



BIRMINGHAM CITY
University

Fast Detection of Zero-Day Phishing Websites Using Machine Learning

Thomas P. Nagunwa

*A thesis submitted in partial fulfilment of the requirements for the degree of
Doctor of Philosophy*

School of Computing and Digital Technology
Faculty of Computing, Engineering and the Built Environment

January 2022

Abstract

The recent global growth in the number of internet users and online applications has led to a massive volume of personal data transactions taking place over the internet. In order to gain access to the valuable data and services involved for undertaking various malicious activities, attackers lure users to phishing websites that steal user credentials and other personal data required to impersonate their victims. Sophisticated phishing toolkits and flux networks are increasingly being used by attackers to create and host phishing websites, respectively, in order to increase the number of phishing attacks and evade detection. This has resulted in an increase in the number of new (zero-day) phishing websites. Anti-malware software and web browsers' anti-phishing filters are widely used to detect the phishing websites thus preventing users from falling victim to phishing. However, these solutions mostly rely on blacklists of known phishing websites. In these techniques, the time lag between creation of a new phishing website and reporting it as malicious leaves a window during which users are exposed to the zero-day phishing websites. This has contributed to a global increase in the number of successful phishing attacks in recent years.

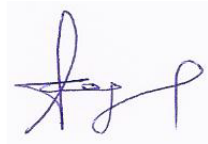
To address the shortcoming, this research proposes three Machine Learning (ML)-based approaches for fast and highly accurate prediction of zero-day phishing websites using novel sets of prediction features. The first approach uses a novel set of 26 features based on URL structure, and webpage structure and contents to predict zero-day phishing webpages that collect users' personal data. The other two approaches detect zero-day phishing webpages, through their hostnames, that are hosted in Fast Flux Service Networks (FFSNs) and Name Server IP Flux Networks (NSIFNs). The networks consist of frequently changing machines hosting malicious websites and their authoritative name servers respectively. The machines provide a layer of protection to the actual service hosts against blacklisting in order to prolong the active life span of the services. Consequently, the websites in these networks become more harmful than those hosted in normal networks. Aiming to address them, our second proposed approach predicts zero-day phishing hostnames hosted in FFSNs using a novel set of 56 features based on DNS, network and host characteristics of the hosting networks. Our last approach predicts zero-day phishing hostnames hosted in NSIFNs using a novel set of 11 features based on DNS and host characteristics of the hosting networks.

The feature set in each approach is evaluated using 11 ML algorithms, achieving a high prediction performance with most of the algorithms. This indicates the relevance and robustness of the feature sets for their respective detection tasks. The feature sets also perform well against data collected over a later time period without retraining the data, indicating their long-term effectiveness in detecting the websites. The approaches use highly diversified feature sets which is expected to enhance the resistance to various detection evasion tactics. The measured prediction times of the first and the third approaches are sufficiently low for potential use for real-time protection of users. This thesis also introduces a multi-class classification technique for evaluating the feature sets in the second and third approaches. The technique predicts each of the hostname types as an independent outcome thus enabling experts to use type-specific measures in taking down the phishing websites. Lastly, highly accurate methods for labelling hostnames based on number of changes of IP addresses of authoritative name servers, monitored over a specific period of time, are proposed.

Declaration of Authorship

I, **Thomas P. Nagunwa**, confirm that this thesis is entirely my own work and based on my own research; that all sources used are appropriately acknowledged and that where the words of others are used these are clearly placed in quotation marks. No material contained in the thesis has been used in any other submission for an academic award. I have published material relating to this research previously, and reference is made in the thesis to all such publications, copies of which are submitted with the thesis.

Signed:

A handwritten signature in blue ink, appearing to be 'T. P. Nagunwa', enclosed in a light blue rectangular box.

Dated: **5th January 2022**

Acknowledgement

Undertaking this PhD programme has been a unique and life-changing experience for me and it would not have been possible to do without the support and guidance that I received from many people and Institutions. First and foremost, I am humbly thankful to the Commonwealth Scholarship Commission and Birmingham City University for funding this programme. I am extremely grateful to my supervision team Dr. Syed Naqvi, Prof. Paul Kearney, Dr. Shereen Fouad and Prof. Hanifa Shah for their invaluable technical advice and support, and patience during the course of this research. Their immense knowledge and plentiful experience have helped to shape this research to produce a quality thesis. I would also like to thank all the staff members in the Faculty of Computing, Engineering and the Built Environment. It is their kind help and support that have made my study and life in the UK a wonderful time. Finally, I would like to express my gratitude to my parents, my wife and my children. Without their tremendous understanding and encouragement in the past few years, it would be impossible for me to complete this programme. Finally, many thanks to my colleagues at the School of Computing and Digital Technology for their direct and indirect support that made this journey one of the most memorable times of my life.

Table of Contents

Abstract	i
List of Figures	ix
List of Tables.....	xv
Abbreviations	xvii
Chapter 1: Introduction.....	1
1.1. Research Background.....	1
1.1.1. Background on Phishing.....	1
1.1.2. Existing Solutions for Preventing Phishing Website Attacks.....	4
1.2. Problem Statement and Motivation.....	7
1.3. Aim and Objectives.....	8
1.4. Contributions.....	9
1.5. Research Scope	11
1.6. Thesis Structure.....	11
Chaper 2: Phishing Background	13
2.1. Introduction	13
2.2. Overview of Phishing.....	13
2.2.1. Techniques for Distributing Phishing Websites	14
2.2.2. Stages of Executing Phishing Website Attacks	17
2.2.3. Current Trends in Phishing Website Attacks.....	18
2.3. Structural Characteristics of Data Capturing Webpages and Their Hosting Networks	
24	
2.3.1. Anatomy of a Webpage Collecting Personal Data	24
2.3.2. Domain Name System (DNS).....	30
2.3.3. Structural Characteristics of Networks Hosting Phishing and Legitimate	
Websites33	

2.4. Summary	42
Chapter 3: Machine Learning Techniques for Phishing Website Detection.....	44
3.1. Introduction	44
3.2. Overview of Machine Learning	45
3.3. Machine Learning for Classification Tasks	46
3.3.1. Binary and Multi-class Classification	47
3.3.2. Flat and Hierarchical Classification.....	47
3.4. A Standard Workflow for Machine Learning Tasks	50
3.4.1. Formulation of a Prediction Hypothesis/Question.....	50
3.4.2. Data Collection	50
3.4.3. Data Exploration and Pre-processing.....	51
3.4.4. Selection and Training of ML Algorithms.....	54
3.4.5. Evaluation of a Prediction Model	55
3.4.6. Hyperparameter Tuning	57
3.4.7. Prediction Model Deployment.....	57
3.5. Machine Learning Algorithms for Classification Tasks	58
3.5.1. Traditional Machine Learning Algorithms for Classification Tasks	58
3.5.2. Deep Learning Algorithms for Classification Tasks	65
3.6. Summary	68
Chapter 4: Detection of Zero-Day Phishing Webpages Using Machine Learning.....	70
4.1. Introduction	70
4.2. Related Works	71
4.2.1. Blacklist and Whitelist Based Approach	71
4.2.2. Visual Similarity Approach	72
4.2.3. Offense Defensive Approach.....	73
4.2.4. Rule Based Approach	74

4.2.5.	Hybrid Approach	75
4.3.	Limitations of Related Works	76
4.4.	Prediction Model Design.....	78
4.4.1.	Phishing Webpage Prediction Features	78
4.4.2.	System Architecture of the Prediction Model.....	93
4.5.	Experiments.....	95
4.5.1.	Experimental Setup.....	96
4.5.2.	Data Collection	96
4.5.3.	Data Pre-processing	97
4.5.4.	Performance Results	98
4.5.5.	Detection Time Analysis.....	108
4.5.6.	Model Validation Using New Data.....	110
4.6.	Discussions.....	110
4.6.1.	Comparison with Existing Works.....	110
4.6.2.	Application of the Proposed Model.....	113
4.7.	Summary	114
Chapter 5: Detection of Zero-Day Phishing Hostnames Hosted in Fast Flux and Name Server IP Flux Networks Using Machine Learning		116
5.1.	Introduction	116
5.2.	Prediction of Phishing Hostnames Hosted in FFSNs.....	118
5.2.1.	Related Works.....	118
5.2.2.	Limitations of Related Works.....	120
5.2.3.	Monitoring and Analysis of Hostnames and Their Networks	122
5.2.4.	Features for Prediction of FF Phishing Hostnames	137
5.2.4.1.	Temporal Features.....	140
5.2.4.2.	Spatial Features	141
5.2.4.3.	DNS Features	141

5.2.4.4.	Network Features	142
5.2.4.5.	Host Features.....	143
5.2.4.6.	Reputation Features.....	143
5.2.5.	System Architecture of the Prediction Model.....	144
5.2.6.	Experiments	145
5.2.7.	Discussions	165
5.3.	Prediction of Phishing Hostnames Hosted in NS IP Flux Networks	171
5.3.1.	Related Works.....	171
5.3.2.	Limitations of Related Works.....	173
5.3.3.	Monitoring and Network Analysis of Name Servers and Their Networks.....	174
5.3.4.	Features for Prediction of Phishing NS IP Flux Hostnames.....	185
5.3.5.	System Architecture of the Prediction Model.....	186
5.3.6.	Experiments	186
5.3.7.	Discussions	203
5.3.7.1.	Comparison with Existing Works	203
5.3.7.2.	Application of the Proposed Model	205
5.4.	Summary	206
Chapter 6: Conclusions and Future Work.....		208
6.1.	Summary of a Research Background.....	208
6.2.	Research Contributions	209
6.3.	Limitations of Our Work.....	212
6.4.	Future Work	213
References.....		216
Appendixes		236

