Malicious URL Detection Using Machine Learning

Prof. Mohammad Sharique
Sandip Institute of Technology and Research Center
Savitribai Phule Pune University
Nashik, Maharashtra, India

Swaroop Shinde
Sandip Institute of Technology and Research Center
Savitribai Phule Pune University
Nashik, Maharashtra, India

Samarth Kulkarni
Sandip Institute of Technology and Research Center
Savitribai Phule Pune University
Nashik, Maharashtra, India

Vaibhav Borse
Sandip Institute of Technology and Research Center
Savitribai Phule Pune University
Nashik, Maharashtra, India

Rushikesh Shewale
Sandip Institute of Technology and Research Center
Savitribai Phule Pune University
Nashik, Maharashtra, India

Abstract – **Recently, with the increase in Internet usage, cybersecurity had been a significant challenge for computer systems Different malicious URLs emit different malicious software and try to capture user information.**

Signature-based approaches have often been used to detect such websites and detected malicious URLs have been attempted to restrict access by using various security components. This chapter proposes using host-based and lexical features of the associated URLs to better improve the performance of classifiers for detecting malicious web sites. Random forest models and gradient boosting classifier are applied to create a URL classifier using URL string attributes as features. The highest accuracy was achieved by random forest as 98.6%. The results show that being able to identify malicious websites based on URL alone and classify them as spam URLs without relying on page content will result in significant resource savings as well as safe browsing experience for the user.

1. INTRODUCTION

The significance of the World Wide Web (WWW) has attracted increasing attention because of the growth and promotion of social networking, online banking, and ecommerce. While new development in communication technologies promote new e-commerce opportunities, it causes new opportunities for attackers as well. Nowadays, on the Internet, millions of such websites are commonly referred to as malicious web sites. It was noted that the technological advancements caused some techniques to attack and scam users such as spam SMS in social networks, online gambling, phishing, financial fraud, fraudulent prize-winning, and fake TV shopping (Jeong, Lee, Park, & Kim, 2017).

In recent years, most attacking methods are applied by spreading compromised URLs and fishing, and malicious Uniform Resource Locators (URLs) addresses are the leading methods used by hackers to perform malicious activities. Common types of attacks using malicious URLs can be categorized into Spam, Drive-by Download, Social Engineering, and Phishing (Kim, Jeong, Kim, & So, 2011). Spam is called to be sent to unsolicited messages by force for advertising or phishing, which we do not request and do not want to receive. These attacks have caused a tremendous amount of damage (Verma, Crane, & Gnawali, 2018). The download of malware while visiting a URL is called as Driveby download (Cova, Kruegel, & Vigna, 2010).

Lastly, Social Engineering and Phishing attacks guide users to reveal sensitive and private information by acting as genuine web pages (Heartfield & Loukas, 2015). The attackers create copies of the popular web pages used by users such as Facebook and Google and compromise victim computers by placing various pieces of malicious code in the manipulated web site’s HTML code. Besides, the ubiquitous use of smartphones encourages the increase of mobile and Quick Response (QR) code phishing activities, especially to deceive the elderly that encode fake URLs in QR codes. The dark side of the Internet has attracted increasing attention and bedeviled the world (Patil & Patil, 2015). Internet security software cannot always detect malware from malicious websites and drive-by downloads. It can, however, prevent you from getting them in the first place (Symantec, 2020). Malicious URLs detection is not adequately addressed yet and causes enormouslosses each year.

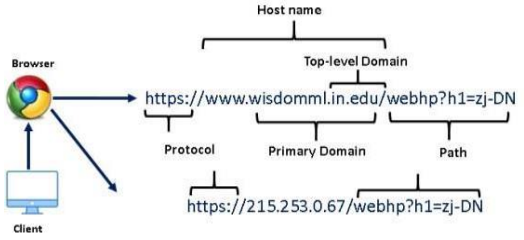
In the fourth quarter of 2019, more than 162,000 unique phishing URLs were detected globally (Statista, 2020). Even though the security components used today are trying to detect such malicious sites and web addresses, these components are evading by using different methods implemented by the attackers. Researchers have studied to gather effective solutions for Malicious URL Detection. One of the most popular ways is the blacklist method that uses records of known malicious URLs to filter the incoming URLs.

However, blacklists have some limitations, and this approach useless for new malicious sites that are created continuously. Security components have started to use innovative applications of machine learning and artificial intelligencebased prediction models to cope with this problem, during the last decades (Garera, Provos, Chew, & Rubin,

2007) (Kuyama & Kakizaki, 2016) (Ma, Saul, Savage, & Voelker, Beyond blacklists: learning to detect malicious web sites from suspicious URLs, 2009) (Ma, Saul, Savage, & Voelker., Learning to Detect Malicious URLs, 2011). They have started to prefer machine learning and artificial intelligence prediction instead of being signature-based for Malicious URL Detection. Machine Learning approaches apply a set of URLs as training data and learn a prediction function to classify whether a URL is malicious or benign. 2 This approach allows them to generalize to new URLs, unlike blacklisting methods. Soon, these solutions will need to be used in Cyber-Physical Systems (CPS), and the other area will be to identify harmful sites and URL addresses. As a result, it can be noted that Artificial Intelligencebased antimalware tools will aid to detect recent malware attacks and develop scanning engines.

The Uniform Resource Locator (URL) is the well-defined structured format unique address for accessing websites over World Wide Web (WWW).Generally, there are three basic components that make up a legitimate URL

i) Protocol: It is basically an identifier that determines what protocol to use e.g., HTTP, HTTPS, etc. ii) Hostname: Also known as the resource name. It contains the IP address or the domain name where the actual resource is located.
iii) Path: It specifies the actual path where the resource is located.


Host name
Browser
Top-level Domain
https://www.wisdomml.in.edu/webhp?h1=zj-DN
Protocol
Primary Domain
Path
https://215.253.0.67/webhp?h1=zj-DN
Client

2. LITERATURE SURVERY

1) Previous work on this topic has involved content analysis of the page itself (Ntoulas, Najork, Manasse, & Fetterly, 2006). These typically include creating features from the HTML structure of the word-length, and the number of words in the title.

2) Other methods involve looking at the amount and percentage of hidden content (not visible to a user) on a page. Another approach is first to determine what are important features in terms of ranking in a search engine and then find which features are likely to be used by spammers Egele, Kolbitsch, & Platzer, 2009. The downside to this approach is that it is infeasible to enumerate every ranking element, and thus important features may be missed.

3) Another work attempt to classify web spam into buckets, such as link spam, redirection, cloaking, and keyword stuffing (Gyongyi & Garcia-Molina, 2005). While splitting spam into more specific buckets will likely lead to improvements in classifier ability, this paper will focus on building a general classifier for all types of spam. While relying on the page content and links increase the amount of data available for spam classification, there are strong motivations for being able to classify spam before crawling a page. This paper explores using the URL string as the primary feature in spam classification.

