RakshaNet: URL - Aware Malicious Website Classifier

1st Shreyas Rajesh Labhsetwar
Department of Computer Engineering
Fr. C. Rodrigues Institute of Technology
Navi Mumbai, India
shreyas.labhsetwar@fcrit.onmicrosoft.com

2nd Piyush Arvind Kolte
Department of Computer Engineering
Fr. C. Rodrigues Institute of Technology
Navi Mumbai, India
piyush.kolte@fcrit.onmicrosoft.com

3rd Atharva Santosh Sawant
Department of Computer Engineering
Fr. C. Rodrigues Institute of Technology
Navi Mumbai, India
atharva.sawant@fcrit.onmicrosoft.com

***Abstract*—Software revolution has resulted in a hyperbolic increase in internet browsing, making internet security a vital issue. Most naive users typically visit unknown websites, and due to their lack of awareness and software proficiency, malicious websites pose a significant threat to their data and security. This paper proposes a classification algorithm that determines whether a website is malicious or benign based on its application layer and network layer features. These features are extracted from the header and body of the HTTP/HTTPS request/response of a website, upon which the ML algorithm acts to determine whether the website is malicious. The client-server data can be intercepted using a proxy service, such as the Squid proxy, and the ML classifier runs as part of the Internet Content Adaptation Protocol (ICAP). If a website is determined as potentially malicious, the user shall be notified immediately and redirected back to the previous benign webpage. The URL parameters extracted include the Server name, DNS query time, TCP details, etc., which are chosen after extensive study of the contribution of these features (importance) to the classification. The study is performed on the decision tree and random forest supervised machine learning algorithms, and it is observed that the random forest algorithm is the most suitable ML classification methodology, achieving a test accuracy of 92%. The classification performance is visualized with the help of the confusion matrix and receiver operating characteristics curve (ROC), with an area under the curve (AUC) of 84%. Thus, this paper proposes an end-to-end software that utilises the random forest algorithm for the classification of websites as malicious or benign, and preempt users from accessing harmful websites.**

***Index Terms*—Application Layer, Network Layer, Proxy Service, Feature Selection, Decision Tree, Random Forest Algorithm**

I. Introduction

The internet has become essential in our daily life; it's the base of banking transactions, shopping, entertainment, resource sharing, news, and social networking. The World Wide Web has become the global platform for millions of users all over the world, and its popularity has also attracted hackers, intruders, attackers, etc., to abuse the Internet and its users to perform illegitimate activities for financial profit. Common attacks using malicious URLs include Drive-by download, phishing, and social engineering & spamming [1], [2]. If a user accesses the malicious website through a search engine with no perception, malicious scripts usually launch attacks to install rogue programs, steal personal identities and credentials, or even control the victim's machine [3]. Because of this menace, prevention is more valuable and preferable than handling the situation after infected. It is very important to detect malicious search result pages before visiting them. Each time when the users decide to access unknown websites or click on an unfamiliar URL, a sanity check must be performed to evaluate the associated risk of visiting that website and the challenges that might be encountered.

In this paper, we explore a machine learning-based classification algorithm, capable of predicting whether a website is malicious or benign by analyzing its application layer and network layer features. Our solution will work by retrieving the HTML code representing a Web page and extracting the important parameters from the header and body of the request/response. These features are preprocessed and served to the Random Forest Machine Learning Classifier, which computes a score indicating the likelihood of the said URL being malicious.

The paper is organized into multiple sections, the first section comprising the introduction to the topic. The second section highlights the related works, the third section delves into data collection, and feature importance. The fourth section discusses the proposed system & classification algorithms, and the experimental results are presented in the fifth section. The conclusion is presented in the sixth section.

II. Related Works

Manling Wang et al. [4] explore the various attacks performed by malicious websites and associate a cost with the detection. These websites make use of various models to attack the detection system. They compare the SVM(Support Vector Machine) and Linear Discriminant Classifier in terms of strength, robustness, and other factors, whilst also exploring their flaws. As per the comparisons, SVM is found to be more robust than Linear Discriminant Classifier. Yet, the study does not find the best algorithm for malicious website detection.

Sunil B. Rathod et al. [5] study the various types of malicious and harmful content found in Emails. They utilize data mining methods such as Bayesian Classifier and Decision Tree to create a detection system for spam and harmful emails. These methods take into consideration various parameters related to emails, yet do not demonstrate a particularly high accuracy rate.

Toshiki Shibahara et al. [6] propose an approach for identifying harmful and suspicious websites using subgraphs. They implement a classifier for the classification of websites into harmful and benign categories and construct a redirection graph out of the websites under consideration. This methodology works well for a small database of websites but suffers from an exponential decrease in speed upon scaling the database for real-time usage.