List of Figures

Figure 1.1. Taxonomy of existing approaches to prevent phishing website attacks.....	5
Figure 2.1. A generic phishing email pretending to be from PayPal [64].	15
Figure 2.2. Typical stages of a phishing website attack.	18
Figure 2.3. Percentage of phishing attacks that were using TLS certificates between 2015 and 2019 [93].....	20
Figure 2.4. The growth of rate of phishing websites hosted in free web hosting services [23].	21
Figure 2.5. Distribution of domain age of phishers-owned domains prior to phishing attacks [39].....	22
Figure 2.6a. A legitimate PayPal webpage.	28
Figure 2.6b. A PayPal phishing webpage.	28
Figure 2.7. An example of a domain tree for the FQDN <i>cs.berkeley.edu</i> . [121].....	31
Figure 2.8. An example of a DNS recursive query operation.....	32
Figure 2.9. A round robin mechanism.	32
Figure 2.10. Architecture of CDNs.....	35
Figure 2.11. Architecture of FFSNs.....	37
Figure 2.12. Architecture of NSIFs.....	41
Figure 3.1. A chart summarizing the categorization of ML.	46
Figure 3.2a. No parent-child relationships in flat classification.	48
Figure 3.2b. Parent-child relationships in hierarchical classification.	48
Figure 3.3. LCPN approach (dashed squares represent binary or multi-class classifiers).....	49
Figure 3.4. A summary of key steps for developing an ML prediction model.....	50
Figure 3.5. ROC curve of a model showing its area under a curve in a shaded part.	57
Figure 3.6. An example of linear decision boundaries created by a linear function for separating instances of three classes.....	59
Figure 3.7. An SVM classifier using support vectors to determine a decision boundary for a binary classification task.	60
Figure 3.8. The structure of a neuron.....	63
Figure 3.9. The network structure of a simple ANN with three input variables and three neurons of a hidden layer.....	64
Figure 3.10. The structure of a memory cell of LSTM.....	67

Figure 3.11. The network structure of 1D CNN with one convolution layer and one fully connected layer of neuron.....	68
Figure 4.1. A system architecture of the proposed model for predicting zero-day phishing PDC webpages.....	95
Figure 4.2. ROC curves of the trained standard ML algorithms.	99
Figure 4.3a. FC-DNN architecture of the classifier.	101
Figure 4.3b. LSTM architecture of the classifier.....	101
Figure 4.3c. 1D CNN architecture of the classifier.	102
Figure 4.4a. Accuracy rates of the feature categories.....	103
Figure 4.4b. Error rates of the feature categories.....	103
Figure 4.5a. Accuracy rates of the existing and novel features.	104
Figure 4.5b. Error rates of the existing and novel features.....	104
Figure 4.7a. Data distribution of counts of hostname’s IP address matching with phishing blacklisted IP addresses.....	107
Figure 4.7b. Data distribution of the matching of hostnames in a search engine’s results....	107
Figure 4.7c. Data distribution of counts of domain identify appearing in the webpage structure and contents.	108
Figure 4.7d. Data distribution of domain age.	108
Figure 4.7e. Data distribution of types of TLS certificate.	108
Figure 4.7f. Data distribution of counts of characters in a URL path.....	108
Figure 4.8. Comparison of extraction times of the best features.	109
Figure 5.1. Monitoring of A records for 5 weeks for labelling classes of the hostnames.	123
Figure 5.2. Lifespans of the collected phishing hostnames during the monitoring period....	124
Figure 5.3. Distribution of number of changes of IP addresses of hosts observed per hostname type. Note that the data bins are not all of equal size.	125
Figure 5.4. The number of cumulative unique IP addresses of FF phishing and CDN hostnames over the monitoring time.	126
Figure 5.5. Distribution of number of changes of IP addresses versus number of unique IP addresses over the monitoring period for FF phishing hostnames.	127
Figure 5.6. Distribution of number of changes of IP addresses versus number of unique IP addresses over the monitoring period for CDN hostnames.....	127
Figure 5.7. Distribution of unique number of subnets of flux and non-flux hostnames.....	128
Figure 5.8. Distribution of number of unique networks of flux and non-flux hostnames.	129

Figure 5.9. Distribution of number of unique ASs of flux and non-flux hostnames.	129
Figure 5.10. Top AS organizations hosting servers of phishing hostnames.	130
Figure 5.11. Top AS organizations hosting servers of legitimate hostnames.	131
Figure 5.12. Distribution of number of unique countries of the observed hostnames.	132
Figure 5.13. Top countries hosting servers of the legitimate hostnames.	133
Figure 5.14. Top countries hosting servers of the phishing hostnames.	133
Figure 5.15. Sizes of 12 largest phishing FFSNs as a percentage of the total. Network IDs were allocated by Gephi.	134
Figure 5.16. Absolute sizes of 25 largest networks. Columns are split based on node type.	134
Figure 5.17. Visual graph of the largest FFSN with 2275 nodes, of which 51 are hostnames and 2224 are IP addresses.	135
Figure 5.18. Hostnames in network 29 ordered by out-degree.	136
Figure 5.19. Frequencies of in-degree counts of IP addresses in network 29.	136
Figure 5.20. A system architecture of our proposed model for the prediction of FF phishing hostnames.	144
Figure 5.21a. Architecture A - flat multi-class classification.	146
Figure 5.21b. Architecture B - flat binary classification.	146
Figure 5.21c. Architecture C – hierarchical classification.	146
Figure 5.21d. Architecture D - hierarchical classification.	147
Figure 5.22a. ROC curves of the traditional ML algorithms for classifier A.	150
Figure 5.22b. ROC curves of the traditional ML algorithms for classifier B.	150
Figure 5.22c. ROC curves of the traditional ML algorithms for classifier C.1.	150
Figure 5.22d. ROC curves of the traditional ML algorithms for classifier C.2.	150
Figure 5.22e. ROC curves of the traditional ML algorithms for classifier D.1.	150
Figure 5.22f. ROC curves of the traditional ML algorithms for classifier D.2.	150
Figure 5.23c. 1D CNN architecture of classifier B.	153
Figure 5.24. Importance weights of the best features of classifier B.	158
Figure 5.25a. Distribution of average of total number of occurrences of hosts' IP addresses of the four types of hostnames in a list of IP addresses of blacklisted phishing websites. .	158
Figure 5.25b. Distribution of average of total number of occurrences of hosts' IP addresses of two types of hostnames in classifiers B in a list of IP addresses of blacklisted phishing websites.	158

Figure 5.26a. Distribution of differences of edit distances (Levenshtein) between FF phishing hostnames and each of the four types of the hostnames.	159
Figure 5.26b. Distribution of differences of edit distances between FF phishing hostnames and each of the two types of the hostnames in classifier B.	159
Figure 5.27a. Distribution of TTLs of the four types of hostnames.	159
Figure 5.27b. Distribution of TTLs of the two types of hostnames in classifier B.	159
Figure 5.28a. Distribution of average domain age of the four types of hostnames.	160
Figure 5.28b. Distribution of average domain age of hostnames of the two types of hostnames in classifier B.	160
Figure 5.29a. Distribution of average number of hops between user and hosts of a hostname in each of the four types of hostnames.	160
Figure 5.29b. Distribution of average number of hops between user and hosts of a hostname in each of the two types of hostnames.	160
Figure 5.30a. Distribution of average number of unique co-hosted websites in the hostname's hosts for all four types of hostnames.	161
Figure 5.30b. Distribution of average number of unique co-hosted websites in the hostname's hosts for the two types of hostnames in classifier B.	161
Figure 5.31a. Comparison of accuracy rates of feature categories of classifier B.	162
Figure 5.31b. Comparison of error rates of feature categories of classifier B.	162
Figure 5.31c. Comparison of accuracy rates between novel, adopted and overall features of classifier B.	163
Figure 5.31d. Comparison of error rates between novel, adopted and overall features of classifier B.	163
Figure 5.32. Distribution of feature extraction times by activities.	164
Figure 5.33. Monitoring of A records of NSs for labelling classes of the hostnames.	175
Figure 5.34. Distribution of number of changes of IP addresses of authoritative NSs observed per NS type.	176
Figure 5.35. The change in the number of cumulative unique IP addresses of fluxing NSs of phishing hostnames.	177
Figure 5.36. Distribution of number of unique networks of NSs of phishing hostnames.	178
Figure 5.37. Distribution of number of unique ASs networks of NSs of phishing hostnames.	178
Figure 5.38. Top AS organizations hosting NSs of phishing hostnames.	179

Figure 5.39. Top AS organizations hosting NSs of legitimate non-flux hostnames.....	179
Figure 5.40. Distribution of number of unique hosting countries of NSs of the hostnames.	180
Figure 5.41. Top countries hosting NSs of phishing hostnames.....	181
Figure 5.42. Top countries hosting NSs of legitimate hostnames.	181
Figure 5.43. Number of changes of IP addresses observed in double fluxing phishing hostnames.	182
Figure 5.44. Cumulative number of unique IP addresses mapped against the double fluxing phishing hostnames.....	183
Figure 5.45. Cumulative number of unique IP addresses of double fluxing phishing hostnames at various stages of the monitoring period.....	183
Figure 5.46. Number of the shared IP addresses between FFSNs and NSIFNs hosting the double fluxing phishing hostnames.	184
Figure 5.47a. Architecture X - flat classification-based architecture.	187
Figure 5.47b. Architecture Y - flat classification-based architecture.	187
Figure 5.47c. Architecture Z – hierarchical classification-based architecture.....	188
Figure 5.48a. ROC curves of the traditional ML algorithms for classifier X.	190
Figure 5.48b. ROC curves of the traditional ML algorithms for classifier Y.....	190
Figure 5.48c. ROC curves of the traditional ML algorithms for classifier Z.1.	191
Figure 5.48d. ROC curves of the traditional ML algorithms for classifier Z.2.	191
Figure 5.49c. 1D CNN architecture of classifier Y.	193
Figure 5.50. Ranking by importance weights of the best features of classifier Y. Numbers in the brackets represent numbers of the features as indicated in Table 5.12.	196
Figure 5.51a. Percentage of hostnames per each class with their NSs installed with webservers.	197
Figure 5.51b. Percentage of hostnames of classifier Y with their NSs installed with webservers.	197
Figure 5.52a. Distribution of average number of unique co-hosted websites in the hostname’s NSs for each hostname class.	197
Figure 5.53a. Distribution of average uptime of NSs of hostnames per each class.....	198
Figure 5.53b. Distribution of average uptime of NSs of hostnames in classifier Y.	198
Figure 5.54a. Distribution of number of network hops between user and NS hosts of the three hostname classes.....	198