4) Gupta and Singhal examine that the RF tree achieves an excellent result to detect Phishing URLs in minimum execution time. Firdausi et al. analyzed malware and benign files by collecting 250 unique benign and 220 individual malware software samples to train Support Vector Machine (SVM), Multilayer Perceptron (MLP) neural network, kNearest Neighbor, Naive Bayes and J48 decision tree on their dataset. They gained the highest accuracy of 96.8% by the J48 decision tree.

5)Rieck et al. gathered 10, 072 unique samples and divided them into 14 malware families to train Support Vector Machine and achieved 88% accuracy in testing correct malware. Ucci et al. present a model that applied machine learning algorithms to feature types extracted from Portable Executable files.

6) In the literature, Logistic Regression has attracted increasing attention for Malicious URL Detection . proposed a model that applied Naive Bayes for Malicious URL Detection. The extreme Learning Machines (ELM) approach is used for classifying the phishing websites

7) Singh et al. propose a method for malware detection by applying the Support Vector Machine.

8) Kazemian et al. have used machine learning techniques such as K-Nearest Neighbor, Support Vector Machines, Naive Bayes Classifier, and K Means rather than traditional methods of detecting whether they exist in a predetermined blacklist for detecting harmful webpages. For the tests, a data set consisting of 176 harmful samples and 965 harmless samples collected from Stop Badware site were used. In tests performed using the Decision Tree, Naive Bayes, Support Vector Machines, and AdaBoost Decision Tree classifiers, it was determined that the best result belonged to the AdaBoost Decision Tree classifier with a rate of 96.14%.

9) Main aim to detect harmful web pages by using the features of the web page based on URL-based properties, server information, and the content of the web page. Support vector machines and Naive Bayes classifiers and web pages created for phishing and malware distribution were determined with high accuracy

3. MOTIVATION

In 2020, the COVID-19 pandemic caused most office work to be shifted to remote platforms through the internet. Malicious URLs are being used by cybercriminals to take advantage of this situation. URLs can contain malware and spyware. Spam emails can also be used to deceive users into clicking on malicious URLs. Some URLs may be authentic while others are used for phishing and spam attacks. ML is one of the most rapidly expanding and effective areas of technology in the modern world. With the use of existing data, analysis using ML can help identify future outcomes. This provides opportunities to predict important things like the weather and game results. As the use of technology grows, it is more important to protect it as it is connected to our livelihoods.

With the power of prediction and connection, we can better combat threats to health and security.

4. PROBLEM DEFINATION

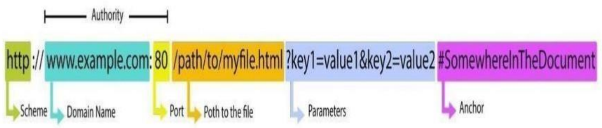
Web Security has become very important in recent years as internet connectivity has penetrated more and more regions across the world. While this penetration is great for global connectivity, it also means that more people have access to websites that can potentially attack them using malwares, viruses, and other malicious agents. Thus, it becomes more important than ever to identify and deal with such websites before a normal user has access to them (Jang-Jaccard and Nepal, 2014). Current approaches to deal with this problem have many limitations in terms of effectiveness and efficiency (Eshete, Villafiorita and Weldemariam, 2011). The aim of this study is to detect malicious websites using a group of machine learning algorithms called classifiers, we will try to detect malware on websites. This will help in safe web surfing and better user experience. By timely reporting malicious websites, the users will be able to avoid any violation and serious privacy breach. The users will also be able to avoid any illegal activities that they can get involved in. Labelling malicious websites will also help to eliminating fraud, as users become victim of attacks that use blackmailing and false information to get monetary advantage of their victim. For example, ransomware attacks are getting quite common.

Systems get infected by such viruses through surfing malicious websites.

5. URL CLASSIFICATION AND MALICIOUS ATTACKS

A URL is known as a specific unique resource on the Internet . URLs are associated with resources such as HTML pages, CSS documents, and images. There are a few exceptions where resources either do not exist or have been moved from the servers. 2.1 URL CLASSFICATION A URL identifies and locates a web resource. The type of protocol, source domain, top, second, and third-level domains, primary domains, and pathways are the components that comprise a URL. The complexity of a URL is determined by the resource being referenced, as well as how and where it is located.

URLs are used to gain access to the worldwide web.


Authority
http :// www.example.com: 80 /path/to/myfile.html ?key1=value1&key2=value2 #SomewhereInTheDocument
Scheme
Domain Name
Port
Poth to the file
Parameters
Anchor

5.1) Scheme

Scheme is the first component of a URL. It specifies the protocol that the browser utilizes for resource requests. A protocol is a set of instructions for data exchange or transfer over the internet . Hypertext Transfer Protocol (HTTP) and Hypertext Transfer Protocol Secure (HTTPS) are the most common protocols for webpages. HTTP is an unsecured version and HTTPS is a secured version. Browsers can also employ other schemes, for example mailto: to launch a mail client.

5.2) Authority

The authority component is followed by the pattern: //. When the domain, e.g. www.example.com and port, e.g. 80 exist, the authority divides them with a colon (:). The domain identifies the server being accessed. This is usually a domain name but can also be an IP address. 2.1.3 Path A path locates physical files on the server through its location.

5.3) Parameters

Parameters are extra values in a URL. The symbol & is used to separate key and value pairs. The server can utilize parameters to do further tasks. Every server is unique in terms of the rules to handle parameters.

5.4) Anchor

The anchor is the last component of the URL. It is a link to a different section of the document. An anchor acts as a bookmark within the resource, instructing the browser to display the content at the bookmarked location.

5.5) SPAM URLS

Spam URLs can spread through a variety of channels including emails, texts, and social media platforms. Social media is an easy and common channel for spammers and fraudsters. For a successful attack, personal information is often required, and it is easier to collect such information using these channels. This could make it more likely for a userto click on an unknown URL.

5.6) URL Shortening

URL shortening facilities, such as Bitly, Google URL shortener, Is Good, and TinyURL, are popular spam masking methods . For link sharing, an attacker may create many short versions of a long URL. Spam attackers use URL shortening to hide the true landing page of a malicious URL. Shortening a URL is a common approach, however social media platforms rarely detect and block them.

5.7) MALWARE URLS

A malware URL is a link that can take a user to a false webpage or website. The goal of creating malicious webpages is to carry out an attack agenda, which can be a political agenda, or to steal personal or organizational data. Actions such as simply clicking on a malicious URL, trying 6 to open an attached file, or trying to engage with an advertisement can have significant effects. Opening a malicious URL may download the payload to the machine. The payload contains malicious code which can harm the computer and compromise the data.

5.8) PHISHING URLS

Phishing is a type of social engineering attack that seeks to trick people into giving up personal information. Attackers focus on user personal details such as bank information, corporate data, login credentials, and anything valuable. Due to a lack of security awareness, organizations can have vulnerabilities. Attackers can find vulnerable people to infiltrate organizations using phishing attacks. One successful phishing attack on an employee can put an entire corporation in jeopardy. A solution to this problem is to effectively train users to identify malicious webpages.

