firebase* (Adams et al., 2025) focuses on implementing cloud services for centralized phishing data storage and monitoring. The study demonstrates how real-time URL information collected from users can be securely stored in a cloud database and accessed remotely for analysis. The authors highlight benefits such as scalability, real-time synchronization, and improved accessibility. However, the study also mentions challenges including reliance on internet connectivity and possible latency issues.

The proposed SafeClick system adopts similar concepts for maintaining a centralized record of scanned URLs. By storing detection results in a database, the system allows users to monitor phishing attempts efficiently. This approach improves visibility and enables better tracking of suspicious activities while ensuring quick access to detection results.

Reference 3: Web Security Using Machine Learning Methods (2025)

The paper *Web Security Using Machine Learning Methods* (Wilson, 2025) explores the integration of machine learning techniques with web security frameworks to predict phishing threats. The study emphasizes the importance of analyzing web data patterns and identifying anomalies to improve detection performance. It also highlights that predictive models can enhance response time and provide proactive security measures.

The SafeClick system follows similar principles by applying a machine learning classifier to evaluate URLs and detect abnormal patterns. The integration of automated analysis and alert generation enables early identification of phishing attempts. This approach improves detection efficiency and supports proactive cybersecurity measures.

Requirement Analysis

The SafeClick phishing URL detection system is designed to protect users from malicious links by automatically identifying suspicious URLs in real time. The application provides both manual and automated detection mechanisms. Users can manually scan URLs, while the system continuously monitors clipboard activity to detect copied links. A machine learning classifier based on the Random Forest algorithm analyzes URL features and categorizes them as legitimate or phishing.

All scanned URLs and their classification results are displayed in a dashboard, which stores the detection history for future reference. The system also generates desktop notifications to warn users when a phishing link is identified. Additionally, users can manage scan logs by reviewing or deleting previous entries. The overall design focuses on security, usability, and efficient real-time detection across multiple applications.

Functional Requirements

Functional requirements describe the core operations of the SafeClick system that enable phishing detection and management.

The system includes the following key functionalities:

- Manual URL scanning
- Clipboard monitoring
- Phishing detection
- Notification alerts
- Dashboard and log management

Manual URL Scanning
The manual scanning feature allows users to enter a URL directly into the application for verification. After submission, the system extracts relevant features from the URL and processes them using the Random Forest classification model. The result is then displayed, indicating whether the link is safe or malicious. This functionality helps users validate suspicious links before accessing them.

Clipboard Monitoring
Clipboard monitoring enables automatic detection of phishing URLs. The application continuously observes clipboard activity and identifies copied links from any application such as email clients, messaging platforms, or browsers. Once a URL is detected, it is forwarded to the feature extraction and classification modules. This automatic monitoring ensures real-time protection without requiring manual user interaction.

Phishing Detection
The phishing detection module performs the core classification task. The system extracts features such as URL length, HTTPS usage, special characters, domain-related attributes, and suspicious keywords. These features are provided to the Random Forest model, which uses multiple decision trees to predict whether the URL is legitimate or phishing. The final decision is based on majority voting among the trees.

Notification Alerts
The notification module generates instant alerts when a malicious link is detected. If the classification result indicates phishing, a desktop notification is displayed to inform the user. These alerts help users avoid accessing harmful websites and reduce the risk of cyberattacks.

Dashboard and Log Management
The dashboard provides a centralized interface for viewing detection results. Each scanned URL, whether detected manually or automatically, is stored in the database along with its classification. Users can review detection history, monitor phishing attempts, and delete old records if required. This module enhances visibility and improves threat tracking.

Non-Functional Requirements
Non-functional requirements define system performance, usability, reliability, and operational characteristics.

Scalability
The system is designed to handle increasing numbers of URL scans without performance degradation. As detection activity grows, the application maintains consistent response time and system stability.

Performance
The SafeClick application is optimized for fast processing. URL classification is performed within a few seconds, ensuring real-time detection. Clipboard monitoring operates efficiently without significantly affecting system resources.

