International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

DETECTION OF MALICIOUS URL'S USING FLASK

Mr. E. Ranjith, M.C.A, M. Phil[1], Mrs. V. Manjula[2]

[1]Assistance professor, Krishnasamy College of Engineering &Technology
[2]Final Year MCA, Krishnasamy College of Engineering &Technology

ABSTRACT

Internet surfing has become a vital part of our daily life. So, to catch the attention of the users' different browser vendors compete to set up the new functionality and advanced features that become the source of attacks for the intruder and the websites are put at hazard. However, the existing approaches are not adequate to protect the surfers which require an expeditious and precise model that can be able to distinguish between the benign or malicious WebPages. In this research article, we design a new classification system to analyse and detect the malicious web pages using machine learning classifiers and Some special URL (Uniform Resource Locator) based on extricated features the classifiers are trained to predict the malicious web pages. The experimental results have shown that the performance of the random forest classifier achieves better accuracy in comparison to other machine learning classifiers.

1. INTRODUCTION

The rapid development of the web, more and more services like internet banking, e-commerce, social networking, shopping, making a bill payment, e-learning, etc. are available to users and they are surfing the internet via browsers or web application. As the browsers are come up with different advanced features and functionalities which leads to risk by losing their personal and sensitive information. As the naïve users are not aware of the different malware so they are easily trapped by the intruder by just a single click on the malicious web sites which allows the invaders to detect the vulnerabilities on the web page and inject the payloads to get remote access to victim's web page. Therefore, the precise identification of web pages in an ever-growing web environment is very important. Blacklisting services were embedded in the browsers to face the challenges but it has several disadvantages like incorrect listing. In this article, we explore a self-learning approach to classify the web page based on a small feature set. We use four machine learning classifiers to classify the web site into two classes benign and malicious web pages. "The rest of the research work is planned as follows: Section II presents related work, the methodology is discussed in section III, experimental result analysis is depicted in Section IV and Section V contains the conclusion of the research work and suggests some future work".

2. EXISTING SYSTEM

In Existing System, there have been many filtering mechanisms to detect the malicious URLs. Some of them are Black-Listing, Heuristic Classification etc. word matching and URL syntax matching. Therefore, these conventional mechanisms cannot effectively deal with the ever evolving technologies and web-access techniques.

3. PROPOSED SYSTEM

We propose a novel classification method to address the challenges faced by the traditional mechanisms in malicious URL detection. The proposed classification model is built on sophisticated flask methods that not only takes care about the syntactical nature of the URL, but also the semantic and lexical meaning of these dynamically changing URLs. The proposed approach is expected to outperform the existing techniques.

4. LITERATURE STUDY

A. Context-sensitive and keyword density-based supervised machine learning techniques for malicious webpage detection

The Conventional malicious webpage detection methods use blacklists in order to decide whether a webpage is malicious or not. The blacklists are generally maintained by third-party organizations. However, keeping a list of all malicious Web sites and updating this list regularly is not an easy task for the frequently changing and rapidly growing number of webpages on the web. In this study, we propose a novel context-sensitive and keyword density-based method for the classification of webpages by using three supervised machine learning techniques, support vector machine, maximum entropy, and extreme learning machine. Features (words) of webpages are obtained from HTML contents and information is extracted by using feature extraction methods: existence of words, keyword frequencies, and keyword density techniques. The performance of proposed machine learning models is evaluated by using a benchmark data set which consists of one hundred thousand webpages.

B. Detection of malicious web pages based on hybrid analysis

Malicious web pages have become an increasingly serious threat to web security in recent years. In this paper, we propose a new detection method that consists of static and dynamic analyses for detecting malicious web pages. Static analysis utilizes classification algorithms in machine learning to identify certain benign and malicious web pages. As a complement to static analysis, dynamic analysis mainly checks the unknown web pages to determine whether they have malicious shellcodes during their execution. Because of the combination of static and dynamic analyses, the proposed detection method achieves high performance, and it has a light weight.