Figure 5.54b. Distribution of number of network hops between user and NS hosts of the two classes of hostnames in classifier Y.	198
Figure 5.55a. Distribution of registrars of NS records of the three classes of hostnames.	199
Figure 5.55b. Distribution of registrars of NS records of the two classes of hostnames in classifier Y.....	199
Figure 5.56a. Percentage distribution of TTLs of NS records of hostnames for the three hostname classes.....	200
Figure 5.56b. Percentage distribution of TTLs of NS records of hostnames in classifier Y.	200
Figure 5.57a. Comparison of accuracy rates of feature categories of classifier Y.	201
Figure 5.57b. Comparison of error rates of feature categories of classifier Y.....	201
Figure 5.58. Distribution of feature extraction times by activities.	202

List of Tables

Table 2.1. Types of phishing websites categorized based on the three similarity metrics [117].	28
Table 2.3. A summary of the expected differences in the structural and operational characteristics of the four types of the website hosting networks.	39
Table 2.4. The expected structural and operational differences between the three types of networks hosting authoritative NSs.	42
Table 4.1. Common PDC phrases used in PDC webpages.	79
Table 4.2. Summary of the proposed features.	81
Table 4.3. Summary of the collected data used to build the training dataset.	97
Table 4.4. Features with missing values.	98
Table 4.6. Values of hyperparameters of the tuned RF.	99
Table 4.7. Hyperparameters and their ranges of values evaluated for tuning the three DL algorithms.	100
Table 4.8. Performance results of the DL algorithms.	101
Table 4.10. Performances of various combinations of the feature categories of the best features.	104
Table 4.11. Runtimes of the model’s modules.	109
Table 4.12. Performance comparison of some of the related works with our work. The reported performances are of the best performing algorithms in bold.	111
Table 5.1. A list of the proposed features for predicting phishing FF hostnames.	139
Table 5.2. Sources of data for each feature.	140
Table 5.3. Classifiers that are useful for the prediction of FF phishing hostnames in each architecture.	147
Table 5.4. Classes and dataset sizes of training datasets used for each classifier.	148
Table 5.5a. Performance results of top four best performing traditional ML algorithms for classifiers A, B and C.1.	151
Table 5.5b. Performance results of top four best performing traditional ML algorithms for classifiers C.2, D.1 and D.2.	151
Table 5.6. The optimal values of the tuned RF hyperparameters for classifier B.	151
Table 5.7a. Performance results of the evaluated DL algorithms for classifiers A, B and C.1.	152

Table 5.7b. Performance results of the evaluated DL algorithms for classifiers C.2, D.1 and D.2.	152
Table 5.8. Prediction performances of the model architectures.....	154
Table 5.9. Composition of categories in the best feature set of classifier B.....	162
Table 5.10. Runtimes of the classifier B’s three stages for predicting FF phishing hostnames.	164
Table 5.11. A summarized comparison chart between our work and the related works.	166
Table 5.12. A list of proposed features for predicting phishing NS IP flux hostnames.	186
Table 5.13. Classifiers that are useful for the prediction of phishing NS flux hostnames in each architecture.	188
Table 5.14. Classes and dataset sizes of training datasets used for the classifiers.	189
Table 5.15a. Performance results of top four best performing ML algorithms for classifiers X and Y.....	191
Table 5.15b. Performance results of top four best performing ML algorithms for classifiers Z.1 and Z.2.....	191
Table 5.17a. Performance results of the evaluated DL algorithms for classifiers X and Y...	192
Table 5.17b. Performance results of the evaluated DL algorithms for classifiers Z.1 and Z.2.	192
Table 5.18. Prediction performances of the model architectures.....	194
Table 5.19. Composition of categories in the best feature set of classifier Y.....	201
Table 5.20. Runtimes of the classifier Y’s three stages for predicting phishing NS flux hostnames.	202
Table 5.21. Performance comparison of some of the related works with our work.	205

Abbreviations

ID CNN	One Dimensional Convolution Neural Network
AGD	Algorithmically Generated Domain
AI	Artificial Intelligence
ANN	Artificial Neural Network
APWG	Anti-Phishing Working Group
AS	Autonomous System
ASCII	American Standard Code for Information Interchange
ASN	Autonomous System Number
AUC	Area Under Curve
BEC	Business Email Compromise
CA	Certificate Authority
CDN	Content Delivery Network
CNAME	Canonical Name
DKIM	DomainKeys Identified Mail
DL	Deep Learning
DMARC	Domain-based Message Authentication, Reporting and Conformance
DNS	Domain Name System
DT	Decision Tree
DUL	Dynamic User List
DV	Domain Validation
ED	Edit Distance
EDA	Exploratory Data Analysis
EV	Extended Validation
FBI	Federal Bureau of Investigations
FC-DNN	Fully Connected Feedforward Deep Neural Network
FF	Fast Flux
FFSN	Fast Flux Service Network
FH	Form Handler
FNR	False Negative Rate
FPR	False Positive Rate

FQDN	Full Qualified Domain Name
FTC	Federal Trade Commission
GB	Gradient Boosting
HTML	Hyper Text Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IANA	Internet Assigned Numbers Authority
IG	Information Gain
IoT	Internet of Things
IP	Internet Protocol
JI	Jaccard Index
KL	Kullback-Leibler Distance
k-NN	k-Nearest Neighbour
LCL	Local Classifier per Level
LCN	Local Classifier per Node
LCPN	Local Classifier per Parent Node
LR	Logistic Regression
LSTM	Long Short-Term Memory
ML	Machine Learning
MX	Mail Exchange
NB	Naïve Bayes
NCSC	National Cyber Security Centre
NS	Name Server
NSIFN	Name Server IP Flux Network
NSNF	Name Server Name Flux
OS	Operating System
OV	Organization Validation
OVA	One-vs-All
OVO	One-vs-One
PaaS	Phishing-as-a-Service
PDC	Personal Data Capturing
PTR	Pointer

RF	Random Forest
RL	Reinforcement Learning
ROC	Receiver Operating Characteristics
RR	Resource Record
RTT	Round Trip Time
SEP	Search Engine Poisoning
SL	Supervised Learning
SLD	Second level Domain Name
SMOTE	Synthetic Minority Over-sampling Technique
SMS	Short Message Service
SORBS	Spam and Open Relay Blocking System
SSL	Secure Sockets Layer
SURBL	Spam Uniform Resource Identifier Realtime Block List
SVM	Support Vector Machine
TLS	Transport Layer Security
TTL	Time to Live
UL	Unsupervised Learning
URIBL	Uniform Resource Identifier Black Listing
URL	Uniform Resource Locator
WEKA	Waikato Environment for Knowledge Analysis

Chapter 1

Introduction

1.1. Research Background

1.1.1. Background on Phishing

The rapid growth of global internet usage in recent years has led to a boom in online services in domains such as e-commerce, social networking and e-government. This has resulted in a surge in the volume of transactions of sensitive information such as personal data in the internet. Online availability of such data has lured attackers to devise a cyberattack mechanism, known as *phishing*, to enable them to steal the data and use it to impersonate the victims for undertaking malicious activities. Various studies have provided different definitions of phishing based on specific contexts. APWG [1], for instance, defined phishing as “a crime employing both social engineering and technical subterfuge to steal consumers’ personal identity data and financial account credentials”. Though this definition is widely accepted, it focuses on the financial data excluding other non-financial data such as family, health and employment records. We therefore provide a modified version of this definition. We define phishing as *a form of social engineering tactic employed by an attacker that uses technological means to acquire personal data from an individual or organization*. Social engineering refers to the psychological tactic of tricking users into performing actions believing that they are engaging with a legitimate party. Examples of this are phony phone calls and emails. Technological means are technological tools that are used to engage with users in order to collect the data. Examples of these are fake websites and malware. Popular data targeted by phishing attacks includes usernames, passwords, social security numbers, credit card details and bank account numbers. Common motives for phishing include theft of money, execution of other cyberattacks and cyber-espionage.