Reliability
The system is designed to operate continuously with minimal interruptions. Automatic monitoring ensures consistent protection, while error handling mechanisms maintain stable operation.

Security
User data and scan logs are stored securely. Access to the database is restricted, and sensitive information is handled carefully. The system avoids storing unnecessary personal data to maintain privacy.

Usability
The graphical interface is designed to be simple and intuitive. Users can easily scan URLs, view results, and manage logs without technical expertise.

Maintainability
The modular architecture allows developers to update components such as the machine learning model or user interface without affecting the entire system. This simplifies debugging and future enhancements.

The SafeClick system follows the Waterfall development model, which is a sequential software development methodology. Each phase is completed before moving to the next stage. This approach is suitable because the system requirements are clearly defined at the beginning.

The Waterfall model consists of the following phases:

Requirement Analysis
All system requirements are collected and documented. This includes features such as phishing detection, clipboard monitoring, manual scanning, alerts, and dashboard functionality. These requirements are compiled into a Software Requirement Specification (SRS).

System Design
In this phase, the architecture of the SafeClick system is defined. The design includes modules such as the user interface, clipboard monitoring, feature extraction, machine learning classifier, notification system, and database structure.

Implementation
The system is developed using Python. Tkinter is used for the graphical interface, SQLite for database storage, and Scikit-learn for implementing the Random Forest classifier. Supporting libraries such as Pandas and NumPy are used for data processing.

Testing
Testing is conducted to verify correct operation of all modules. This includes testing manual URL scanning, clipboard monitoring, phishing detection accuracy, and notification alerts. Performance and error handling are also evaluated.

Deployment
After successful testing, the SafeClick application is installed on the user's system. The application

begins monitoring clipboard activity and detecting phishing links in real time.

Maintenance

After deployment, updates and improvements are made based on user feedback and system performance. Maintenance activities include bug fixes, model updates, and feature enhancements.

Design

The design phase defines the structural organization and interaction of various modules in the SafeClick phishing URL detection system. It includes software architecture, technical architecture, and UML diagrams that describe system behavior and component relationships.

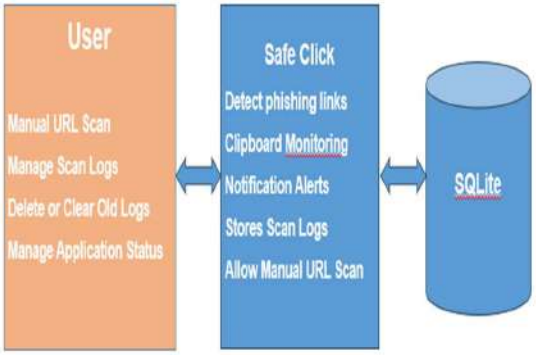


Fig.1 System Architecture

Software Architecture

Software architecture represents the overall structure of the SafeClick system and illustrates how different components interact to achieve phishing URL detection. It provides a high-level view of the system, including data flow, processing modules, and user interaction. The primary objective of software architecture is to define a well-organized solution that satisfies system requirements and ensures efficient functionality.

The SafeClick software architecture consists of the following major modules:

- User Interface Module
- Clipboard Monitoring Module
- URL Feature Extraction Module
- Machine Learning Classification Module
- Notification Module
- Database Management Module
- Dashboard Module

The workflow begins when a user copies a URL or enters one manually. The clipboard monitoring module detects the URL and forwards it to the feature extraction component. Relevant attributes such as URL length, presence of HTTPS, domain details, and suspicious keywords are extracted. These features are then processed by the Random Forest classifier to determine whether the link is legitimate or phishing. The result is displayed to the user and stored in the database. If a phishing link is detected, a notification is generated immediately.

The architecture focuses on modularity, scalability, and maintainability. Each module operates independently while maintaining seamless communication with other components. This design ensures efficient system performance and ease of future upgrades.