C. ML-and YARA-based malicious webpage detection

Attackers use the openness of the Internet to facilitate the dissemination of malware. Their attempts to infect target systems via the Web have increased with time and are unlikely to abate. In response to this threat, we present an automated, low-interaction malicious webpage detector, WebMon, that identifies invasive roots in Web resources loaded from WebKit2-based browsers using machine learning and YARA signatures. WebMon effectively detects hidden exploit codes by tracing linked URLs to confirm whether the relevant websites are malicious. WebMon detects a variety of attacks by running 250 containers simultaneously. In this configuration, the proposed model yields a detection rate of 98%, and is 7.6 times faster (with a container) than previously proposed models.

5. MODULES LIST AND DESCRIPTION

- MODULE 1: Dataset collection
- MODULE 2: Data preprocessing
- MODULE 3: detection
- MODULE 4: identification of URL

MODULE 1: Dataset collection

Datasets are collected from Kaggle website. That dataset includes websites URL, that have both malicious and benign URL.

- Url length

MODULE 2: Data preprocessing

The sklearn .preprocessing package provides several common utility functions and transformer classes to change raw feature vectors into a representation that is more suitable for the downstream estimators. In general, learning algorithms benefit from standardization of the data set. If some outliers are present in the set, robust scalars or transformers are more appropriate. The behaviors of the different scalers, transformers, and normalizers on a dataset containing marginal outliers are highlighted in Compare the effect of different scalers on data with outliers.

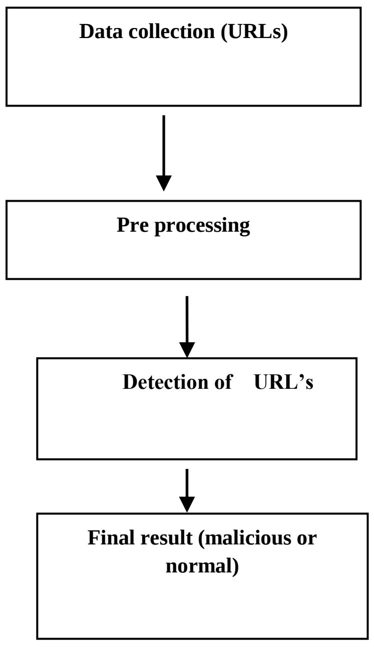
Module 3: detection

Detecting Phishing Domains is a classification problem, so it means we need labeled data which has samples as phish domains and legitimate domains in the training phase. The dataset which will be used in the training phase is a very important point to build successful detection mechanism. We have to use samples whose classes are precisely known. So, it means, the samples which are labeled as phishing must be absolutely detected as phishing. Likewise, the samples which are labeled as legitimate must be absolutely detected as legitimate. Otherwise, the system will not work correctly if we use samples that we are not sure about. Using flask model, we will detect the URLs

Module 4: identification of URL

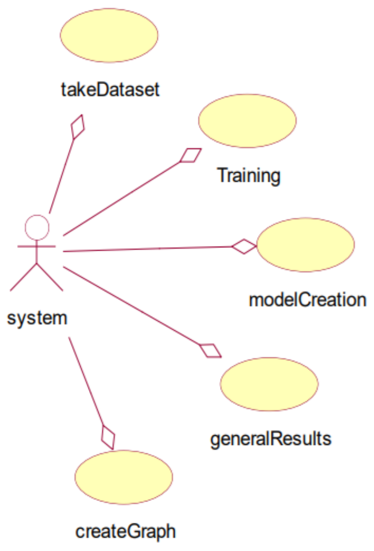
After giving the inputs of URL we need load the webpage URL then it will predict the output as whether the given input is malicious or Normal.