Phishing is still one of the most important and effective types of cyberattacks and it has been growing steadily since it was first perceived in 1996. Over the years, phishing attacks have caused significant economic, political and social impacts to individuals, organizations and governments. These include:

- *Financial losses.* The losses can be caused through direct and indirect means. In the former, stolen personal data obtained through phishing is used to compromise finance-related accounts of users and organizations, allowing attackers to steal money. The latter are the financial consequences of other events caused by phishing such as data breach. In the case of data breach, for instance, these include revenues lost due to the shutting down of web services, the costs for conducting forensic investigations and auditing to determine root causes and impacts of the event, pay outs of compensations to the victimized customers, penalties from authorities and damage to the business reputation which often results in the drop of market values and number of customers [2-6]. Consequently, up to 60% of small and medium sized businesses which have experienced significant data breaches go out of business permanently [7]. Estimated financial losses due to phishing have been reported by various studies. For instance Federal Bureau of Investigation (FBI) reported that between 2013 and 2018 global losses due to spear phishing (a specific type of phishing) alone reached \$12 billion [8].
- *Theft of proprietary and confidential information.* In these attacks, the compromised credentials obtained through phishing provide access to the targeted systems, lead to theft or leakage of confidential information. High profile cyber-espionage incidents in recent years such as theft of research works from various universities around the globe in 2019 were the results of phishing [9]. Other attacks aim at stealing secrets of businesses and governments [10], resulting in loss of competitive advantage or profits and breaches of national security, among other things [11].
- *Enabling of other cyberattacks.* The compromised credentials of administrative accounts of systems enable attackers to execute attacks such as ransomware within the systems or to distribute the attacks to other machines within the organisation's network [12]. In general, phishing is responsible for up to 93% of all global data breaches [13]. Recent cyber-attacks on critical infrastructures such as electricity and nuclear plants were caused by attackers acquiring accesses to the controlling systems through phishing [14, 15]. Phishing contributes to 71% of all such attacks [16].
- *Interfering democracy.* The compromised credentials have been used to influence elections through various ways, including exposing confidential information that discredits some of the candidates and altering of votes in the voting systems [17-20].

The execution of a successful phishing attack involves two main stages namely the distribution/delivery of the attack to the targeted users and the capturing of victims' data. Attackers use various techniques to distribute the attacks to their targets. The most popular one is phishing emails [13, 21]. These are phony emails that are crafted to entice their targets to do certain actions that lead to theft of data or money. Verizon [22] reported that the emails deliver 96% of all phishing attacks while malware injected websites and phones (i.e calls and text messages) deliver 3% and 1% of the attacks respectively. Other significant techniques are social media posts, DNS poisoning and search engine poisoning.

In the second stage, the use of phishing websites remains to be the main tactic used by attackers to capture the data [23]. Other common techniques include the use of phone calls to directly ask user for the data, the use of phishing emails to prompt users to directly pass the data as email responses and the injection of malware into the application to steal the stored or transmitted data. A phishing website is often a replica of a legitimate website that prompts users for similar personal data to that requested by the legitimate website. The majority of phishing websites are delivered through phishing emails to lure potential victims [13, 21]. PhishLabs [23], for instance, revealed that 65% of all the phishing emails, estimated by Valimail [24] at 3.4 billion per day worldwide, involve credential theft in which 88% of these is through phishing websites. In a typical phishing website attack distributed by email, an attacker creates a legitimately looking email with a message to entice users to click on a link that takes them to a phishing website. The website then invites them to submit their personal data for some apparently-legitimate reason. The submitted information is then sent to the attacker.

Today, with the help of highly sophisticated and automated phishing toolkits, which are widely available at a low cost, high-quality phishing websites are being developed even by technically unskilled attackers. With a few clicks, the toolkits allow attackers to create phishing websites with the same look and feel as the legitimate ones, create a large number of variants of the same website and incorporate several detection evasion techniques in the websites [25-28]. The use of these toolkits have resulted into the rise of number of *zero-day phishing websites* created on daily basis [13]. These are newly created websites which have not been discovered by cybersecurity communities.

1.1.2. Existing Solutions for Preventing Phishing Website Attacks

In an attempt to protect users from phishing website attacks, several types of solutions have been used or proposed to intervene in the attacks at various stages, as summarized in Figure 1.1. These are grouped into two main categories, those that aim to prevent delivery of the attacks to the targets and those attempting to prevent users from accessing phishing webpages (through clicking a link in a phishing email, for instance). There are three main types under the former category namely email authentication standards, anti-spam email filters and phishing awareness training. Email authentication standards such as DomainKeys Identified Mail (DKIM) and Domain-based Message Authentication, Reporting and Conformance (DMARC) [29] are security protocols which an organization can use to define policies at the email gateways to authenticate the legitimacy of the email senders, thus filtering out incoming emails from suspected malicious senders. Though they have shown to be effective for the task, their deployment rate in organizations worldwide is still low (at 20%) because of the difficulty in configuring the filtering policies such that legitimate emails are also not filtered out [24]. Anti-spam email filters such as Mailwasher¹ and SpamTitan² are used to detect phishing emails in order to prevent them from getting into the email inboxes of users. They filter the emails by using mainly blacklists of IP addresses, domains or email addresses known to send spams or by behavioural analysis based on the email structure and contents. The attackers, however, have been able to evade the techniques used by these solutions by frequently adapting to new ways of creating the emails, making them ineffective to filter all the phishing emails [13, 27, 30]. Consequently, up to 30% of the total phishing emails sent successfully penetrate to the inboxes [13]. In order to prevent users from reading these emails, various phishing awareness training programs have been designed and used to educate users on how to spot them. However, studies including Bowen, et al. [31], Williams, et al. [32] and Reinheimer, et al. [33] demonstrated that a significant number of users are still unable to spot the emails and fall prey to the attacks even after receiving the training. In addition, Berggren [34] observed that 1 in 3 users would still open emails even if they know that the emails are malicious. This suggests that training has a limited effect. Also, such programs are accessible only to employees of a small number of organisations [35] while home users are unlikely to get access at all.

¹ <https://www.mailwasher.net>

² <https://www.spamtitan.com/anti-spam-software/>

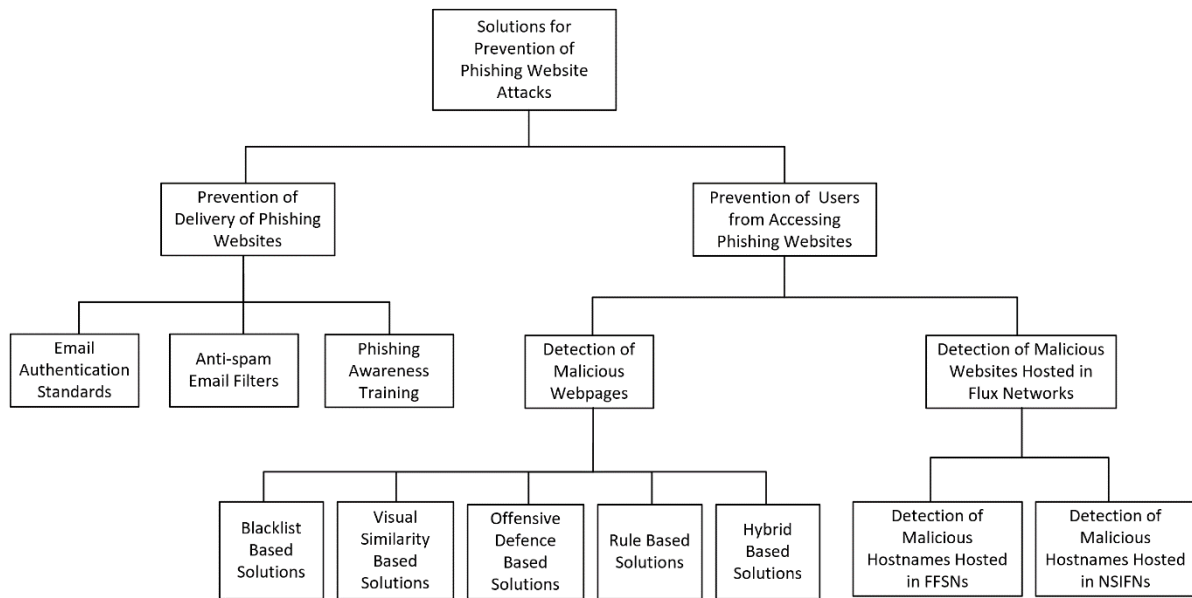


Figure 1.1. Taxonomy of existing approaches to prevent phishing website attacks.

The second category consists of two groups of solutions. The first group are those which detect phishing webpages using information related to their webpage structure and contents, and URLs. In this group, anti-phishing filters are the most popular type of solutions often deployed to protect end users. Most existing filters, including those built into web browsers (e.g Microsoft’s SmartScreen³ and Google Chrome’s Safe Browsing⁴), web browser plug-ins (e.g Bitdefender TrafficLight⁵ and Netcraft Anti-phishing Extension⁶) and anti-malware software (e.g ESET⁷ and Kaspersky⁸) mainly use blacklists to filter the websites [36]. The blacklists are databases that contain lists of URLs or IP addresses of known phishing webpages. The technique, however, is not effective at detecting freshly-created phishing webpages because it often depends on manual operations to report and confirm the webpages, which causes delays in updating the databases [37, 38]. By the time the webpages are added to the database, a significant number of users will have been affected already. Phishing webpages often require to be operational only for a short time to victimize users [26, 39-41].

³ <https://support.microsoft.com/en-us/microsoft-edge/what-is-smartscreen-and-how-can-it-help-protect-me-1c9a874a-6826-be5e-45b1-67fa445a74c8>

⁴ <https://developers.google.com/safe-browsing>

⁵ <https://dottech.org/132882/review-bitdefender-trafficlight/>

⁶ <https://www.netcraft.com/cybercrime/malicious-site-feeds/>

⁷ <https://www.eset.com/us/anti-phishing/>

⁸ https://media.kaspersky.com/pdf/Kaspersky_Lab_Whitepaper_Anti_phishing.pdf

Researchers have also proposed other techniques to detect phishing webpages. One is based on visual similarity whereby images of suspicious webpages alone or in combination with webpage structure and contents are compared against those of pre-collected legitimate webpages using computed visual similarity scores [42-44]. Other studies including Knickerbocker, et al. [45] and Shahriar and Zulkernine [46] used an offensive defence technique in which a fake login credential is fed into a suspected phishing webpage and the response from the webpage is compared against the expected response from the corresponding legitimate webpage. Dissimilar response indicates that the webpage is a phishing one. Both visual similarity and offensive based techniques provide limited protection, that is they only protect against phishing webpages that imitate the legitimate webpages recorded in the database. A rule based technique has emerged to be a popular technique among recent studies including Zuhair, et al. [47], Shirazi H. [48], Jain and Gupta [49], Sahingoz, et al. [50] and Li, et al. [51]. The technique uses rules manually set or automatically determined by Machine Learning (ML) algorithms to distinguish phishing webpages from legitimate ones using features extracted mainly from URLs, webpage structure and contents, and third-party information. Though a few techniques have achieved good detection performance, most have achieved low to moderate detection rates and/or have used a small number of different feature categories, which increases their susceptibility to detection evasion. In addition, the performance of most techniques was measured using a small number of metrics, limiting our understanding of their all-round effectiveness.