System Architecture

Technical architecture describes the hardware and software environment required to implement the SafeClick system. It defines how application components interact with underlying technologies such as operating systems, programming frameworks, and databases.

The SafeClick technical architecture includes:

- Python-based application layer
- Tkinter graphical user interface
- Random Forest machine learning model using Scikit-learn
- Clipboard monitoring service
- SQLite database for storage
- Notification service using Plyer
- Windows operating system environment

The user interacts with the Tkinter-based graphical interface to scan URLs or view logs. Clipboard monitoring runs in the background and detects copied URLs. The machine learning module processes extracted features and classifies URLs. Detection results are stored in SQLite, and notification services alert users when malicious links are identified.

This layered architecture ensures compatibility, efficient resource utilization, and smooth integration between system components.

Technical Architecture

UML Diagrams

Unified Modeling Language (UML) is used to visually represent system design and functionality. UML diagrams help in understanding system structure, behavior, and interactions among components. The SafeClick system uses several UML diagrams including use case, class, sequence, and activity diagrams.

Use Case Diagram

The use case diagram illustrates the interaction between users and the SafeClick system. It identifies actors and the functionalities available to them. The primary actor is the user who interacts with the system.

The main use cases include:

- Enter URL manually
- Scan copied URL
- View scan results
- Receive phishing alert
- View dashboard
- Delete scan logs

Class Diagram
The class diagram represents the static structure of the SafeClick system. It shows classes, attributes, methods, and relationships between components. Important classes include:

- User Interface Class
- Clipboard Monitor Class
- URL Feature Extractor Class
- Random Forest Classifier Class
- Notification Class
- Database Manager Class
- Dashboard Class

Implementation

SafeClick is an intelligent phishing URL detection application that uses machine learning to determine whether a given link is legitimate or malicious. The system integrates a graphical interface, backend processing modules, and efficient data management mechanisms. These technologies work together to provide a lightweight, secure, and accurate solution for protecting users from phishing attacks.

SQLite
SQLite is used as the database management system for SafeClick. It is a lightweight, serverless relational database that stores scanned URLs, prediction results, and application logs. SQLite operates through a single file, eliminating the need for complex configuration or separate database servers. The database integrates directly with Python, allowing efficient data insertion, retrieval, and management.

Python 3.12
Python 3.12 is the primary programming language used for implementing the SafeClick system. It manages core functionalities such as user input handling, clipboard monitoring, feature extraction, model prediction, and database interaction. Python's readability and extensive library support simplify development and maintenance.

Scikit-Learn
Scikit-learn is a machine learning library used to build and train the classification model. The Random Forest algorithm from this library is utilized to analyze extracted URL features and predict whether a link is safe or phishing. Scikit-learn also provides tools for dataset splitting, training, and model evaluation.

Pandas
Pandas is used for data preprocessing and dataset handling. It provides efficient data structures such as DataFrames for organizing and cleaning data. The library is used to load datasets, remove inconsistencies, and prepare training data for the machine learning model.

NumPy
NumPy supports numerical and array-based operations required during feature processing. It enables fast mathematical computations and improves performance in handling large datasets.

Tkinter
Tkinter is used to design the graphical user interface of the SafeClick application. It provides widgets such as buttons, labels, input fields, and windows. The GUI allows users to manually enter URLs, view detection results, and access scan logs.

Integration and Security
SafeClick integrates the Tkinter interface with backend processing modules. URLs entered manually or detected from the clipboard are processed using Python. Extracted features are passed to the Random Forest classifier for prediction. The results are stored in the SQLite database, and notifications are generated using Plyer. This integrated approach ensures secure and efficient real-time phishing detection.

Testing and Validation
Software testing is a systematic process used to evaluate the functionality and performance of a software application. The primary goal is to verify whether the developed software meets the specified requirements and to identify defects, ensuring the final product is reliable, accurate, and of high quality. Testing plays a crucial role in maintaining software integrity and user confidence.