Siddharth Singhal et al. [7] explore the various machine learning-based methodologies to detect malicious and harmful websites. They implement software that accepts a URL as an input and classifies the corresponding website as benign or malicious, based on the features of the website. However, they identify a flaw with their methodology; the training data that they used could be exploited by the attackers to modify the aspects of the harmful URLs and trick the classifier to believe the URLs as benign websites.

Yu-Chen Chen et al. [8] study the use of feature analysis to detect and identify malicious URLs. They implement a machine learning-based software to extract and analyze 41 attributes of any URL, based upon which a score is assigned to the URL indicating its probability of being malicious/benign. They use several algorithms including Analysis of Variance and eXtreme Gradient Boosting for the identification of features and classification. However, the results of the research aren't robust, as their software achieves an accuracy of 99.99%, indicating extreme overfitting, as also indicated by their performance graphs.

III. Data Collection & Preprocessing

A. Dataset

The dataset used for this research is an open-source dataset of Malicious and Benign Websites obtained on Kaggle [9]. The data was collated from a range of verified sources of websites, in a low and interactive client honeypot to isolate the network traffic. Software tools were utilized to gather additional information about the websites, including the server and country. A total of 1781 URLs make up the dataset, with each possessing 21 attributes, as described in Table II.

B. Data Interpolation

The missing values in the dataset are estimated using Python's data interpolation techniques. The various techniques experimented with were linear, time, index, values, nearest, zero, slinear, quadratic, cubic, barycentric, krogh, and polynomial [10].

C. Feature Importance

Given a database D, with D = $\{t_1$,t_2,t_3,.....,$t_n\}$, which consists of n attributes A = $\{A_1$, A_2, A_3, ..., $A_n\}$, then the importance of each attribute/feature refers to the relevance of each feature to the ML classifier when making a prediction. Feature importance is extremely important for:

1) Understanding the Data & inter-feature dependencies
2) Feature Selection (Reduction)
3) Deciding the ML model to be used

In this paper, we have used the Python's Scikit-learn library, which computes the Feature Importance in four steps [11]:

- Step 1: Calculation of Node Importance base on the Gini Importance

$$node_j^i = wt_j Imp_j - wt_{left(j)} Imp_{left(j)} - wt_{right(j)} Imp_{right(j)}$$

  Where,
  $node_j^i$ is the importance of node j, wt_j is the weighted number of samples reaching node j, Imp_j is the impurity value of node j, $left(j)$ is the child node from left split on node j, $right(j)$ is the child node from right split on node j
- Step 2: Computing the Importance of each Feature

$$f_i^i = \frac{\sum j : NodejSplitsOnFeaturei^{nodei_j}}{\sum k \in AllNodes^{nodei_k}}$$

  Where,
  f_i^i is the importance of feature i, $nodei_j$ is the importance of node j
- Step 3: Normalized Feature Importance

$$NORMf_i^i = \frac{f_i^i}{\sum j \in AllFeatures^{f_i^i}}$$

- Step 4: Random Forest Feature Importance

$$RFf_i^i = \frac{\sum j \in AllTrees^{NORMf_i^{ij}}}{T}$$

  Where,
  RFf_i^i is the importance of feature i calculated from all trees in the Random Forest model, $NORMf_i^{ij}$ is the normalized feature importance for i in tree j, T is the total number of trees

Thus, the feature importance values for all the dataset attributes are summarised in Table I.

TABLE I
Feature Importance

Feature	Importance
WHOIS_STATEPRO	0.125
SERVER	0.090
WHOIS_COUNTRY	0.076
WHOIS_REGDATE	0.075
REMOTE_APP_PACKETS	0.066
SOURCE_APP_BYTES	0.064
DIST_REMOTE_TCP_PORT	0.061
WHOIS_UPDATED_DATE	0.050
URL_LENGTH	0.045
NUMBER_SPECIAL_CHARACTERS	0.042
CONTENT_LENGTH	0.041
REMOTE_APP_BYTES	0.039
CHARSET	0.038
TCP_CONVERSATION_EXCHANGE	0.036
APP_PACKETS	0.034
APP_BYTES	0.032
SOURCE_APP_PACKETS	0.030
REMOTE_IPS	0.025
DNS_QUERY_TIMES	0.021