6. BLACKLISTING AND HEURISTIC TECHNIQUES

In general, there are two approaches for classifying URLs, blacklisting and heuristic techniques. These methods rely on database lookup to allow or restrict good or bad URLs, respectively. A large database of blacklisted URLs is maintained which is acquired from trustworthy sources. As a result, when a new URL is added to the list, the utility software checks the database to see if it is in the list. If the URL matches one in the list, the user will be notified of a potential threat, otherwise 7 it will be regarded as nonmalicious or benign. These traditional methods take significant time, and it is hard to keep track of the URLs, especially the ones which have been shortened.

7. MACHINE LEARNING TECHNIQUES

Machine Learning (ML) techniques learn URL patterns using information gathered in a variety of ways. Feature extraction methods can be static or dynamic. Static features are typically collected from graphical images of webpages, URL strings, and scripting languages such as HTML and JavaScript Dynamic feature extraction is done by monitoring the dynamic behaviour of the system for anomalous activity. This is accomplished by looking for unusual or abnormal behaviour in the system logs and sequence calls. Because the systems are vulnerable to attacks, dynamic feature extraction methods are difficult to generalize and implement. ML techniques can be used to solve this problem.

8. DATASET

7 In this case study, we will be using a Malicious URLs dataset of 6,51,191 URLs, out of which 4,28,103 benign or safe URLs, 96,457 defacement URLs, 94,111 phishing URLs, and 32,520 malware URLs.

9. RINCIPAL COMPONENT ANALYSIS

Principal Component Analysis (PCA) is used to reduce the dimensionality of features in a dataset. It is commonly used to convert a large collection of variables into a smaller dataset. ML models are more efficient at exploring and visualizing smaller datasets with extraneous features removed [. In this project, the WEKA tool is first used to standardize the dataset features. Then the correlation matrix is obtained to determine the relationship between the features. Eigen decomposition is then used to obtain the eigenvectors and eigenvalues. The eigenvalues are the variances of the components, whereas the eigenvectors are the principal components. They are then sorted in descending order so the eigenvector with the highest eigenvalue is the first principal component of the dataset.The less important components with smaller eigenvalues are eliminated.

10. MACHINE LEARNING

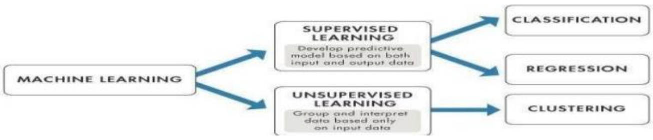
Machine Learning (ML) is an area of Artificial Intelligence (AI). ML is a data analysis technique that automatically forms analytical models. These models can learn from data, recognize patterns, and make decisions with minimal human intervention. The most important task in ML is feature selection. As ML algorithms are developed based on the results of training data, they are non-interactive so previous observations are used to make predictions. Accurate prediction can be a challenging task. In this work, six ML classifiers are employed for malicious URL detection. ML classifiers can be divided into two categories, namely supervised learning and unsupervised learning as described below,

10.1) SUPERVISED LEARNING

Supervised ML is used with labelled datasets. This data is used to train the model and predict the outcome. The outcome is usually a class or value. Supervised learning can solve a variety of complex problems, for example identifying and classifying viruses or spam emails in an inbox. Random Forest (RF), Logistic Regression, Neural Networks, Linear Regression, Naive Bayes, Support Vector Machine (SVM), are examples of supervised ML classifiers.

10.2) UNSUPERVISED LEARNING

Unsupervised ML algorithms are used with unlabelled datasets. Hidden patterns can be detected by these algorithms with no human interference. Due to their ability to identify differences and resemblances in data, they are commonly used in data analysis, product selling strategies, pattern recognition, and customer segmentation. Unsupervised learning is also used for feature extraction via dimensionality reduction.
Unsupervised ML algorithms include K-means clustering and probabilistic clustering.


MACHINE LEARNING
SUPERVISED LEARNING
Develop predictive model based on both input and output data
UNSUPERVISED LEARNING
Group and interpret data based only on input data
CLASSIFICATION
REGRESSION
CLUSTERING

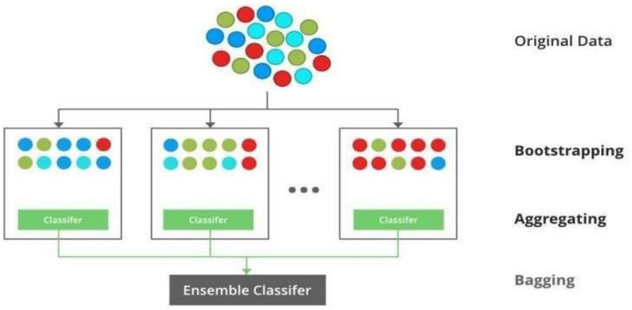
11. MACHINE LEARNING CLASSIFIERS

11.1) XGBoost

XGBoost is an implementation of Gradient Boosted decision trees. XGBoost models majorly dominate in many Kaggle

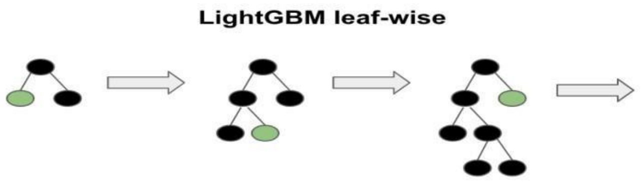
Competitions.In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree.

These individual classifiers/predictors then ensemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined prediction problems.



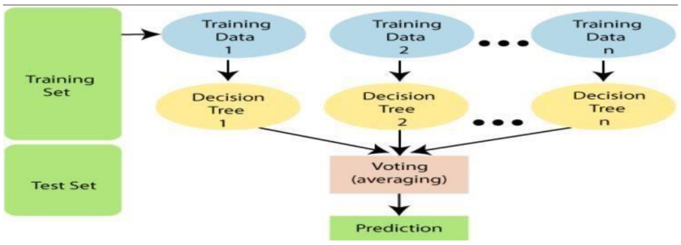
11.2) Light GBM

LightGBM is a gradient boosting framework based on decision trees to increases the efficiency of the model and reduces memory usage. It uses two novel techniques: Gradient-based One Side Sampling and Exclusive Feature Bundling (EFB) which fulfills the limitations of histogrambased algorithm that is primarily used in all GBDT (Gradient Boosting Decision Tree) frameworks. The two techniques of GOSS and EFB described below form the characteristics of LightGBM Algorithm. They comprise together to make the model work efficiently and provide it a cutting edge over other GBDT frameworks.



11.3) Random Forest

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.The below diagram explains the working of the Random Forest algorithm.