The solutions in the second group have focused on detecting malicious webpages (through detecting their hostnames) that are hosted in flux networks. Flux networks are botnets in which the hosts (bots) of the websites or the name servers of the websites are changed rapidly to evade IP address blacklisting and thus prolong their effective life. They are referred to as Fast Flux Service Networks (FFSNs), Name Server IP Flux Networks (NSIFNs) or double flux networks depending on whether it is the host of the website, name server or both that are changing. Using DNS, host and network-based predictive features, some studies have proposed ML-based techniques to detect the hostnames that are hosted specifically in FFSNs while others have focused on the hostnames hosted in NSIFNs. These proposed solutions have several limitations including achieving low to moderate detection performance, impracticability due to their inconsideration of some of the hostname types in the prediction process, lack of instant

detection capability and the use of feature sets with low diversity, which increases their susceptibility to detection evasions.

1.2. Problem Statement and Motivation

Existing anti-spam email filtering, phishing awareness training and phishing website detection solutions all play an important role in reducing the number of users falling for phishing website attacks. Despite their use, the number of zero-day phishing websites and their successful attacks have been steadily growing globally over the years. For instance, APWG [52] observed that the number of new unique phishing websites per month jumped to 92,564 in 2016, an increase of 65% from 2015 and 5,753% from 2004 whereas Federal Bureau of Investigations (FBI) reported that they received 11 times as many complaints from phishing victims in 2020 as in 2016 [53]. Similarly, Federal Trade Commission (FTC) reported an increase of 113% of identity theft complaints in 2020 from 2019 [54]. Thus, there remains a need for new and improved techniques. This research focuses on solutions addressing the second stage of the attack (indicated on the right-hand side of Figure 1.1). Two particular challenges facing them are:

- They are executed in line as part of interactive user sessions (e.g. web browsing) and must not introduce delays that would interfere with user experience.
- The most widespread and fastest detection technique is blacklist but the technique fails to detect a large number of recently-created (zero-day) phishing websites.

Given the prevalence and significance of zero-day phishing websites, an ideal solution for effective and efficient detection of the websites should have the following design characteristics:

- It must not rely on lists of known or suspected phishing websites compiled from human or software-generated reports.
- ML has an advantage over heuristics in its flexibility to update the prediction rules through data re-collection and re-training, which is useful in maintaining the optimal detection performance when phishing website techniques change over time. Therefore, it is a preferable approach. The heuristic technique, on the hand, is difficult to maintain and is prone to high error rates when the changes occur.

- It must achieve high prediction accuracy and low misclassification rates. An accuracy range of 99% to 100% and misclassification rates between 0% and 1% are desired in this domain.
- It must perform detection in real-time i.e the additional time taken to determine whether a URL will take a user to a phishing webpage or not must not degrade the user's overall web browsing experience.
- It should focus specifically on the detection of phishing websites in order to optimize a detection performance. As illustrated by Caglayan, et al. [55], different types of malicious web services such as malware, spam and phishing have variations in some of their characteristics that are often used to derive prediction features. This suggests that generic solutions that detect all types of malicious website are likely to be less effective in detecting phishing websites than solutions that are specific to phishing.
- It should use novel prediction features. This is because the attackers may already have learnt prediction features used by existing detection solutions and have developed mechanisms to circumvent the solutions. The use of novel features will enable the solution to be ahead of attackers.
- It should use highly diversified prediction features (features selected from a wide variety of categories) to make the solutions more resistant to detection evasion. The attackers would need to develop at least one detection evasion technique for each feature category to have any chance of eluding the solutions, which is likely to be a difficult and time-consuming task for most attackers to invest in.

1.3. Aim and Objectives

The aim of this research is to investigate, propose and evaluate novel set of features that can be used to instantly and accurately predict zero-day phishing webpages using an ML technique. In order to achieve this aim, the following objectives were identified;

1. To conduct a review of structural designs of webpages collecting personal data, and of distinctive characteristics of their flux and non-flux networks.
2. To investigate and identify ML techniques that have successfully been used to address cyber security problems especially in the detection of malicious websites.
3. To identify specific approaches for predicting zero-day phishing webpages to be explored in this study.

4. To identify the appropriate ML techniques for each identified prediction approach.
5. To investigate and identify potential sets of prediction features for each approach based on the webpage structural designs and/or network characteristics of website hosting networks.
6. To identify the most relevant sets of features and use them to build an ML based prediction model for each approach.
7. To evaluate the prediction performance and efficiency of the resulting prediction models.

1.4. Contributions

The research documented in this thesis has produced the following original research contributions, which will be described in detail in Chapters 4 and 5:

1. A method for identifying webpages that capture users' personal data is proposed. The method uses a combination of webpage structural components designed specifically for data collection, and words or phrases in the webpage structure and contents that indicate the type of personal data being collected by the webpages. The method is useful in filtering out, thus, averting the prediction of webpages that do not capture users' personal data as they do not carry any phishing threat. This will avoid slowing down the web browsing experience when users are accessing these webpages and the generation of potential false positive errors on the webpages.
2. A novel set of 26 predictive features derived from five different feature categories related to URL, webpage structure and contents and information from third party services was identified to be effective in predicting zero-day phishing webpages using an ML technique. The feature set was evaluated using several ML algorithms and produced good prediction performances with most of them, indicating that it is robust for the prediction task. The feature set was found still to be effective when the evaluation was repeated with respect to an independent data set collected at a later time, without retraining. The measured prediction time was found to be sufficiently low for potential use for real-time protection of users.
3. A novel set of 56 features derived from six different feature categories based on DNS, network and host characteristics of FFSNs was found to be useful for predicting zero-day phishing hostnames hosted in FFSNs using an ML technique. The feature set was

applied using several binary and multi-class classification ML techniques and achieved the highest prediction performance with a binary classification technique in which phishing hostnames hosted in FFSNs are distinguished from those hosted in phishing non-flux networks, CDNs and legitimate non-flux networks, combined as a single hostname class. The feature set has shown that it is relevant and reliable for the prediction task by producing high prediction performances in most ML algorithms used for evaluation. The feature set also produced a good prediction performance without retraining when evaluated against phishing webpages collected at a later time.

4. A novel set of 11 features derived from five different feature categories based on DNS, network and host characteristics of NSIFNs was found to be useful for predicting zero-day phishing hostnames hosted in NSIFNs using an ML approach. The feature set was applied using several binary and multi-class classification ML techniques and achieved the highest prediction performance with a binary classification technique in which phishing hostnames hosted in NSIFNs are distinguished from those hosted in both phishing and legitimate non-flux networks, combined as a single hostname class. The feature set consistently produced good prediction performances with most ML algorithms used for evaluation, indicating that it is effective for the prediction task. The feature set is also found to be effective without retraining in predicting phishing webpages collected at a later time. The low prediction time measured suggests that the approach can potentially be used to protect users in real-time.
5. An approach to evaluation of feature sets was devised whereby the performances of the features were measured using a large number of different ML algorithms. As each algorithm relies on different data distribution assumptions and uses a different statistical approach in creating the prediction rules, evaluating the features using a large set of algorithms and comparing the resulting performances allows us to draw conclusions on the general effectiveness of a feature set for a specific prediction task. This was applied in each prediction approach studied.
6. A multi-class ML classification technique is introduced for the first time in this domain. It is used to evaluate the feature sets in two prediction tasks (predictions of phishing hostnames hosted in FFSNs and NSIFNs). In this technique, which is a more difficult task compared to a binary classification technique from the ML perspective, all hostname types in each prediction task were predicted as independent classes, allowing for the detection of exact type of a hostname. Though the technique performed less well

than the binary classification technique in both prediction tasks, it provides more useful information to security experts for effective decision making and still has produced comparable prediction performance to related works carrying out binary classification.

7. An approach to behavioural classification whereby a dynamic feature (i.e. changes in the IP addresses returned by DNS queries about a particular name server) is monitored over time in order to label a training dataset, then ‘instantaneous’ features are used for classification. In effect, the classifier is trained to use a combination of instantaneous features as a proxy for the dynamic feature. This results in a much shorter classification time than would be the case if features derived directly from monitoring behaviour over time were used for classification.

1.5. Research Scope

The following describes the scope of this research:

- The first prediction task is designed to predict phishing and legitimate webpages which prompt and collect personal data using HTML forms, JavaScript based dialogue windows or combination of both. Those which collect the data using other mechanisms were not addressed by this study.
- The second and third prediction tasks are designed to predict hostnames hosted in FFSNs and NSIFNs in which IP addresses of the hosts and authoritative name servers of the hostnames frequently change respectively. The flux networks in which hostnames of the websites and those of the authoritative name servers frequently change were not addressed by this study.

1.6. Thesis Structure

This thesis is organized in six chapters. *Chapter 2* describes background information on phishing website attacks. First, we go deeper into the meaning of phishing and phishing websites, the methods attackers often use to distribute the websites to their targets, typical stages undertaken by attackers to launch successful phishing website attacks, and the recent attack trends. The anatomy of webpages that collect personal data is also described. In the second part of the chapter, we describe the network infrastructures hosting phishing and legitimate hostnames. We explain the structure and operation of the Domain Name System (DNS), non-flux networks, Content Delivery Networks (CDNs), FFSNs and NSIFNs.