TABLE II
DATASET DESCRIPTION

Dataset Column	Parameter						
	Count	**Unique**	**Freq**	**Mean**	**Std**	**25%**	**75%**
URL	1781	1781	1	NaN	NaN	NaN	NaN
URL_LENGTH	1781.000000	NaN	NaN	56.961258	27.555586	39.000000	68.000000
NUMBER_SPECIAL_CHARACTERS	1781.000000	NaN	NaN	11.111735	4.549896	8.000000	13.000000
CHARSET	1781	9	676	NaN	NaN	NaN	NaN
SERVER	1780	239	386	NaN	NaN	NaN	NaN
CONTENT_LENGTH	969.000000	NaN	NaN	11726.927761	36391.809051	324.000000	11323.000000
WHOIS_COUNTRY	1781	49	1103	NaN	NaN	NaN	NaN
WHOIS_STATEPRO	1781	182	372	NaN	NaN	NaN	NaN
WHOIS_REGDATE	1781	891	127	NaN	NaN	NaN	NaN
WHOIS_UPDATED_DATE	1781	594	139	NaN	NaN	NaN	NaN
TCP_CONVERSATION_EXCHANGE	1781.000000	NaN	NaN	16.261089	40.500975	0.000000	22.000000
DIST_REMOTE_TCP_PORT	1781.000000	NaN	NaN	5.472768	21.807327	0.000000	5.000000
REMOTE_IPS	1781.000000	NaN	NaN	3.060640	3.386975	0.000000	5.000000
APP_BYTES	1.781000e+03	NaN	NaN	2.982339e+03	5.605057e+04	0.000000e+00	2.328000e+03
SOURCE_APP_PACKETS	1781.000000	NaN	NaN	18.540146	41.627173	0.000000	26.000000
REMOTE_APP_PACKETS	1781.000000	NaN	NaN	18.746210	46.397969	0.000000	25.000000
SOURCE_APP_BYTES	1.781000e+03	NaN	NaN	1.589255e+04	6.986193e+04	0.000000e+00	9.806000e+03
REMOTE_APP_BYTES	1.781000e+03	NaN	NaN	3.155599e+03	5.605378e+04	0.000000e+00	2.701000e+03
APP_PACKETS	1781.000000	NaN	NaN	18.540146	41.627173	0.000000	26.000000
DNS_QUERY_TIMES	1780.000000	NaN	NaN	2.263483	2.930853	0.000000	4.000000
Type	1781.000000	NaN	NaN	0.121280	0.326544	0.000000	0.000000

D. Train & Test Split

The dataset is split into two subgroups for the purpose of training the ML model, and then testing/evaluating its performance. A split ratio of 80:20 is chosen on an experimental basis. Thus, a total of 1424 rows make up the training data, and 357 rows are used for model evaluation.

IV. PROPOSED SYSTEM & CLASSIFICATION ALGORITHMS

A. Proposed System

The end-to-end RakshaNet software's pictorial representation is given in Fig. 1.

- A proxy service such as the Squid proxy will be used to intercept the requests and responses being exchanged between clients and servers.
- The intercepted messages will be forwarded by the proxy's Internet Content Adaptation Protocol (ICAP) client to the ICAP server.
- The ICAP server will extract the useful features of the message header and body, preprocess them, and serve to the ML classifier.
- The classifier predicts whether a particular website is malicious or benign, and based upon its prediction, the user can be allowed to visit the website (benign) or redirected back to the previous webpage (in case the next webpage is detected malicious)

B. Decision Tree Algorithm

1) Decision Trees: It is a rule-based classifier [12]. Given a database D, with D = $\{t_1,t_2,t_3,.....,t_n\}$, where t_i = $\{t_{i1}$, t_{i2}, t_{i3}, ..., $t_{in}\}$, and the database schema consists of n attributes A = $\{A_1$, A_2, A_3, ..., $A_n\}$ belonging to m classes C = $\{C_1$, C_2, C_3, ..., $C_m\}$, then decision tree is a classification tree associated with the dataset D such that:

- Each internal node is labelled with attribute A_i and each arc (edge) is labelled with a predicate.
- Each leaf node is labelled with class C_j.

2) Splitting Attributes: Attributes in the database schema used to label the nodes in the tree, and around which divisions can take place [13].

3) Pruning: Tree pruning can be performed using two methods:

- Pre-pruning: It is performed while the tree is being created. This prevents the tree from becoming too large.
- Post-pruning: It is performed after the construction of the tree. In this, portions of the tree can be either removed or combined to reduce the overall size of the tree.

4) Stopping Criteria: The creation of the tree stops when the training data gets properly classified. To prevent the creation of large trees, the algorithm must be stopped in between, which also reduces overfitting. In such cases, a trade-off occurs between the accuracy of classification and the performance.