12. PERFORMANCE EVALUATION

EVALUATION METRICS

The performance metrics used are as follows:-

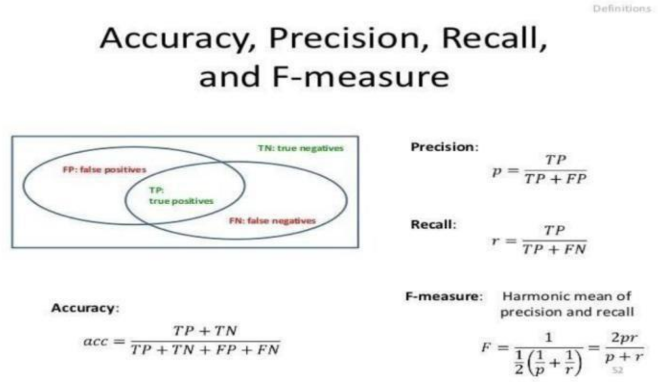
12.1) Precision is the ratio of true positive to the sum of false positive and true positive where true positive (tp) is the number of malicious URLs correctly classified and false positive (fp) is the number of URLs incorrectly classified.

12.2) Recall is the ratio of true positive to the sum of false negative and true positive where false negative (fn) is the number of incorrectly classified URLs.

12.3) Accuracy is the number of correct classifications of either malicious or benign URLs out of all URLs in the dataset where true negative (tn) is the number of correct classifications of benign as benign.

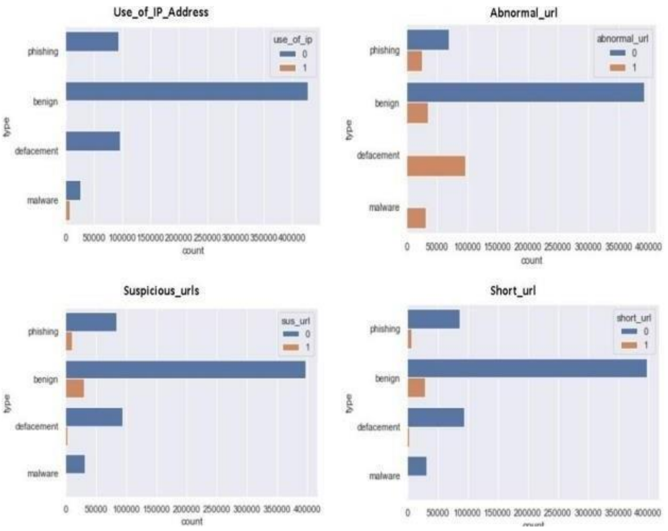
12.4) F-Measure is the harmonic mean of recall and precision

12.5) Execution Time is the time required to train and test the classification model.


Definitions
Accuracy, Precision, Recall, and F-measure
TN: true negatives
FP: false positives
TP: true positives
FN: false negatives
Precision:
$p = \frac{TP}{TP+FP}$
Recall:
$r = \frac{TP}{TP+FN}$
Accuracy:
$acc = \frac{TP+TN}{TP+TN+FP+FN}$
F-measure: Harmonic mean of precision and recall
$F = \frac{1}{\frac{1}{2}\left(\frac{1}{p}+\frac{1}{r}\right)} = \frac{2pr}{p+r}$
52

13. EXPLORATORY DATA ANALYSIS

In this step, we will check the distribution of different features for all four classes of URLs



As we can observe from the above distribution of use_ip_address feature, only malware URLs have IP addresses. In the case of abnormal_url, defacement URLs have higher distribution. From the distribution of suspicious_urls, it is clear that benign URLs have highest distribution while phishing URLs have a second highest distribution. As suspicious URLs consist of transaction and payment-related keywords and generally genuine banking or payment-related URLs consist of such keywords that's why benign URLs have the highest distribution.As per the short_url distribution, we can observe that benign URLs have the highest short URLs as we know that generally, we use URL shortening services for easily sharing long-length URLs.

14. FEATURE ENGINEERING

In this step, we will extract the following lexical features from raw URLs, as these features will be used as the input features for training the machine learning model. The following features are created as follows:

- having_ip_address: Generally cyber attackers use an IP address in place of the domain name to hide the identity of the website. This feature will check whether the URL has IP address or not.

- abnormal_url: This feature can be extracted from the WHOIS database. For a legitimate website, identity is typically part of its URL.

- google_index: In this feature, we check whether the URL is indexed in google search console or not.

- Count. : The phishing or malware websites generally use more than two sub-domains in the URL. Each domain is separated by dot (.). If any URL contains more than three dots(.), then it increases the probability of a malicious site.

- Count-www: Generally most of the safe websites have one www in its URL. This feature helps in detecting malicious websites if the URL has no or more than one www in its URL. cou□t@: The presence of the "@" symbol in the URL ignores everything previous to it.

- Count_dir: The presence of multiple directories in the URL generally indicates suspicious websites.

- Count_embed_domain: The number of the embedded domains can be helpful in detecting malicious URLs. It can be done by checking the occurrence of "//" in the URL.

- Suspicious words in URL: Malicious URLs generally contain suspicious words in the URL such as PayPal, login, sign in, bank, account, update, bonus, service, ebayisapi, token, etc. We have found the presence of such frequently occurring suspicious words in the URL as a binary variable i.e., whether such words present in the URL or not.

- Short_url: This feature is created to identify whether the URL uses URL shortening services like bit. \ly, goo.gl, go2l.ink, etc. □ Count_https: Generally malicious URLs do not use HTTPS protocols as it generally requires user credentials and ensures that the website is safe for transactions. So, the presence or absence of HTTPS protocol in the URL is an important feature. 14

- Count_http: Most of the time, phishing or malicious websites have more than one HTTP in their URL whereas safe sites have only one HTTP.

- Count%: As we know URLs cannot contain spaces. URL encoding normally replaces spaces with symbol (%). Safe sites generally contain less number of spaces whereas malicious websites generally contain more spaces in their URL hence more number of %.

- Count?: The presence of symbol (?) in URL denotes a query string that contains the data to be passed to the server. More number of ? in URL definitely indicates suspicious URL.

- Count-: Phishers or cybercriminals generally add dashes(-) in prefix or suffix of the brand name so that it looks genuine URL. For example. www.flipkart-india.com.

- Count=: Presence of equal to (=) in URL indicates passing of variable values from one form page t another. It is considered as riskier in URL as anyone can change the values to modify the page.

- url_length: Attackers generally use long URLs to hide the domain name. We found the average length of a safe URL is 74.

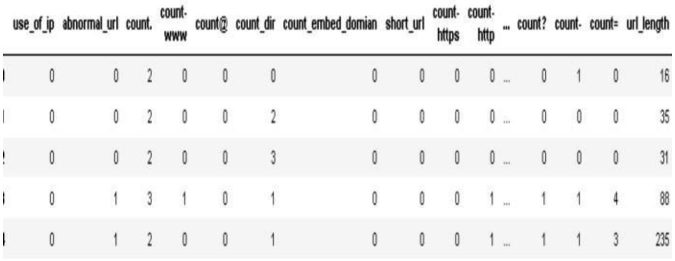
- hostname_length: The length of the hostname is also an important feature for detecting malicious URLs.