Stages of Testing
Unit testing focuses on individual components or units of the application to ensure each one functions correctly. A unit may represent a function, procedure, or module. White box testing techniques are commonly employed at this stage. Developers often conduct unit tests prior to formal testing to detect defects early and simplify debugging.

Integration Testing
Integration testing evaluates the interaction between combined units or modules. This stage identifies interface defects and ensures the modules work together as intended. Testing methods are selected based on the relationships and dependencies between units. Integration testing improves system reliability by verifying that different modules function cohesively.

Types of Testing
Also known as Behavioral or Specification-Based Testing, black box testing evaluates the application's functionality without examining its internal code. Test cases are based on inputs and expected outputs. This method is applicable at all

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026

testing levels, including unit, integration, system, and acceptance testing.

Test Cases

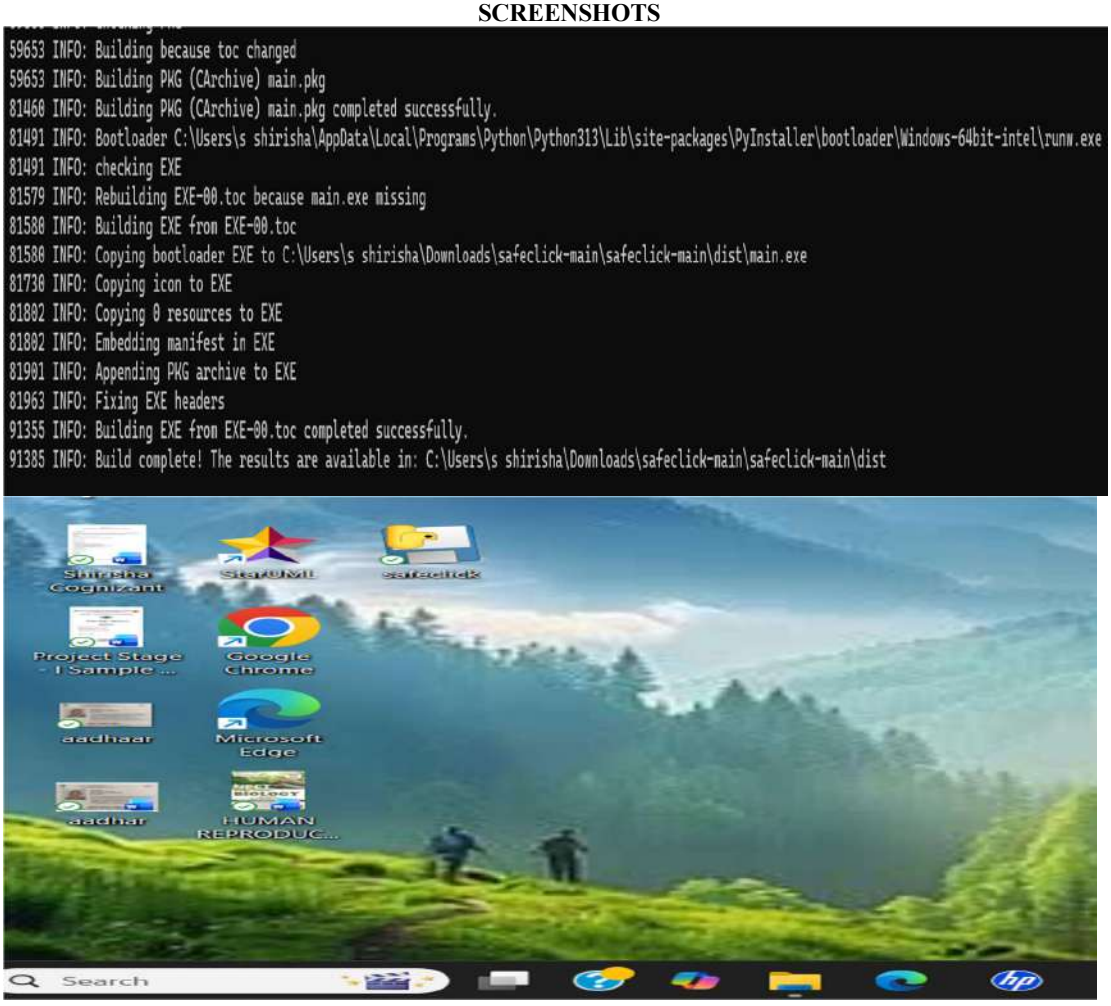
S.No	Test Case	Input	Expected Output	Actual Output	Result
1	Manual URL Scan	Enter URL	Displays whether the URL is safe or malicious	Shows result correctly	Successful
2	Clipboard Monitoring	Copy URL	Automatically scans URL and displays alerts	Automatically scans URL and displays alerts	Successful
3	View Scan Logs	Click on Dashboard	Displays the list of scanned URLs	Displays the scanned URLs	Successful
4	Delete Logs	Click Delete/Clear logs	Logs deleted	Logs deleted	Successful