5) Entropy: Entropy is a measure of the disorderliness in the dataset. Given the probabilities of classes as P = P_1, P_2, P_3, ..., P_c, entropy is defined as:

$$H(P_1, P_2, P_3, ..., P_c) = \sum_{i=1}^{c} -P_i * \log_2 P_i$$

6) Information Gain: Information gain is a measure of the amount of information a feature gives about a class. Given dataset D and split S, information gain is defined as:

$$Gain(D, S) = H(D) - \sum_{i=1}^{S} P(D_i) * H(D_i)$$

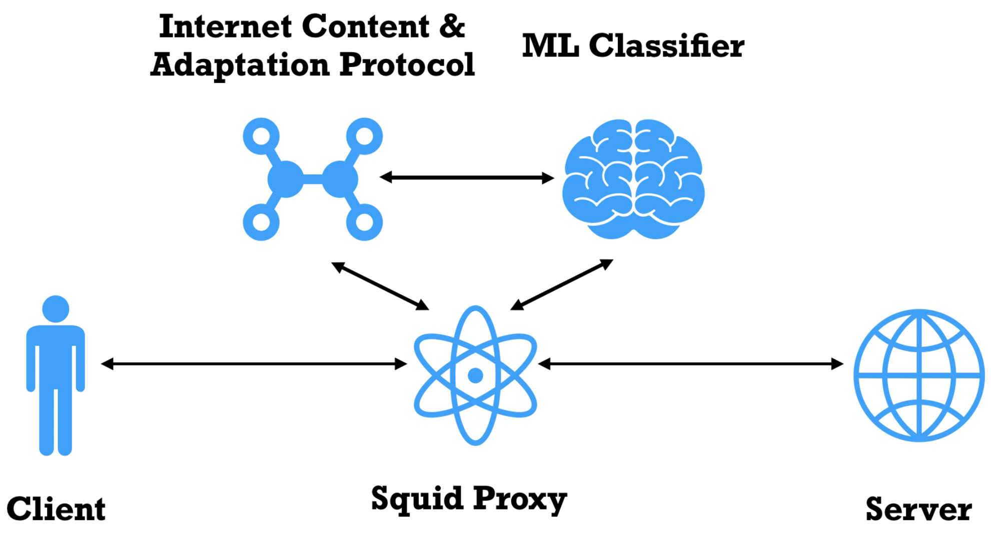

Fig. 1. Proposed System

Algorithm 1 Random Forest Algorithm

Precondition: A training set S := (x1,y1),...,(xn,yn), features F, and number of trees in forest **B**.

```
Function RandomForest($S$,$F$):
    $H = \phi$
    for $i \in 1,..,B$ do
        $S^{(i)} \leftarrow$ A bootstrap sample from $S$
        $h_i \leftarrow RANDOMIZED\_TREE\_LEARN(S^{(i)},F)$
        $H \leftarrow H \cup \{h_i\}$
    end
      return $H$
end function
Function RANDOMIZED_TREE_LEARN($S$,$F$):
    for each node do
        $f \leftarrow$ very small subset of F
        Split on best feature in $f$
    end
      return The learned tree
end function
```

C. Random Forest Algorithm

1) Working: Random forest is a supervised learning algorithm that operates by constructing multiple decision trees. The final decision is made by the forest-based on the majority of decisions pooled by the constituent trees [14], [15]. Random Forest works in two-phases:

- Creation of the random forest by combining N-Decision Trees
- Pool predictions for each tree & compute the majority decision

2) Important Hyperparameters: The hyperparameters in random forest algorithm are used to increase the predictive power of the algorithm and to also make the model faster.

- Predictive Power
    1) Number of Estimators: The number of Decision Trees in a Random Forest. For this research, n_estimators = 100.
    2) Max Number of Features: Random forest models randomly resample features prior to determining the best split. Larger max_feature values can result in improved model performance (trees have a larger selection of features). The value is determined automatically and dynamically.
    3) Min sample leaf: Minimum number of leafs required to split an internal node. For this research, min_sample_leaf = 1.
- Increase Model speed
    1) Number of jobs: The number of jobs to run in parallel. For this research, n= -1, which instructs to use all the available processors.
    2) Maximum depth: It represents the maximum depth of each tree. For this research, max_depth = 30.

3) Advantages over Decision Tree:

- Random Forest performs both Classification and Regression tasks
- It can handle large datasets with high dimensionality
- It enhances the accuracy of the model and prevents overfitting
- Pooling the majority decision ensures superior classification performance in case of complex prediction tasks

Thus, due to these advantages, this paper utilises the Random Forest Algorithm for the classification task of Malicious and Benign websites.