- First directory length: This feature helps in determining the length of the first directory in the URL. So looking for the first '/' and counting the length of the URL up to this point helps in finding the first directory length of the URL. For accessing directory level information we need to install python library TLD. You can check this link for installing TLD.

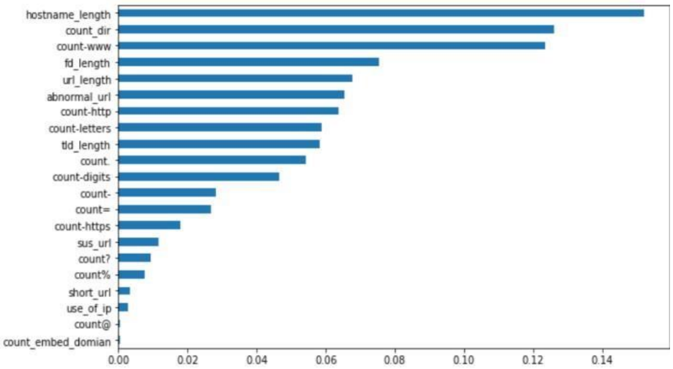
- Length of top-level domains: A top-level domain (TLD) is one of the domains at the highest level in the hierarchical Domain Name System of the Internet. For example, in the domain name www.example.com, the top-level domain is com. So, the length of TLD is also important in identifying malicious URLs. As most of the URLs have .com extension. TLDs in the range from 2 to 3 generally indicate safe URLs.

- Count_digits: The presence of digits in URL generally indicate suspicious URLs. Safe URLs generally do not have digits so counting the number of digits in URL is an important feature for detecting malicious URLs.

- Count_letters: The number of letters in the URL also plays a significant role in identifying malicious URLs. As attackers try to increase the length of the URL to hide the domain name and this is generally done by increasing the number of letters and digits in the URL.



use_of_ip	abnormal_url	count.	count-www	count@	count_dir	count_embed_domian	short_url	count-https	count-http	...	count?	count-	count=	url_length
0	0	2	0	0	0	0	0	0	0	...	0	1	0	16
0	0	2	0	0	2	0	0	0	0	...	0	0	0	35
0	0	2	0	0	3	0	0	0	0	...	0	0	0	31
0	1	3	1	0	1	0	0	0	1	...	1	1	4	88
0	1	2	0	0	1	0	0	0	1	...	1	1	3	235



15. TRAINING AND TEST SPLIT

The next step is to split the dataset into train and test sets. We have split the dataset into 80:20 ratio i.e., 80% of the data was used to train the machine learning models, and the rest 20% was used to test the model.As we know we have an imbalanced dataset. The reason for this is around 66% of the data has benign URLs, 5% malware, 14% phishing, and 15% defacement URLs. So after randomly splitting the dataset into train and test, it may happen that the distribution of different categories got disturbed which will highly affect the performance of the machine learning model. So to maintain the same proportion of the target variable stratification is needed.This stratify parameter makes a split so that the proportion of values in the sample produced will be the same as the proportion of values provided to the parameter stratify.

16. DEVOPS ARCHITECTURE:

16.1) GITHUB

It All Starts with Writing of Code, Developer will Write a Code & Store it in A Public open Source Registry known as GitHub (Version Control System). Why Version Control System ?, Because The Code keeps on getting Updated every time as we Get Updates on the App Store/ play Store.

16.2) JENKINS

Jenkins is a popular open source tool for CI/CD that is free to use. While you may need some server administration skills to configure and monitor Jenkins, there are many advantages to consider. The Jenkins project includes a large plugin ecosystem, the community around it is thriving and it is actively developed.

In Other Words, Jenkins is an open source continuous integration/continuous delivery and deployment (CI/CD) automation software DevOps tool written in the Java programming language. It is used to implement CI/CD workflows, called pipelines.

Jenkins will be integrated with GitHub through a Concept called Poll SCM which we will see wherever needed. What this will do is it will trigger the Jenkins Code which is to be executed whenever there is Some change in the Code Committed at GitHub. In Other words whenever the Code is updated, the Jenkins Code have to Run.

16.3) ANSIBLE

Ansible is the simplest way to automate apps and IT infrastructure. Application Deployment + Configuration Management + Continuous Delivery.

We Will Have a Separate EC2 Instance for the Ansible Because it will also Contain Docker Platform which will Build the Docker Image using Docker Build Command, Then Log in to Docker Hub Id using docker login -u xyzz -p xyzzz and Push it to the Docker Hub.

16.4) KUBERNETES

Kubernetes, often abbreviated as “K8s”, orchestrates containerized applications to run on a cluster of hosts. The K8s system automates the deployment and management of cloud native applications using on-premises infrastructure or public cloud platforms.

We Will Have two Separate EC2 Instances for the k8s Because Kubernetes Requires More Computer Resources Compared to Docker & Ansible, etc. So One Instance will act as a Master Node & One as a Slave Node, The Master Node Keeps a track of Slave Node & Note : having at least one Slave Node is necessary to make Kubernetes Work the Way we Want.

The Kubernetes Resources i.e (Deployment & Service) will be automated using an Ansible Playbook. The Ansible hosts Will contain the IP address of the Kubernetes Master Node which will be used for Configuration through SSH protocol.

17. SOFTWARE AND HARDWARE REQUIRMENTS

17.1) FOR CODING

i. Language:-Python-3 ii.
IDLE:-Google Colab

17.2) FOR DEPLOYMENT

i. AWS (Amazon Web Services).

ii. EC2 Instances.

iii. 1 Kubernetes Cluster, 2 Worker Nodes, 1 Instance to Setup CI/CD, 1 Instance for Ansible Automation and Containerisation of Web Application.

v. Load Balancer to Split traffic Equally among Instances.

vi. Route53 Service to associate a Domain name with Load Balancer.

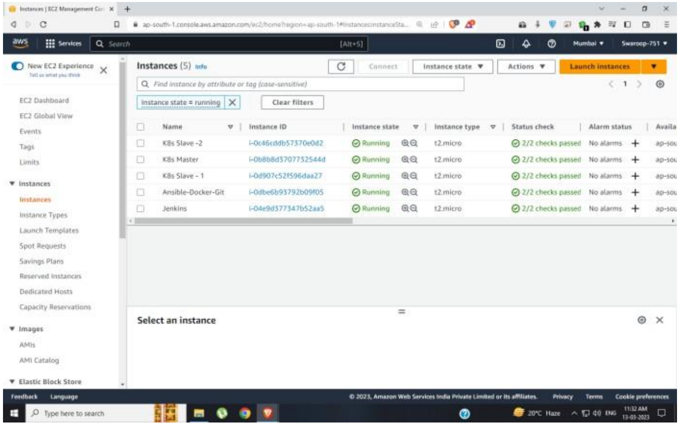
17.3) FOR AUTOMATION

i. Git-GitHub Version Control and Source Code Management.

ii. Jenkins CI/CD to set up one click Automation through end to end using pipeline. iii. Docker to Containerize the Application. iv. Kubernetes for managing the Deployments.

v. Ansible to automate Kubernetes Deployments and Services.