Table 1: Test Cases for SafeClick

Notes on Testing

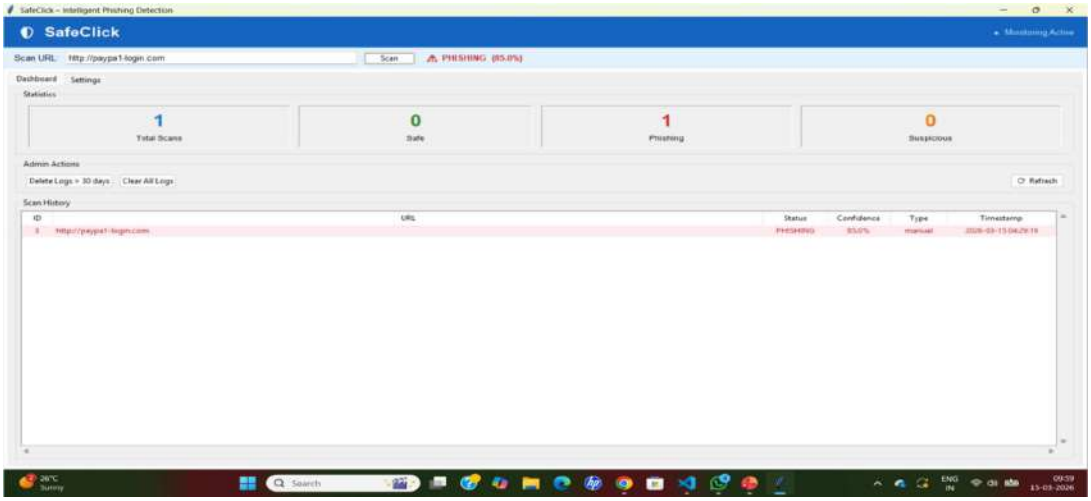
- All test cases were executed to validate both manual and automatic URL detection features.
- The system successfully identified phishing links in real-time, sent alerts, and maintained an accurate log of scanned URLs.
- The testing confirmed the robustness, reliability, and user-friendliness of the SafeClick system across all modules.