V. Experimental Results

The performance of the Random Forest Classifier is evaluated using various performance graphs including confusion

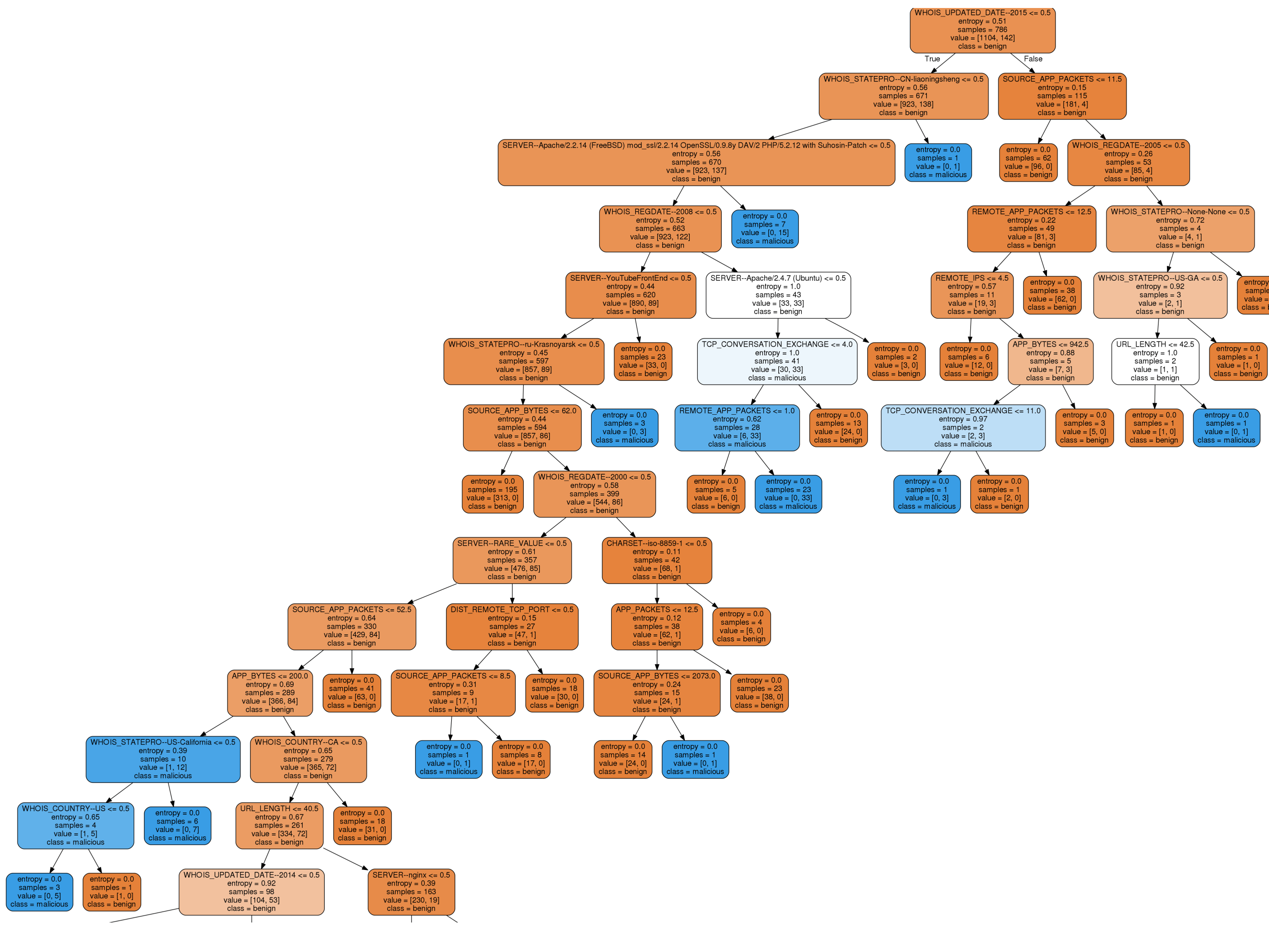
WHOIS_UPDATED_DATE--2015 <= 0.5 entropy = 0.51 samples = 786 value = [1104, 142] class = benign
True
False
WHOIS_STATEPRO--CN-liaoningsheng <= 0.5 entropy = 0.56 samples = 671 value = [923, 138] class = benign
SOURCE_APP_PACKETS <= 11.5 entropy = 0.15 samples = 115 value = [181, 4] class = benign
SERVER--Apache/2.2.14 (FreeBSD) mod_ssl/2.2.14 OpenSSL/0.9.8y DAV/2 PHP/5.2.12 with Suhosin-Patch <= 0.5 entropy = 0.56 samples = 670 value = [923, 137] class = benign
entropy = 0.0 samples = 1 value = [0, 1] class = malicious
entropy = 0.0 samples = 62 value = [96, 0] class = benign
WHOIS_REGDATE--2005 <= 0.5 entropy = 0.26 samples = 53 value = [85, 4] class = benign
WHOIS_REGDATE--2008 <= 0.5 entropy = 0.52 samples = 663 value = [923, 122] class = benign
entropy = 0.0 samples = 7 value = [0, 15] class = malicious
REMOTE_APP_PACKETS <= 12.5 entropy = 0.22 samples = 49 value = [81, 3] class = benign
WHOIS_STATEPRO--None-None <= 0.5 entropy = 0.72 samples = 4 value = [4, 1] class = benign
SERVER--YouTubeFrontEnd <= 0.5 entropy = 0.44 samples = 620 value = [890, 89] class = benign
SERVER--Apache/2.4.7 (Ubuntu) <= 0.5 entropy = 1.0 samples = 43 value = [33, 33] class = benign
REMOTE_IPS <= 4.5 entropy = 0.57 samples = 11 value = [19, 3] class = benign
entropy = 0.0 samples = 38 value = [62, 0] class = benign
WHOIS_STATEPRO--US-GA <= 0.5 entropy = 0.92 samples = 3 value = [2, 1] class = benign
entropy = 0.0 samples = 1 value = [2, 0] class = benign
WHOIS_STATEPRO--ru-Krasnoyarsk <= 0.5 entropy = 0.45 samples = 597 value = [857, 89] class = benign
entropy = 0.0 samples = 23 value = [33, 0] class = benign
TCP_CONVERSATION_EXCHANGE <= 4.0 entropy = 1.0 samples = 41 value = [30, 33] class = malicious
entropy = 0.0 samples = 2 value = [3, 0] class = benign
entropy = 0.0 samples = 6 value = [12, 0] class = benign
APP_BYTES <= 942.5 entropy = 0.88 samples = 5 value = [7, 3] class = benign
URL_LENGTH <= 42.5 entropy = 1.0 samples = 2 value = [1, 1] class = benign
entropy = 0.0 samples = 1 value = [1, 0] class = benign
SOURCE_APP_BYTES <= 62.0 entropy = 0.44 samples = 594 value = [857, 86] class = benign
entropy = 0.0 samples = 3 value = [0, 3] class = malicious
REMOTE_APP_PACKETS <= 1.0 entropy = 0.62 samples = 28 value = [6, 33] class = malicious
entropy = 0.0 samples = 13 value = [24, 0] class = benign