18. PROJECT EXECUTION

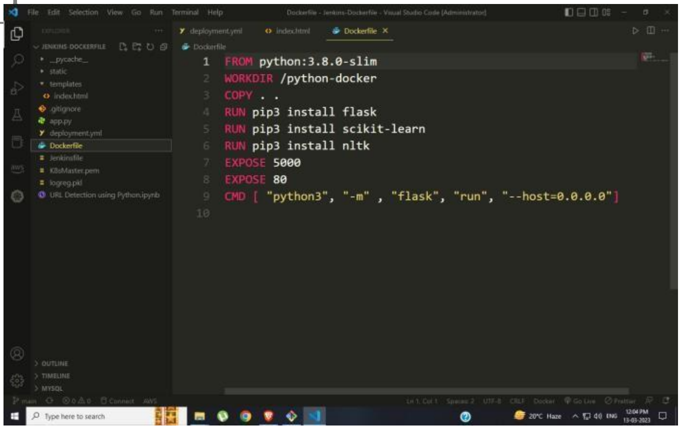


Instances (5)

Instance state = running

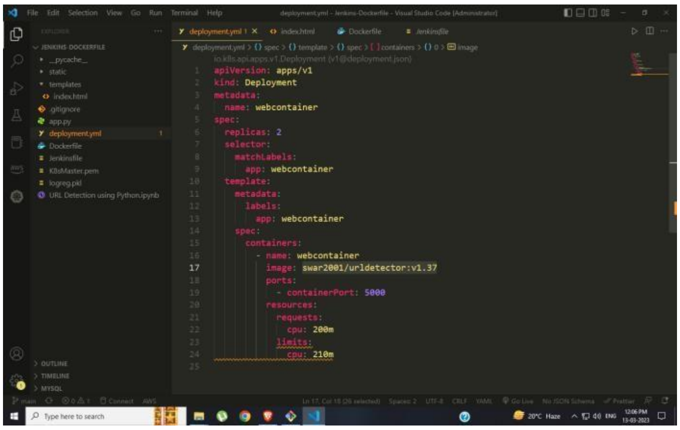
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availa
K8s Slave -2	i-0c46cddb57370e0d2	Running	t2.micro	2/2 checks passed	No alarms	ap-sou
K8s Master	i-0b8b8d37077325444d	Running	t2.micro	2/2 checks passed	No alarms	ap-sou
K8s Slave - 1	i-0d907c52f596daa27	Running	t2.micro	2/2 checks passed	No alarms	ap-sou
Ansible-Docker-Git	i-0dbe6b93792b09f05	Running	t2.micro	2/2 checks passed	No alarms	ap-sou
Jenkins	i-04e9d377347b52aa5	Running	t2.micro	2/2 checks passed	No alarms	ap-sou

5 EC2-Instances (1 - Jenkins Server, 1- Kubernetes Master node, 2 - Kubernetes Slave Node, 1- Ansible & Docker Node)



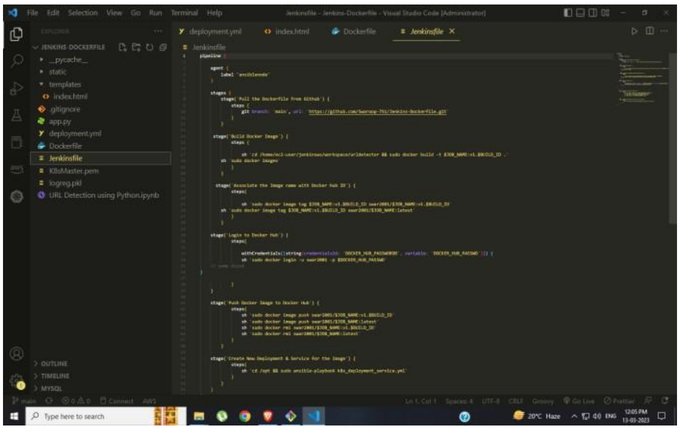
```
FROM python:3.8.0-slim
WORKDIR /python-docker
COPY . .
RUN pip3 install flask
RUN pip3 install scikit-learn
RUN pip3 install nltk
EXPOSE 5000
EXPOSE 80
CMD [ "python3", "-m" , "flask", "run", "--host=0.0.0.0"]
```

Dockerfile for Containerizing Flask app



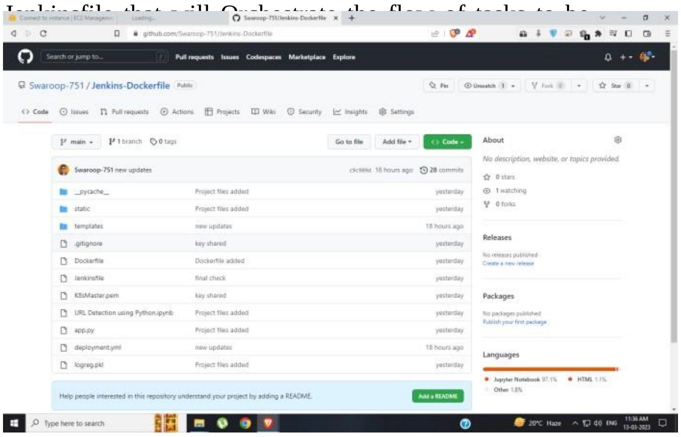
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webcontainer
spec:
  replicas: 2
  selector:
    matchLabels:
      app: webcontainer
  template:
    metadata:
      labels:
        app: webcontainer
    spec:
      containers:
        - name: webcontainer
          image: swar2001/urldetector:v1.37
          ports:
            - containerPort: 5000
          resources:
            requests:
              cpu: 200m
            limits:
              cpu: 210m
```

K8s Deployment file to create flask app Deployment



```
pipeline {

    agent {
        label 'ansiblenode'
    }

    stages {
        stage('Pull the Dockerfile from Github') {
            steps {
                git branch: 'main', url: 'https://github.com/Swaroop-751/Jenkins-Dockerfile.git'
            }
        }

     stage('Build Docker Image') {
            steps {

                sh 'cd /home/ec2-user/jenkinsws/workspace/urldetector && sudo docker build -t $JOB_NAME:v1.$BUILD_ID .'
        sh 'sudo docker images'
            }
        }

      stage('Associate the Image name with Docker Hub ID') {
            steps{

                sh 'sudo docker image tag $JOB_NAME:v1.$BUILD_ID swar2001/$JOB_NAME:v1.$BUILD_ID'
        sh 'sudo docker image tag $JOB_NAME:v1.$BUILD_ID swar2001/$JOB_NAME:latest'
            }
        }

    stage('Login to Docker Hub') {
            steps{

                withCredentials([string(credentialsId: 'DOCKER_HUB_PASSWORD', variable: 'DOCKER_HUB_PASSWD')]) {
                sh 'sudo docker login -u swar2001 -p $DOCKER_HUB_PASSWD'
    // some block
}