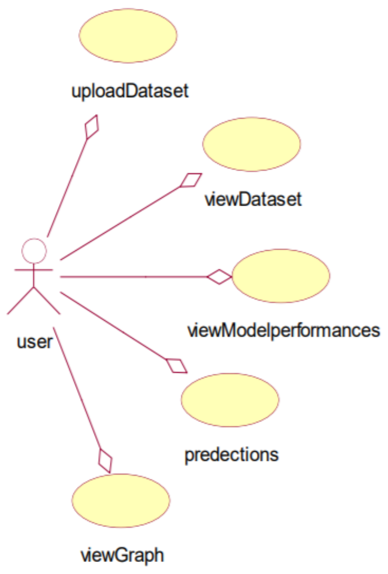
DATA FLOW DIAGRAM:



USE CASE DIAGRAM:

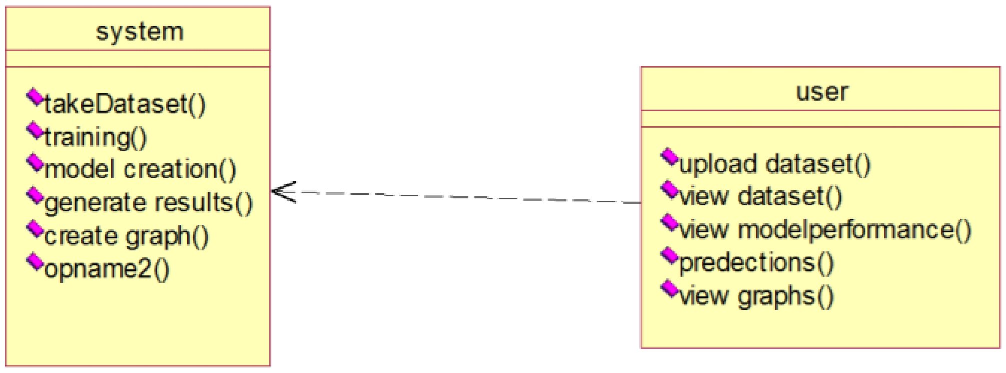
A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.





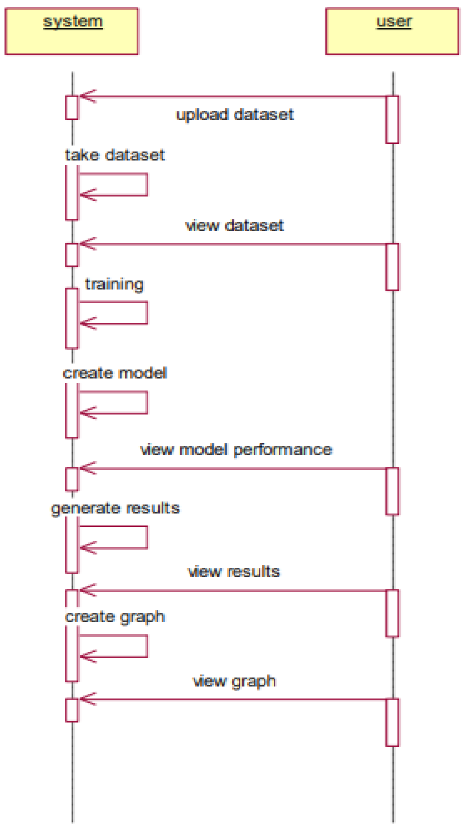
CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information



SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



DEPLOYMENT DIAGRAM:

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware's used to deploy the application.

system
user

6. SOFTWARE TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub – assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

TYPES OF TESTS

1. UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

2. INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

3. FUNCTIONAL TEST

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised

Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

4. SYSTEM TEST

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

5. WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

6. BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

7. UNIT TESTING

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach:

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

7. FUTURE SCOPE

PYTHON:

Python is a dynamic, high level, free open source and interpreted programming language. It supports object-oriented programming as well as procedural oriented programming. In Python, we don’t need to declare the type of variable because it is a dynamically typed language. Easy to code

Features in Python

There are many features in Python, some of which are discussed below

- Object-Oriented Language
- GUI Programming
- High-Level Language

ANACONDA:

Anaconda distribution comes with over 250 packages automatically installed, and over 7,500 additional open-source packages can be installed from PyPI as well as the conda package and virtual environment manager. It also includes a GUI, Anaconda Navigator,[12] as a graphical alternative to the command line interface (CLI). The big difference between conda and the pip package manager is in how package dependencies are managed, which is a significant challenge for Python data science and the reason conda exists.