TCP_CONVERSATION_EXCHANGE <= 11.0 entropy = 0.97 samples = 2 value = [2, 3] class = malicious
entropy = 0.0 samples = 3 value = [5, 0] class = benign
entropy = 0.0 samples = 1 value = [1, 0] class = benign
entropy = 0.0 samples = 1 value = [0, 1] class = malicious
entropy = 0.0 samples = 195 value = [313, 0] class = benign
WHOIS_REGDATE--2000 <= 0.5 entropy = 0.58 samples = 399 value = [544, 86] class = benign
entropy = 0.0 samples = 5 value = [6, 0] class = benign
entropy = 0.0 samples = 23 value = [0, 33] class = malicious
entropy = 0.0 samples = 1 value = [0, 3] class = malicious
entropy = 0.0 samples = 1 value = [2, 0] class = benign
SERVER--RARE_VALUE <= 0.5 entropy = 0.61 samples = 357 value = [476, 85] class = benign
CHARSET--iso-8859-1 <= 0.5 entropy = 0.11 samples = 42 value = [68, 1] class = benign
SOURCE_APP_PACKETS <= 52.5 entropy = 0.64 samples = 330 value = [429, 84] class = benign
DIST_REMOTE_TCP_PORT <= 0.5 entropy = 0.15 samples = 27 value = [47, 1] class = benign
APP_PACKETS <= 12.5 entropy = 0.12 samples = 38 value = [62, 1] class = benign
entropy = 0.0 samples = 4 value = [6, 0] class = benign
APP_BYTES <= 200.0 entropy = 0.69 samples = 289 value = [366, 84] class = benign
entropy = 0.0 samples = 41 value = [63, 0] class = benign
SOURCE_APP_PACKETS <= 8.5 entropy = 0.31 samples = 9 value = [17, 1] class = benign
entropy = 0.0 samples = 18 value = [30, 0] class = benign
SOURCE_APP_BYTES <= 2073.0 entropy = 0.24 samples = 15 value = [24, 1] class = benign
entropy = 0.0 samples = 23 value = [38, 0] class = benign
WHOIS_STATEPRO--US-California <= 0.5 entropy = 0.39 samples = 10 value = [1, 12] class = malicious
WHOIS_COUNTRY--CA <= 0.5 entropy = 0.65 samples = 279 value = [365, 72] class = benign
entropy = 0.0 samples = 1 value = [0, 1] class = malicious
entropy = 0.0 samples = 8 value = [17, 0] class = benign
entropy = 0.0 samples = 14 value = [24, 0] class = benign
entropy = 0.0 samples = 1 value = [0, 1] class = malicious
WHOIS_COUNTRY--US <= 0.5 entropy = 0.65 samples = 4 value = [1, 5] class = malicious
entropy = 0.0 samples = 6 value = [0, 7] class = malicious
URL_LENGTH <= 40.5 entropy = 0.67 samples = 261 value = [334, 72] class = benign
entropy = 0.0 samples = 18 value = [31, 0] class = benign
entropy = 0.0 samples = 3 value = [0, 5] class = malicious
entropy = 0.0 samples = 1 value = [1, 0] class = benign
WHOIS_UPDATED_DATE--2014 <= 0.5 entropy = 0.92 samples = 98 value = [104, 53] class = benign
SERVER--nginx <= 0.5 entropy = 0.39 samples = 163 value = [230, 19] class = benign