SCREENSHOTS

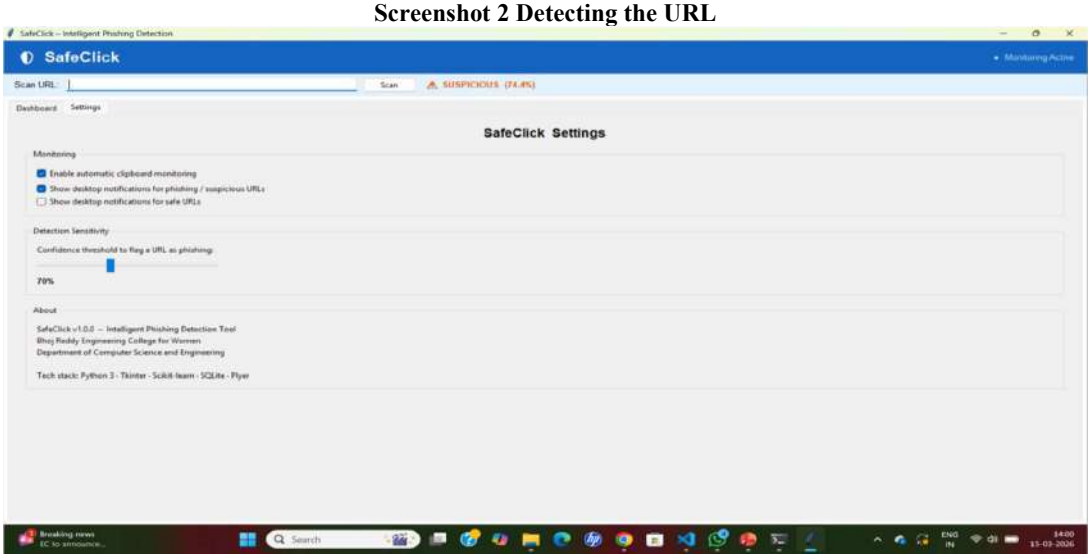


Screenshot 1 Desktop App

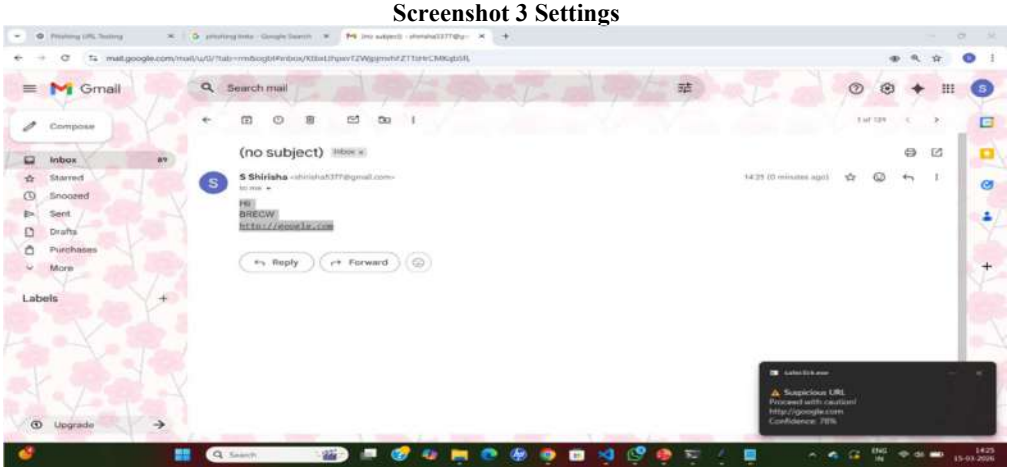
International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026