Chapter 3 describes ML techniques which are commonly used in the detection of malicious websites. We begin by defining ML and describing types of ML, and then delve deeper into ML for classification tasks, different types of ML classification task, and the concepts of flat and hierarchical classifications. We then describe popular traditional ML and Deep Learning (DL) algorithms for classification tasks specifically in the detection of malicious websites, and the standard workflow for undertaking ML classification tasks.

Chapter 4 presents our ML-based approach for predicting zero-day phishing webpages using a novel set of features based on the webpage's URL, its structure and content, and information from third party services. First, we review related works and discuss their limitations. Next, we describe the potential predictive features for the approach, the proposed system architecture and the experiments conducted to develop the prediction model. Results of the experiments are also presented, discussed and compared against the related works.

Chapter 5 presents our ML-based approaches for predicting phishing hostnames hosted in flux networks using featured based on DNS, network and host characteristics. We divide the chapter into two main sections. The first section presents the approach for predicting phishing hostnames hosted in FFSNs while the second section describes the approach for predicting phishing hostnames hosted in NSIFNs. In each section, we first review related works and discuss their limitations. We then describe the potential predictive features for the prediction task, the proposed system architecture and the experiments conducted to develop the prediction model. Results of the experiments are also presented, discussed and compared against other similar works.

Chapter 6 summarizes the research background and our contributions, discusses the limitations of our proposed solutions and identifies areas of future work.

Chapter 2

Phishing Background

2.1. Introduction

Chapter 1 introduced this research by highlighting the significance of phishing website attacks to the internet community, existing solutions that are being deployed or have been proposed to address the attacks and their limitations, and the aim and objectives of the research to address the attacks. This chapter addresses the first objective of the research by reviewing of state-of-the-art attack techniques, structural characteristics of phishing websites, and structural and operational characteristics of their hosting networks. It provides a foundation for identifying potential features that can be used to predict zero-day phishing webpages as described Chapters 4 and 5. The chapter is organised in four sections. Section 2.2 describes the historical background of phishing website attacks, the common techniques for distributing the websites, stages of executing the attacks and the current trends of the attacks. Section 2.3 describes the structural characteristics of legitimate and phishing webpages collecting personal data, the structure and operation of the Domain Name System (DNS), and the networks hosting legitimate and phishing websites. Section 2.4 summarizes the chapter.

2.2. Overview of Phishing

The term phishing was first recorded in 1996 in the alt.2600's hacker newsgroup [56]. It was derived from the word "fishing", replacing "f" with "ph" [57]. Fishing, in this context, meant capturing credentials of online user accounts from the sea of internet users. The user account whose credentials were stolen was referred as the *phish* whereas an attacker who succeeded in stealing the credentials was known as the *phisher*. In early phishing attacks, social engineering skills were used to design legitimate-looking emails, with enticing messages as bait, to prompt users to respond to the email by submitting the credentials. Over the years, other more technological approaches including the use of phishing websites and malware have been increasingly used to phish the data, enriching phishing attacking options to yield high phishing success. Attackers have also been increasing the types of personal data to steal apart from the login credentials in order to perform a wide range of malicious activities. Other popular data include family records and employment history.

Today, phishing websites are the preferred method employed by attackers to collect users' data [23]. Other common techniques include the use of phone calls to directly ask user for the data (also known as vishing), the use of phishing emails to prompt users to directly pass the data as email responses and the injection of malware into the host application to steal the stored or transmitted data. A phishing website is often a replica of a legitimate website that prompts users for similar data requested by the legitimate website [40, 58]. Phishing websites that have operated for more than a short period of time are often detected by internet users, security experts or anti-phishing technologies. They may be reported to various government agencies (such as the National Cyber Security Centre (NCSC) in the UK), search engines, security service providers and community clearing houses (e.g. PhishTank). After being confirmed by the operators, the URLs are recorded in databases of blacklisted phishing websites. The majority of newly created phishing websites (also referred to as zero-day phishing websites) usually operate undetected for a while.

2.2.1. Techniques for Distributing Phishing Websites

Phishing websites are distributed to their targeted users through various techniques. Here, we describe some of the techniques that are often deployed by phishers.

2.2.1.1. Phishing Emails

From early days of phishing to today, this has been the most popular technique [59, 60]. Phishing emails use baits to entice users for the exchange with personal data [61]. Typical baits include offers of free goods, services and vouchers, and activation of accounts to continue accessing specific services. The emails are crafted in such a way that they look to be from legitimate businesses or from people the recipients of the emails have associations with. The emails include hyperlinks that recipients are lured to click, taking them to the phishing webpages, which prompt for the data (APWG, 2015). Once user data has been submitted, it is sent to email accounts or servers owned by the phishers [60]. Traditionally, a phishing email campaign broadcasts the same generic email to all users targeted by the phisher [26]. However, phishers have been increasingly employing new sophisticated approaches whereby more personalized emails are developed to target a specific small group of users. The approaches

include spear phishing [59, 62] and Business Email Compromise (BEC) [40, 63]. Figures 2.1 gives an example of a generic phishing email.

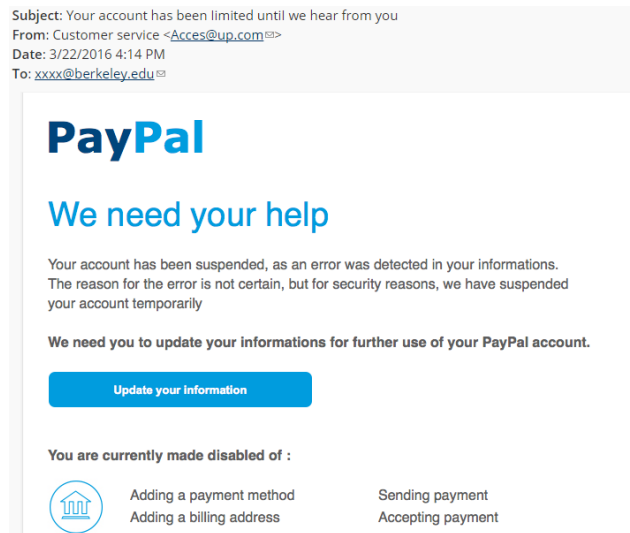


Figure 2.1. A generic phishing email pretending to be from PayPal [64].

2.2.1.2. Social Media Phishing

Phishers also distribute the spams using fake social media accounts or through compromised genuine accounts belonging to others [65, 66]. In the former case, phishers create new accounts using names that impersonate other organizations/brands and spread the spam to contacts they acquired previously through other malicious means such as malware scanning. In the latter case, phishers compromise accounts of legitimate users or organizations, through techniques such phishing emails and brute force attacks, and send spams to the contacts of the accounts. The phishing spams are often drafted in similar ways to phishing emails with links that take users to phishing websites. The messages are sent via messaging applications of the networks such as Facebook messenger or as account posts.

2.2.1.3. Mobile Phishing

The recent growth of internet access via mobile devices has caught the attention of phishers to distribute spams to mobile devices through SMS (also known as smishing) and messaging apps such as WhatsApp [67, 68]. Similar to phishing emails, the spams contain links to phishing websites, among other attacks. Lockout [69] estimated that the rate at which mobile users

receive phishing SMSs and tap into phishing URLs has been growing at an annual rate of 85%. They also observed that by 2016, 56% of mobile users using their product received phishing SMSs and clicked their embedded URLs.

2.2.1.4. Malware Injections

Software vulnerabilities are often exploited by phishers to launch phishing website attacks. For instance, vulnerabilities in web browsers and plug-ins have been exploited by phishers to install malware that redirect users to phishing websites when targeted legitimate websites are visited [70]. Other forms of installed malware can passively intercept and steal data sent to the legitimate websites (man-in-the-middle-attack) or directly scan for data stored in the devices [71, 72]. The malware are commonly distributed through phishing email attachments, webpage content injections or the copying of corrupted files [71, 72].

2.2.1.5. Pharming

In Pharming, also known as DNS poisoning/hijacking, an attacker gains an access to the targeted organization's domain registrar account or the DNS provider's administration account, using compromised credentials or other mechanisms, and then replaces IP address of a legitimate website with an IP address of a phishing website [73-75]. Upon user's request to access the legitimate website, the domain name service returns the new IP address, thus redirecting the user to the phishing website.

2.2.1.6. Search Engine Poisoning

Search Engine Poisoning (SEP) refers to manipulation of search engine ranking metrics on particular websites to ensure the websites are highly ranked when are searched using certain keywords [76-78]. Attackers exploit this technique to directly or indirectly increase the ranking of their malicious websites in the search results. There are various techniques to achieve SEP but the most common approach is through compromising legitimate websites with a high reputation in search engines [76, 78, 79]. The websites are injected with multiple hidden keywords desired by attackers and the redirecting scripts pointing to phishing websites. When the keywords are searched, the compromised websites are returned in highly ranking positions in the results. By opening the websites, the redirectors direct users to the phishing websites. In

another approach attackers insert links of their phishing websites in many compromised websites. This enhances reputation of their websites as they are linked to many highly reputable websites and therefore become highly ranked in search engine results [77, 80].