Fig. 2. One Decision Tree computed by the Random Forest Algorithm

matrix and ROC curve. Also, a classification report is presented which includes the performance results across various metrics including Precision, Recall, F-1 Score, and Support.

A. Confusion Matrix

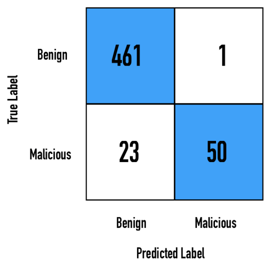

Fig. 3. Confusion Matrix

The Confusion Matrix in Fig. 3 shows that 461 benign and 50 malicious websites are classified perfectly, whereas 1 benign website is classified as malicious and 23 malicious websites are classified as benign. This demonstrates that the Random Forest Classifier is able to nearly-perfectly predict whether a given website is benign in nature, and also demonstrates good performance in the prediction of malicious websites.

B. Receiver Operating Characteristics Curve

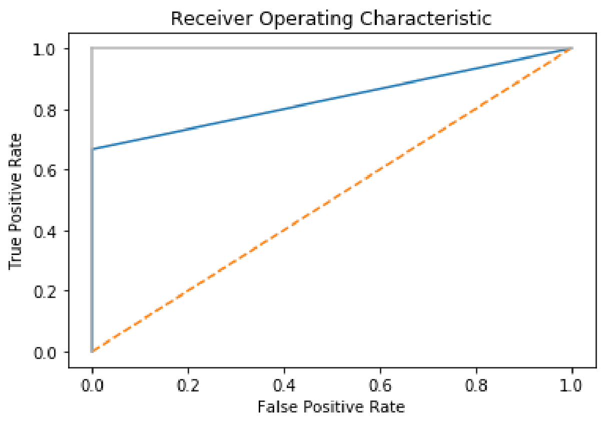

Fig. 4. ROC Curve

The area under the ROC curve (AUC) in Fig. 4. is 84%. The closer the value of AUC to 100%, the better is the performance of the classifier. Thus, the ROC curve further demonstrates the superior performance achieved by the Random Forest Classifier.

C. *Classification Report*

TABLE III
CLASSIFICATION REPORT

Class	Precision	Recall	F-1 Score	Support
Benign	0.95	1.00	0.97	462
Malicious	0.98	0.68	0.81	73
Micro avg	0.96	0.96	0.96	535
Macro avg	0.97	0.84	0.89	535
Weighted avg	0.96	0.96	0.95	535

The classification report of the Random Forest Classifier is presented in Table III. It can be inferred that the classifier demonstrates a very high precision score for both the classes, and achieves a perfect recall value for the benign class of websites. Also, the F-1 scores for both the classes are very high, with 81% F-1 score for Malicious websites.

VI. CONCLUSION AND FUTURE SCOPE

This paper studies the performance of supervised machine learning algorithms for the task of predicting whether a website is malicious or benign based upon its application layer and network layer features. These features are extracted from the header and body of the requests and responses being exchanged between the client and the server. Optimal features for this classification task are determined by computing the feature importance values for each attribute. The research focuses on decision tree and random forest algorithms, and the classification performance is visualized by plotting the ROC curve and Confusion Matrix. The analysis of architecture performance demonstrates that the best classification is achieved with the Random Forest Algorithm with the highest accuracy of 92%, and an AUC score of 0.84. Thus, this paper puts forth an end-to-end software solution for the prediction of malicious websites and preempting naive users from accessing them. We look forward to integrate RakshaNet with DataSec.AI - Data Anonymization Microservice to develop a full-fledged suite of security software capable of determining wether visiting a given website is secure ot not, and then also apply content moderation and data masking to anonymize all the personally identifiable information (PII) from the HTTP/HTTPS requests and responses.