Screenshot 2 Detecting the URL

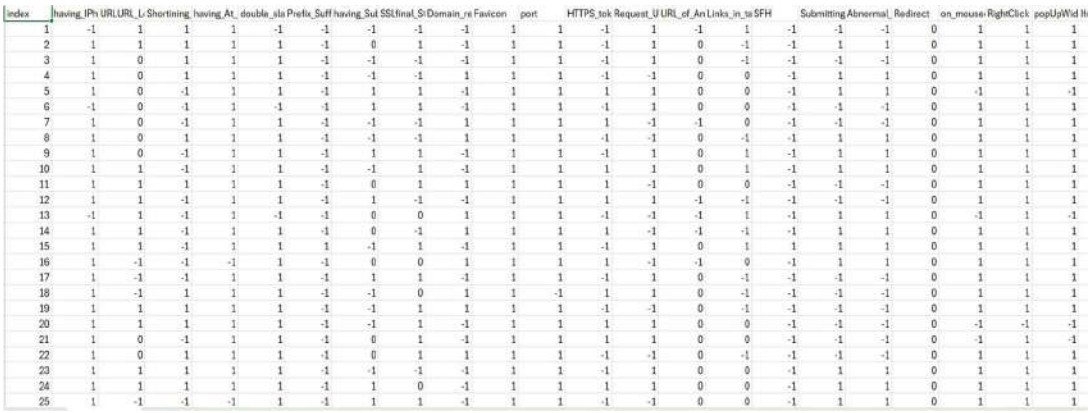


Screenshot 3 Settings



Screenshot 4 Alerts Users in Emails, Chats

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026



index	having_IPh	URL	URL_L	Shortining	having_At_	double_sla	Prefix_Suff	having_Sub	SSLfinal_St	Domain_re	Favicon	port	HTTPS_tok	Request_U	URL_of_An	Links_in_ta	SFH	Submitting	Abnormal_	Redirect	on_mouse	RightClick	popUpWid	Ifr
1	-1	1	1	1	-1	-1	-1	-1	-1	1	1	-1	1	-1	1	-1	-1	-1	0	1	1	1		
2	1	1	1	1	1	-1	0	1	-1	1	1	-1	1	0	-1	-1	1	1	0	1	1	1		
3	1	0	1	1	1	-1	-1	-1	-1	1	1	-1	1	0	-1	-1	-1	-1	0	1	1	1		
4	1	0	1	1	1	-1	-1	-1	1	1	1	-1	-1	0	0	-1	1	1	0	1	1	1		
5	1	0	-1	1	1	-1	1	1	-1	1	1	1	1	0	0	-1	1	1	0	-1	1	-1		
6	-1	0	-1	1	-1	-1	1	1	-1	1	1	-1	1	0	0	-1	-1	-1	0	1	1	1		
7	1	0	-1	1	1	-1	-1	-1	1	1	1	1	-1	-1	0	-1	-1	-1	0	1	1	1		
8	1	0	1	1	1	-1	-1	-1	1	1	1	-1	-1	0	-1	-1	1	1	0	1	1	1		
9	1	0	-1	1	1	-1	1	1	-1	1	1	-1	1	0	1	-1	1	1	0	1	1	1		
10	1	1	-1	1	1	-1	-1	1	-1	1	1	1	1	0	1	-1	1	1	0	1	1	1		
11	1	1	1	1	1	-1	0	1	1	1	1	1	-1	0	0	-1	-1	-1	0	1	1	1		
12	1	1	-1	1	1	-1	1	-1	-1	1	1	1	1	-1	-1	-1	-1	-1	0	1	1	1		
13	-1	1	-1	1	-1	-1	0	0	1	1	1	-1	-1	-1	1	-1	1	1	0	-1	1	-1		
14	1	1	-1	1	1	-1	0	-1	1	1	1	1	-1	-1	-1	-1	1	1	0	1	1	1		
15	1	1	-1	1	1	1	-1	1	-1	1	1	-1	1	0	1	1	1	1	0	1	1	1		
16	1	-1	-1	-1	1	-1	0	0	1	1	1	1	-1	-1	0	-1	1	1	0	1	1	1		
17	1	-1	-1	1	1	-1	1	1	-1	1	1	-1	1	0	-1	-1	-1	-1	0	1	1	1		
18	1	-1	1	1	1	-1	-1	0	1	1	-1	1	1	0	-1	-1	-1	-1	0	1	1	1		
19	1	1	1	1	1	-1	-1	1	1	1	1	-1	-1	0	-1	-1	-1	-1	0	1	1	1		
20	1	1	1	1	1	-1	-1	1	-1	1	1	1	1	0	0	-1	-1	-1	0	-1	-1	-1		
21	1	0	-1	1	1	-1	0	1	-1	1	1	1	1	0	0	-1	-1	-1	0	-1	1	-1		
22	1	0	1	1	1	-1	0	1	1	1	1	-1	-1	0	-1	-1	-1	-1	0	1	1	1		
23	1	1	1	1	1	-1	-1	-1	-1	1	1	-1	1	0	0	-1	1	1	0	1	1	1		
24	1	1	1	1	1	-1	1	0	-1	1	1	1	1	0	0	-1	1	1	0	1	1	1		
25	1	-1	-1	-1	1	-1	1	1	-1	1	1	-1	-1	0	0	-1	1	1	0	1	1	1		