            }
    }

    stage('Push Docker Image to Docker Hub') {
            steps{
                sh 'sudo docker image push swar2001/$JOB_NAME:v1.$BUILD_ID'
                sh 'sudo docker image push swar2001/$JOB_NAME:latest'
                sh 'sudo docker rmi swar2001/$JOB_NAME:v1.$BUILD_ID'
                sh 'sudo docker rmi swar2001/$JOB_NAME:latest'
            }
        }

    stage('Create New Deployment & Service for the Image') {
            steps{
                sh 'cd /opt && sudo ansible-playbook k8s_deployment_service.yml'
            }
        }
    }
```

Jenkinsfile that will Orchestrate the flow of tasks to be



Swaroop-751 / Jenkins-Dockerfile Public

main 1 branch 0 tags

Swaroop-751 new updates c9c989d 18 hours ago 28 commits

pycache	Project files added	yesterday
static	Project files added	yesterday
templates	new updates	18 hours ago
.gitignore	key shared	yesterday
Dockerfile	Dockerfile added	yesterday
Jenkinsfile	final check	yesterday
K8sMaster.pem	key shared	yesterday
URL Detection using Python.ipynb	Project files added	yesterday
app.py	Project files added	yesterday
deployment.yml	new updates	18 hours ago
logreg.pkl	Project files added	yesterday

Help people interested in this repository understand your project by adding a README.

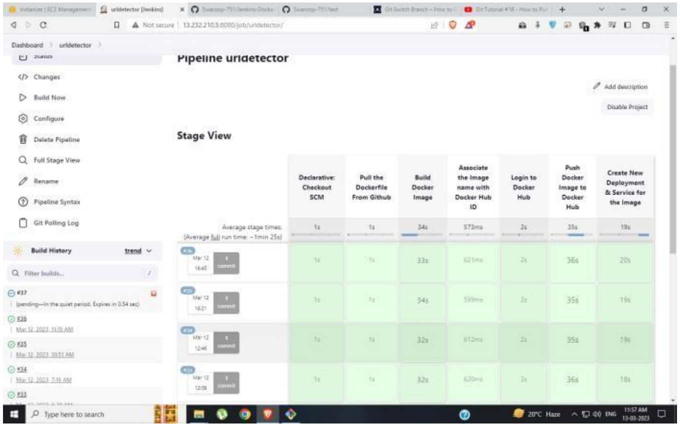
About
No description, website, or topics provided.
0 stars
1 watching
0 forks

Releases
No releases published

Packages
No packages published

Languages
Jupyter Notebook 97.1% HTML 1.1% Other 1.8%

Gitub Repo Integrated with Jenkins Pol SCM

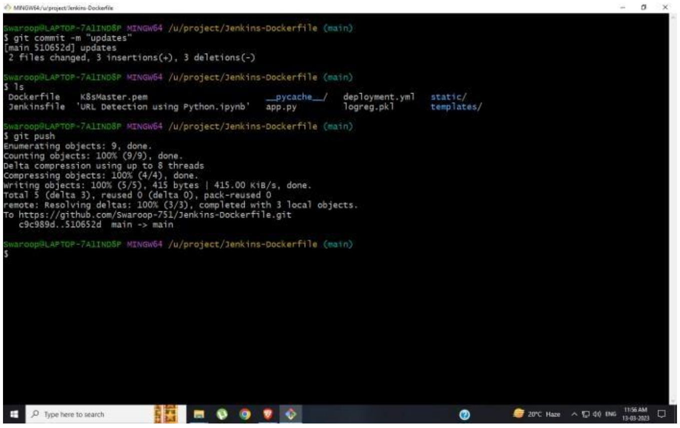


Pipeline urldetector

Stage View

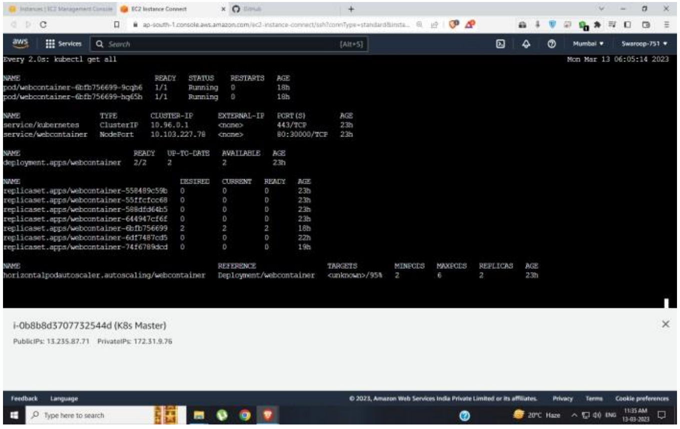
	Declarative: Checkout SCM	Pull the Dockerfile From Github	Build Docker Image	Associate the Image name with Docker Hub ID	Login to Docker Hub	Push Docker Image to Docker Hub	Create New Deployment & Service for the Image
Average stage times: (Average full run time: ~1min 25s)	1s	1s	34s	573ms	2s	35s	19s
#36 Mar 12 16:45 1 commit	1s	1s	33s	621ms	2s	36s	20s
#35 Mar 12 16:21 1 commit	1s	1s	34s	599ms	2s	35s	19s
#34 Mar 12 12:46 1 commit	1s	1s	32s	612ms	2s	35s	19s
#33 Mar 12 12:09 1 commit	1s	1s	32s	620ms	2s	36s	18s

A Pipeline Gets Triggered



```
Swaroop@LAPTOP-7A1IND8P MINGW64 /u/project/Jenkins-Dockerfile (main)
$ git commit -m "updates"
[main 510652d] updates
 2 files changed, 3 insertions(+), 3 deletions(-)

Swaroop@LAPTOP-7A1IND8P MINGW64 /u/project/Jenkins-Dockerfile (main)
$ ls
 Dockerfile    K8sMaster.pem                          __pycache__/   deployment.yml   static/
 Jenkinsfile  'URL Detection using Python.ipynb'      app.py         logreg.pkl       templates/

Swaroop@LAPTOP-7A1IND8P MINGW64 /u/project/Jenkins-Dockerfile (main)
$ git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 415 bytes | 415.00 KiB/s, done.
Total 5 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/Swaroop-751/Jenkins-Dockerfile.git
   c9c989d..510652d  main -> main