ACKNOWLEDGMENT

We would like to thank our Professor Ms. Smita Dange, Department of Computer Engineering, Fr. C. Rodrigues Institute of Technology, Vashi for her support and guidance.

REFERENCES

[1] Patil, Dharmaraj Patil, Jayantrao. (2015). Survey on Malicious Web Pages Detection Techniques. International Journal of u- and e-Service, Science and Technology. 8. 195-206. 10.14257/ijunesst.2015.8.5.18.

[2] P. Yadav and C. D. Parekh, "A report on CSRF security challenges & prevention techniques," 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), Coimbatore, 2017, pp. 1-4, doi: 10.1109/ICIIECS.2017.8275852.

[3] Zlatanov, Nikola. (2015). Computer Security and Mobile Security Challenges.

[4] Manlin Wang, Fei Zhang and P. P. K. Chan, "Malicious website detection under the exploratory attack," 2013 International Conference on Machine Learning and Cybernetics, Tianjin, 2013, pp. 565-570, doi: 10.1109/ICMLC.2013.6890356.

[5] S. B. Rathod and T. M. Pattewar, "A comparative performance evaluation of content based spam and malicious URL detection in E-mail," 2015 IEEE International Conference on Computer Graphics, Vision and Information Security (CGVIS), Bhubaneswar, 2015, pp. 49-54, doi: 10.1109/CGVIS.2015.7449891.

[6] T. Shibahara, Y. Takata, M. Akiyama, T. Yagi and T. Yada, "Detecting Malicious Websites by Integrating Malicious, Benign, and Compromised Redirection Subgraph Similarities," 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), Turin, 2017, pp. 655-664, doi: 10.1109/COMPSAC.2017.105.

[7] S. Singhal, U. Chawla and R. Shorey, "Machine Learning & Concept Drift based Approach for Malicious Website Detection," 2020 International Conference on COMmunication Systems & NETworkS (COMSNETS), Bengaluru, India, 2020, pp. 582-585, doi: 10.1109/COMSNETS48256.2020.9027485.

[8] Y. -C. Chen, Y. -W. Ma and J. -L. Chen, "Intelligent Malicious URL Detection with Feature Analysis," 2020 IEEE Symposium on Computers and Communications (ISCC), Rennes, France, 2020, pp. 1-5, doi: 10.1109/ISCC50000.2020.9219637.

[9] Urcuqui, C., Navarro, A., Osorio, J., Garcıa, M. (2017). Machine Learning Classifiers to Detect Malicious Websites. CEUR Workshop Proceedings. Vol 1950, 14-17.

[10] Ashraf, Imran Hur, Soojung Park, Yongwan. (2017). An Investigation of Interpolation Techniques to Generate 2D Intensity Image from LIDAR Data. IEEE Access. PP. 1-1. 10.1109/ACCESS.2017.2699686.

[11] Ronaghan, Stacey. 'The Mathematics of Decision Trees, Random Forest and Feature Importance in Scikit-Learn and Spark'. Medium, 1 Nov. 2019, https://towardsdatascience.com/the-mathematics-of-decision-trees-random-forest-and-feature-importance-in-scikit-learn-and-spark-f2861df67e3.

[12] Budiman, Edy & Haviluddin, Haviluddin & Dengan, Nataniel & Kridalaksana, Awang & Wati, Masna & Purnawansyah,. (2018). Performance of Decision Tree C4.5 Algorithm in Student Academic Evaluation. 10.1007/978-981-10-8276-4_36.

[13] R. Rivera-Lopez and J. Canul-Reich, "Construction of Near-Optimal Axis-Parallel Decision Trees Using a Differential-Evolution-Based Approach," in IEEE Access, vol. 6, pp. 5548-5563, 2018, doi: 10.1109/ACCESS.2017.2788700.

[14] Y. Liu, L. Liu, Y. Gao and L. Yang, "An Improved Random Forest Algorithm Based on Attribute Compatibility," 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chengdu, China, 2019, pp. 2558-2561, doi: 10.1109/ITNEC.2019.8729146.

[15] A complete guide to the random forest algorithm. (2019, June 16). Built In. https://builtin.com/data-science/random-forest-algorithm