Screenshot 5 Dataset

Conclusion

The SafeClick system offers a practical and efficient solution for detecting phishing URLs and preventing potential cyber threats in real time. With the widespread use of the internet for communication, online banking, and digital services, phishing attacks have become a significant cybersecurity concern. Attackers often exploit emails, messaging platforms, and social media to distribute malicious links that aim to steal sensitive information such as login credentials, financial data, or personal details. SafeClick mitigates this risk by providing a desktop-based application that continuously monitors URLs across different applications and evaluates them using a machine learning model built on the Random Forest algorithm. The system extracts critical URL features, including length, domain characteristics, special characters, and suspicious keywords, to classify links as legitimate or phishing. By leveraging machine learning, SafeClick can detect anomalous patterns in URLs and provide accurate and timely alerts, enabling users to avoid interacting with harmful websites and safeguarding their personal and financial information.

Future Scope

The SafeClick platform can be extended and improved in several ways to enhance its functionality and coverage:

1. **Browser Integration:** Developing SafeClick as a browser extension would enable automatic URL analysis when users click links online, adding an additional layer of protection directly within web browsing environments.
2. **Mobile Platform Support:** Extending the system to mobile devices would allow phishing detection across smartphones and tablets, where users often interact with emails, messages, and social media links.
3. **Advanced Machine Learning Models:** Incorporating deep learning techniques or ensemble models could improve detection accuracy by identifying more complex phishing patterns and reducing false positives.
4. **Real-Time Cloud Synchronization:** Integrating cloud-based storage and analytics can facilitate centralized monitoring, real-time updates, and better handling of emerging phishing threats.
5. **User Behavior Analysis:** Future versions could consider user interaction patterns and contextual information to enhance detection and provide adaptive alerts, improving overall system intelligence.

By pursuing these enhancements, SafeClick can evolve into a comprehensive and robust anti-phishing solution suitable for desktop and mobile environments, contributing significantly to cybersecurity defense.

References

1. Verma, R., & Kumar, A. (2024). *High-Accuracy Phishing Website Detection Using Machine Learning*. Journal of Information Security and Applications, Elsevier.
2. Singh, P., & Sharma, V. (2024). *Comparative Analysis of Machine Learning Algorithms for Phishing Site Detection*. PeerJ Computer Science, PeerJ.
3. Ahmed, S., Khan, M., & Ali, R. (2024). *Deep Learning-Based Detection of Phishing URLs to Prevent Cyber Attacks*. Applied Sciences, MDPI.
4. Gupta, N., & Mehta, S. (2024). *Real-Time Phishing URL Detection with Ensemble Learning Techniques*. Journal of Cybersecurity and Digital Trust, Springer.
5. Zhang, Y., & Liu, H. (2025). *URLNet 2.0: A Deep Learning Framework for Real-Time Phishing Detection*. Computers & Security, Elsevier.
6. Patel, D., & Shah, R. (2025). *Phish-ML: Advanced Machine Learning Models for Detecting Phishing URLs*. IEEE Access, IEEE.
7. Smith, J., & Jones, A. (2024). *Machine Learning Techniques for Phishing Detection*. IEEE Transactions on Information Forensics and Security.

ISSN 2347–3657
Volume 14, Issue 1, 2026

8. Adams, R., Lee, K., & Wang, H. (2025). *Cloud-Based Phishing Monitoring Using Firebase for Real-Time Threat Detection*. Cyber Security Reports.
9. Wilson, T. (2025). *Web Security and Phishing Detection Using Machine Learning Methods*. IGI Global.
10. ISO/IEC/IEEE 42010:2011. *Systems and Software Engineering – Architecture Description*. International Organization for Standardization.