Swaroop@LAPTOP-7A1IND8P MINGW64 /u/project/Jenkins-Dockerfile (main)
$
```

On Git Push

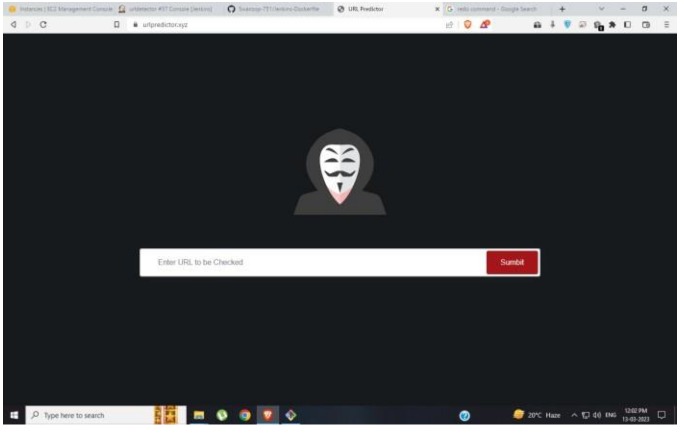


```
Every 2.0s: kubectl get all                                                    Mon Mar 13 06:05:14 2023

NAME                                READY   STATUS    RESTARTS   AGE
pod/webcontainer-6bfb756699-9cqh6   1/1     Running   0          18h
pod/webcontainer-6bfb756699-hq65h   1/1     Running   0          18h

NAME                     TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
service/kubernetes       ClusterIP   10.96.0.1       <none>        443/TCP        23h
service/webcontainer     NodePort    10.103.227.78   <none>        80:30000/TCP   23h

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/webcontainer   2/2     2            2           23h

NAME                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/webcontainer-558489c59b   0         0         0       23h
replicaset.apps/webcontainer-55ffcfcc68   0         0         0       23h
replicaset.apps/webcontainer-588dfd64b5   0         0         0       23h
replicaset.apps/webcontainer-644947cf6f   0         0         0       23h
replicaset.apps/webcontainer-6bfb756699   2         2         2       18h
replicaset.apps/webcontainer-6df7487cd5   0         0         0       22h
replicaset.apps/webcontainer-74f6789dcd   0         0         0       19h

NAME                                               REFERENCE                 TARGETS          MINPODS   MAXPODS   REPLICAS   AGE
horizontalpodautoscaler.autoscaling/webcontainer   Deployment/webcontainer   <unknown>/95%    2         6         2          23h
```

i-0b8b8d3707732544d (K8s Master)

PublicIPs: 13.235.87.71 PrivateIPs: 172.31.9.76

Flask App Deployed inside K8s Pods



GUI with Domain name and SSL Certificate

19. CONCLUSION AND FUTURE WORK

In this, we have demonstrated a machine learning approach to detect Malicious URLs. We have created 22 lexical features from raw URLs and trained three machine learning models XG Boost, Light GBM, and Random forest. Further, we have compared the performance of the 3 machine learning models and found that Random forest outperformed others by attaining the highest accuracy of 96.6%. By plotting the feature importance of Random forest we found that hostname_length, count_dir, count-www, fd_length, and url_length are the top 5 features for detecting the malicious URLs. At last, we have coded the prediction function for classifying any raw URL using our saved model i.e., RandomForest.

For future work, other datasets can be used to evaluate model performance. Deep learning techniques can also be utilized for analysis. Instead of using k-fold cross-validation, splitting methods can be employed to obtain training and testing sets. Unsupervised ML can also be considered to find patterns and similarities between different URL types.

20. REFERENCES

[1] D. Sahoo, C. Liu, S.C.H. Hoi, “Malicious URL Detection using Machine Learning: A Survey”. CoRR, abs/1701.07179, 2017.

[2] M. Khonji, Y. Iraqi, and A. Jones, “Phishing detection: a literature survey,” IEEE Communications Surveys & Tutorials, vol. 15, no. 4, pp. 2091–2121, 2013.

[3] M. Cova, C. Kruegel, and G. Vigna, “Detection and analysis of drivebydownload attacks and malicious JavaScript code,” in Proceedings of the 19th international conference on World wide web. ACM, 2010, pp. 281– 290.

[4] R. Heartfield and G. Loukas, “A taxonomy of attacks and a survey of defence mechanisms for semantic social engineering attacks,” ACM Computing Surveys (CSUR), vol. 48, no. 3, p. 37, 2015.

[5] Internet Security Threat Report (ISTR) 2019– Symantec.

https://www.symantec.com/content/dam/symantec/docs/report s/istr-24- 2019-en.pdf [Last accessed 10/2019].

[6] S. Sheng, B. Wardman, G. Warner, L. F. Cranor, J. Hong, and C. Zhang, "An empirical analysis of phishing blacklists," in Proceedings of Sixth Conference on Email and Anti-Spam (CEAS), 2009.

[7] C. Seifert, I. Welch, and P. Komisarczuk, "Identification of malicious web pages with static heuristics," in Telecommunication Networks and Applications Conference, 2008. ATNAC 2008. Australasian. IEEE, 2008, pp. 91–96. [8] S. Sinha, M. Bailey, and F. Jahanian, "Shades of grey: On the effectiveness of reputation-based "blacklists"," in Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on. IEEE, 2008, pp. 57–64.

[9] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Identifying suspicious urls: an application of large-scale online learning," in Proceedings of the 26th Annual International Conference on Machine Learning. ACM, 2009, pp. 681–688.

[10] B. Eshete, A. Villafiorita, and K. Weldemariam, "Binspect: Holistic analysis and detection of malicious web pages," in Security and Privacy in Communication Networks. Springer, 2013, pp. 149–166.

[11] S. Purkait, "Phishing counter measures and their effectiveness– literature review," Information Management & Computer Security, vol. 20, no. 5, pp. 382–420, 2012. [12] Y. Tao, "Suspicious url and device detection by log mining," Ph.D. dissertation, Applied Sciences: School of Computing Science, 2014. 21

[13] G. Canfora, E. Medvet, F. Mercaldo, and C. A. Visaggio, "Detection of malicious web pages using system calls sequences," in Availability, Reliability, and Security in Information Systems. Springer, 2014, pp. 226–238.

[14] Leo Breiman.: Random Forests. Machine Learning 45 (1), pp. 5- 32, (2001).

[15] Thomas G. Dietterich. Ensemble Methods in Machine Learning. International Workshop on Multiple Classifier Systems, 1-15, Cagliari, Italy, 2000.

[16] Developer Information. https://www.phishtank.com/developer_info.php. [Last accessed 11/2019].

[17] URLhaus Database Dump. https://urlhaus.abuse.ch/downloads/csv/.

[18] Dataset URL. http://downloads.majestic.com/majestic_million.csv

[19] Malicious_n_NonMaliciousURL. https://www.kaggle.com/antonyj453/ urldataset#data.csv.

[20] chrome.zip. https://drive.google.com/file/d/13G_Ndr4hMFx_qWyTEjHuO yJmHFW D0Gud/view?fbclid=IwAR0SLVCrvjHHGmoHZH97nXN3B mDMY7jG4SOsKZYLAZjTFgeoJA Dfli64-g.