2.2.2. Stages of Executing Phishing Website Attacks

A successful phishing website attack requires a phisher to execute a number of actions. Most attacks have the following five stage actions (also summarized in Figure 2.2) [81-83]:

1. *Identify targets.* A phisher identifies a target organization, its website and its online community of users. For spear and BEC phishing, specific employees or members of a community to send their phishing spams to are identified. For BEC attacks, the attacker also identifies the staff whose email address will be spoofed and used to send spams to the targets.
2. *Survey and gather information.* The phisher surveys the website and other resources of the target organization and collects specific information such as products, business processes and contacts to use for creating a phishing website and the contents of phishing spams. Through network probing tools or a help from an insider, the attacker may also survey network related resources of the organization or specific users such as email servers, database systems, operating systems, DNS server and web servers to identify their specific security vulnerabilities. As described in section 2.2.2.4, the vulnerabilities can be exploited to facilitate phishing website attacks.
3. *Prepare the attack.* In this stage, the attacker sets up a hosting infrastructure by acquiring a domain (through registering a new domain or compromising a legitimate domain), configures a hosting webserver and sets up a database or an email account to receive and store the captured data. The phishing website is created, linked to the domain name and the database, and hosted in the webserver. The phishing spam is developed by crafting the message with a bait along and a link of a URL of the phishing website or an attachment of a malware infected file. At this stage, phisher also creates a fake account of an email, social media or other platforms which will be used to send phishing spams.
4. *Execute the attack.* From the fake account, the phisher sends the phishing spam to all the collected or specific contacts. Users receive and open the spams. Upon clicking the URL link, users are directed to the phishing website, where they submit their data. The data is then sent to the attacker's server or email account. For spams with attached

infected files, when the spam attachment is opened, the malware exploits the vulnerabilities of the user's machine to perform phishing activities such as man-in-the-middle attacks and pharming.

5. *Collect data and conceal the traces.* The attacker monitors the captured data. To conceal traces of the compromise and avoid detection, the website is usually run only for few hours, days or weeks before it is taken down. Also, the attacker usually deletes event logs in the network resources (such as email and DNS servers when executing BEC and pharming-based attacks respectively) that were used to facilitate the attack.

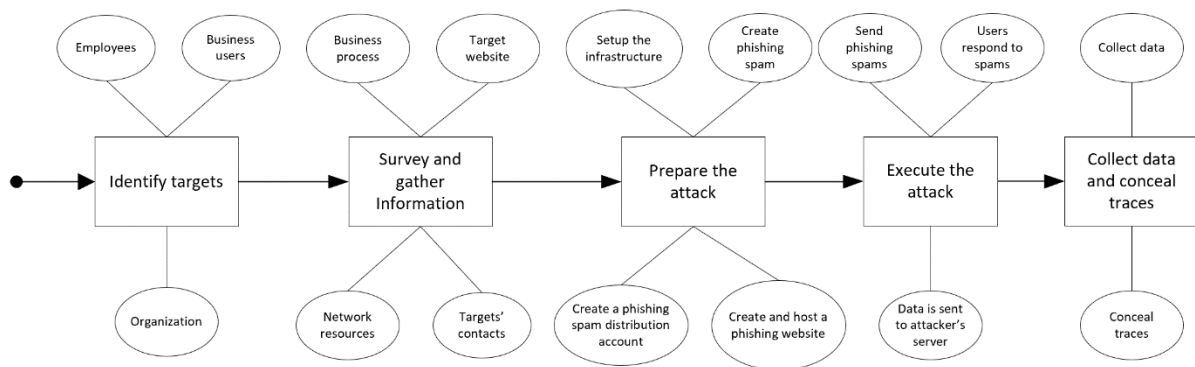


Figure 2.2. Typical stages of a phishing website attack.

2.2.3. Current Trends in Phishing Website Attacks

2.2.3.1. Phishing Emails, Social Media Phishing and Mobile Phishing

One out of every 99 emails is a phishing email [84]. Nearly 4.7 billion phishing emails are sent every day [85]. About 30% of the sent phishing emails pass through the existing anti-spam filters and reach to users' inboxes [84]. 30% of the received emails are opened by users [59], of which almost 5% are responded to by users clicking the links or attachments [86]. 30% of these emails are responded to by users within the first 10 minutes while 52% are responded to in the first hour [27]. PhishLabs [23] reported that of the 65% of phishing emails that reach users' inboxes, 88% are involved with credential thefts through redirecting users to phishing websites.

Use of social networks as a channel for phishing spams has been growing rapidly in the last few years [27, 87]. For instance, social media phishing attacks rose by 500% between 2015

and 2016 [65]. As a results, 5% of all global phishing attacks are initiated by social media spams [87]. Use of fake social media accounts impersonating organizations is significantly growing as a major phishing vector. ProofPoint [88] found that 19% of social media accounts representing top global brands were fake, with 4% of them being involved with phishing, malware, protest and satire. The phishing accounts often posed as customer support services in order to lure customers to submit their data in exchange of services or promotion gifts. In order to increase efficiency and productivity in mass distribution of spams in social media, phishers have been observed to employ social botnets to distribute spam to large number of users in a short period [89].

Mobile phishing is continuing to rise as more users and employees are increasingly using mobile devices to access the internet and their organizations' network resources respectively [69]. The latter was also highlighted by ProofPoint [90]'s study which showed that 84% of organizations in 2019 experienced smishing attacks. La Porta [66] estimated that mobile phishing causes 48% of all phishing attacks. Also, there has been a sharp rise in use of mobile messenger apps such as WhatsApp to distribute phishing spams. La Porta [66] observed an increase of 170% of such attacks between 2017 and 2018. The majority of the recent mobile phishing attacks have been observed to target major banks across the globe [23, 91].

2.2.3.2. Phishing Websites

The number of phishing websites has been growing steadily over the years. APWG [92], for instance, reported that the average number of unique phishing websites in 2016 was 92,564 per month, increased from 1,609 per month in 2004. This represents an increase of 5,753% in 12 years. The same study observed 1.2 million unique phishing websites in 2016, an increase of 65% from 2015. Similarly, the number of mobile phishing websites has been skyrocketing. La Porta [66] observed over 4,000 new mobile phishing websites each day.

Phishers are increasingly using Transport Layer Security (TLS) (formerly SSL) certificates to certify identities of their websites and encrypt the traffic. The certificates also increase the level of trust in the websites to users. Phishers use the certificates to lure users who perceive that any website using the certificate is a legitimate one. La Porta [66] reported that there was an increase in use of TLS certificates of 1000% during 2017. APWG [93]'s research observed that

74% of phishing websites, at the end of 2019, were using the certificate, a rise from less than 1% in early 2015. Figure 2.3 illustrates the monitored growth between 2015 and 2019.

There are three main types of TLS certificates offered namely Extended Validation (EV), Organization Validation (OV) and Domain Validation (DV) [94]. In EV offering, the right of an applicant to use a specific domain as well as extensive validation of the entity/organization owning the website are checked by the Certificate Authority (CA) [95, 96]. Legal, location/physical and operational existence of the organization are thoroughly validated [96]. For OV, the CA checks for a right of the applicant to use a specific domain and does a certain level of vetting on the legitimacy of the organization applying for the certificate [96, 97]. This includes checking the name of the organization against a national registry of the registered organization’s country as well as contacting a representative of the applicant [97]. In the DV offering, the identity and legitimacy of the organization owning the domain and website are not validated by the CA [97]. DV and OV certificates are preferred by phishers because they have fewer validation requirements compared to EV and therefore they can more easily be obtained. Only 1% of the certificates used in phishing websites are EVs, 20% are OV, and the rest are DVs [98]. However, the majority of phishing websites still do not use any certificate.

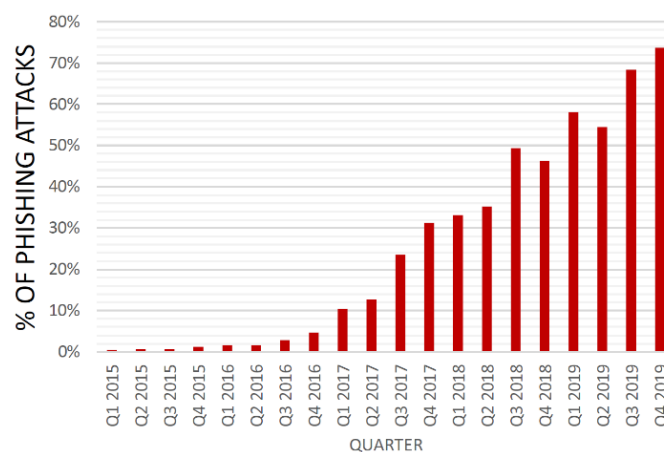


Figure 2.3. Percentage of phishing attacks that were using TLS certificates between 2015 and 2019 [93].

2.2.3.3. Phishing Domains and URLs

Phishers re-use a small number of domains and web servers to host many phishing websites as a means to optimize resources, thus increasing their return on investments. For instance,

PhishLabs [25] observed that the 1 million phishing websites they monitored throughout the year were registered to only 170,000 unique domains and hosted in 66,000 unique IP addresses. Similarly, Webroot [41] observed that 90% of all phishing websites in 2017 were hosted in 62 unique domains whereas one unique IP address was found to host 400,000 phishing websites. One domain can be used to host multiple phishing websites targeting different organizations [25]. Up to 51% of all the phishing websites are hosted in compromised legitimate domains while the rest use domains registered by phishers [39]. Use of compromised domains is becoming increasingly popular among phishers because they are difficult to detect since most of the anti-phishing filters track indicators of maliciously registered domains such as small domain age. The compromised domains are accessed by phishers through hacking servers hosting legitimate websites [99, 100].

Use of free website hosting services to host phishing websites has been rapidly growing in recent years. For instance, percentage of phishing websites hosted in such services grew up from 3% in 2015 to 13.8% in 2018 [23]. The services are preferred by phishers because they are cheap and provide free legitimate domain names, which make phishing URLs less suspicious to users and anti-phishing filters. *000webhostapp* is the most abused service, hosting up to 69% of all the phishing websites hosted in such services [23]. Figure 2.4 shows the growth rate of phishing websites hosted in the free web hosting services between 2015 and 2018.