Anaconda Navigator

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, macOS and Linux.

The following applications are available by default in Navigator:[16]

- JupyterLab
- Jupyter Notebook
- Spyder

- Visual Studio Code

8. CONCLUSION

Malicious web page identification is an emerging topic in cybersecurity. Though several research studies have been performed relating to the issues of malicious web page detection these are very costly as they consume more time and resources. In this research article, we employed a new web site classification system based on URL features to predict the web pages as malicious or benign using machine learning algorithms. The machine learning classifiers Random Forest (RF) achieves a higher accuracy of 95%. The experimental results have shown that our method can perform effectively for detecting the malicious web page

REFERENCES

[1] Tao, Wang, Yu Shunzheng, and Xie Bailin. "A novel framework for learning to detect malicious web pages." In 2010 International Forum on Information Technology and Applications, vol. 2, pp. 353-357. Ieee, 2010.

[2] Eshete, Birhanu, Adolfo Villafiorita, and Komminist Weldemariam. "Malicious website detection: Effectiveness and efficiency issues." In 2011 First SysSec Workshop, pp. 123-126. IEEE, 2011.

[3] Aldwairi, Monther, and Rami Alsalman. "Malurls: A lightweight malicious website classification based on url features." Journal of Emerging Technologies in Web Intelligence 4, no. 2 (2012): 128-133.

[4] Yoo, Suyeon, Sehun Kim, Anil Choudhary, O. P. Roy, and T. Tuithung. "Two-phase malicious web page detection scheme using misuse and anomaly detection." International Journal of Reliable Information and Assurance 2, no. 1 (2014): 1-9.

[5] Hwang, Young Sup, Jin Baek Kwon, Jae Chan Moon, and Seong Je Cho. "Classifying malicious web pages by using an adaptive support vector machine." Journal of Information Processing Systems 9, no. 3 (2013): 395-404.

[6] Yue, Tao, Jianhua Sun, and Hao Chen. "Fine-grained mining and classification of malicious Web pages." In 2013 Fourth International Conference on Digital Manufacturing & Automation, pp. 616-619. IEEE, 2013.

[7] Krishnaveni, S., and K. Sathiyakumari. "SpiderNet: An interaction tool for predicting malicious web pages." In International Conference on Information Communication and Embedded Systems (ICICES2014), pp. 1-6. IEEE, 2014.

[8] Sun, Bo, Mitsuaki Akiyama, Takeshi Yagi, Mitsuhiro Hatada, and Tatsuya Mori. "Automating URL blacklist generation with similarity search approach." IEICE TRANSACTIONS on Information and Systems 99, no. 4 (2016): 873- 882.

[9] Proceedings of the International Conference on Intelligent Computing and Control Systems (ICICCS 2020) IEEE Xplore Part Number:CFP20K74-ART; ISBN: 978-1-7281-4876-2.

[10] Urcuqui, Christian, Andres Navarro, Jose Osorio, and Melisa García. "Machine Learning Classifiers to Detect Malicious Websites." In SSN, pp. 14- 17. 2017.).

[11] Wang, Rong, Yan Zhu, Jiefan Tan, and Binbin Zhou. "Detection of malicious web pages based on hybrid analysis." Journal of Information Security and Applications 35 (2017): 68-74.74.

[12] Kim, Sungjin, Jinkook Kim, Seokwoo Nam, and Dohoon Kim. "WebMon: ML-and YARA-based malicious webpage detection." Computer Networks 137 (2018): 119-131.

[13] Altay, Betul, Tansel Dokeroglu, and Ahmet Cosar. "Context-sensitive and keyword density-based supervised machine learning techniques for malicious webpage detection." Soft Computing 23, no. 12 (2019): 4177-4191.