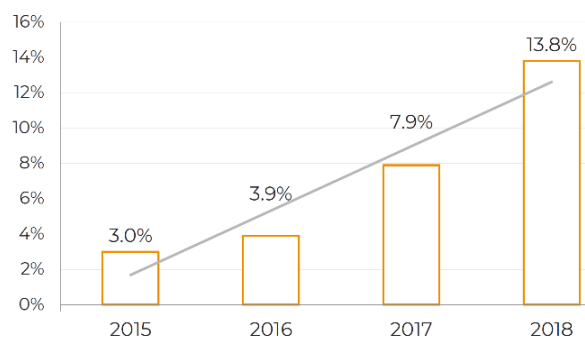


Figure 2.4. The growth of rate of phishing websites hosted in free web hosting services [23].

In terms of the age of domains registered by phishers, some of the phishers have started deploying the domains a few days, weeks or months after registration. This is different from the traditional approach of using them immediately. This aims at misleading some of the anti-phishing filters which use short domain age as a main detection feature. Aaron and Rasmussen

[39] observed that only 10% of the domains were used for phishing activities on the day of registration while the majority of them were used between three days and one week after their registration. A significant number of domains remained dormant from few months to one year before being used. Figure 2.5 illustrates a range of periods the domains were kept dormant before being used.

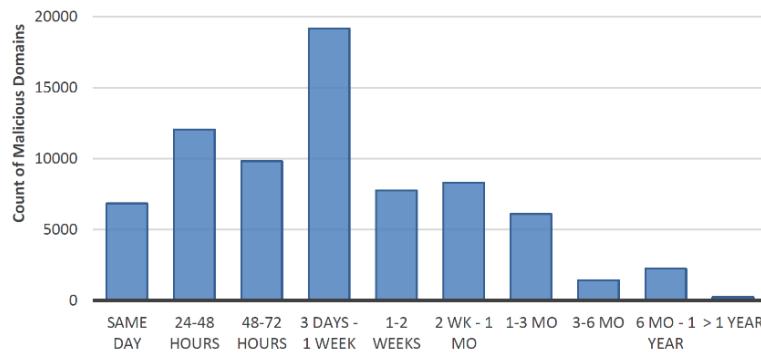


Figure 2.5. Distribution of domain age of phishers-owned domains prior to phishing attacks [39].

Life spans of both active phishing domains and websites have been decreasing over the recent years. Akamai [26] observed that 89% of over 2 billion malicious domains had a life span of activity of less than 24 hours while 94% of them were active for less than three days. The average active life span of phishing websites observed by Webroot [40] was less than 15 hours in 2016. About 84% of the phishing websites observed in 2016 stayed online for less than 24 hours. In 2018, Webroot [41] observed that most of the phishing websites stayed online for between 4 and 8 hours, with the longest and the shortest times were 44 hours and 15 minutes respectively. The main reason for this practice is to avoid the websites from being noticed by security experts, blacklisted and their footprints traced.

The use of shortening URL services including *bit.ly* and *goo.gl* to hide true phishing URLs has been on the rise. Using the services, shortened URLs are generated from the original phishing URLs and are used as links in the phishing spams. Since the services are legitimate with genuine domains, users and most of the anti-phishing filters have difficulty to associate them with phishing websites. Webroot [100], for instance, estimated that 1 in every 130 *bit.ly* URLs and 1 in every 190 *goo.gl* URLs are linked to malicious websites.

2.2.3.4. Most Targeted Brands and Industries

Traditionally, phishers have been targeting companies in the finance, e-commerce and e-payment industries to gain user credentials that would allow them to steal money from the account holders [60]. The attacks focus on obtaining basic credentials such as usernames and passwords. Recently, there has been a rapid shift to targeting other industries. The major ones include web email (e.g Gmail and Yahoo!), cloud storage (e.g Dropbox and Google Drive) and Software-as-a-Service (SaaS) (e.g Office365 and Adobe) services. For instance, the proportion of phishing attacks on web email accounts rose from 12% in 2013 to 24% in 2018 while those targeting SaaS jumped from less than 0.1% to 7.1% in the same period [23, 60]. Meanwhile, the total volume of phishing attacks in each industry has been rising in each year [23].

The shift has been due to the fact that there has been a growing trend of web applications using third party-based authentication methods, such as those of Facebook and Gmail, as their login mechanism. By obtaining the user's credential, phishers can infiltrate many accounts in various applications related to the same victim, thus maximizing their gains. Another significant phishing trend is that many phishing websites are collecting more information than just the basic credentials, with an aim of exploiting new avenues of phishing attacks such as fraudulent tax claims or using it to bypass multi-factor authentication. Extended information harvested by phishers include employment and family records [60].

2.2.3.5. Phishing Toolkits and Phishing-as-a-Service

One of the main reasons for the surging numbers of phishing attacks in recent years is the availability of automated tools and services such as phishing toolkits and Phishing-as-a-Service (PaaS) [59]. Phishing toolkits are collections of files containing scripts, images and other resources that are used to create phishing spams and websites [25]. The toolkits are widely distributed via the darknet [25, 26]. The tools have contributed to the increased number of non-technical phishers since they automate most of the phishing tasks [25, 26]. The tasks include creation and distribution of phishing emails, creation of desktop and mobile based phishing websites, and implementation of various detection evasion techniques [26, 59, 60, 101]. Examples of such techniques include blocking a website access to IP addresses and/or domain names of known ISPs, security vendors and researchers, detecting and blocking webpage automated scanning and crawling operations, and random generation of unique URLs and/or

HTML codes for the same phishing webpage every time it is loaded (to evade signature-based detection). The toolkits are often evolved by their authors to produce many variants to improve their capabilities as well as evading signature-based detection. Each variant is capable of creating many variations of phishing websites targeting the same organization. For instance, Akamai [26] observed that 14 variants of one toolkit were responsible for multiple phishing websites hosted in 1,669 different domains, all targeting PayPal. The toolkits often stay active for less than 300 days, with 60% of them stay active for 20 days or less to limit their exposure to detection [26].

PaaS is the latest innovation for developing phishing attacks with high productivity. This is a cloud-based application that is provided as a service and allows phishers to create and distribute phishing attacks including phishing emails and websites for a fee. The PaaS platforms are mostly hosted in botnets for high efficiency and enhanced security against detection [59].

2.3. Structural Characteristics of Data Capturing Webpages and Their Hosting Networks

This section describes the structure of webpages that collect personal data from users and the networks hosting them. Section 2.3.1 describes the former to provide a background of our proposed work in Chapter 4 while sections 2.3.2 and 2.3.3 describe the latter as a background for our proposed work described in Chapter 5.

2.3.1. Anatomy of a Webpage Collecting Personal Data

A webpage is a hypertext document, usually in a Hyper Text Markup Language (HTML) format, that is viewed remotely through a web browser. We term a webpage that prompts for and collects personal data as *Personal Data Capturing (PDC) webpage*. Each webpage has two main parts: Uniform Resource Locator (URL) and the HTML file describing its structure and contents.

2.3.1.1. URL

URL, also referred to as a webpage address, is a unique string identifying a location of a webpage file in the internet. A URL has three main components namely the protocol, hostname

and path. For instance, for a URL *https://cs.berkeley.edu/resources/faculty-staff*, *https* is a protocol, *cs.berkeley.edu* is a name of the host (hostname) and *resources/faculty-staff* is a path. Hostnames have hierarchical structure: *edu* is a top-level domain used by educational organisations, *berkeley* is a sub-domain of *edu* owned by University of Berkeley, and *cs* is sub-domain of *berkeley.edu* issued to the Berkeley's Computer Science Department. The hostname denotes a website (host) and the path gives the location of the webpage in the file system of the website. See section 2.3.2 for further details.

To download a webpage from its server to a client machine (e.g. in order to display it in a browser), a communication protocol known as Hypertext Transfer Protocol (HTTP) or its encrypted version HTTP Secure (HTTPS) are normally used. These two are indicated as *http* and *https* in the URL respectively. In order to use HTTPS, the owner of the website has to acquire a TLS certificate for the domain. Not only is the certificate useful for encrypting the traffic, it is also used to authenticate hosts of the websites.

2.3.1.2. Webpage Structure and Contents

The components of an HTML webpage structure are built around a basic element known as tag. Examples of tags are `<title>.... </title>`, `<meta....>`, `<link....>` and `<script>.... </scripts>`. The HTML webpage structure is made up of two sections; head and body. The head section may contain components including title, meta data, links and scripts. The Title describes the heading of a webpage. The meta component contain data which provides various specifications about the webpage such as character set, page description, keywords and author. Links provide addresses of other webpages or objects, such as images and stylesheets, connected to the webpage. There are three types of links namely links to other webpages or objects, links to various sections of the same webpage and void links (those with no URL assigned) [58, 102-104]. Some of the webpages are created in multiple versions for various reasons such as translation of a webpage in multiple languages. In order to identify the related versions, canonical and alternate links are included in the head section with URLs of all the related webpages [105-109]. Below are examples of alternate links of an Instagram's PDC webpage which indicate URLs (assigned using *href* attribute) of the same webpage translated in English, French and Italian languages

```
<link rel="alternate" href="https://www.instagram.com/accounts/login/?  
hl=en" hreflang="en" />
```

```
<link rel="alternate" href="https://www.instagram.com/accounts/login/?  
hl=fr" hreflang="fr" />  
<link rel="alternate" href="https://www.instagram.com/accounts/login/?  
hl=it" hreflang="it" />
```

The main component of the body section of the PDC webpage is the mechanism for capturing and submitting user's data. The webpage can use an HTML form or a script-based dialogue window for the task. The HTML form is a structural component of a webpage for collecting input data from the user and send it to a specified URL for processing. The form is delimited by a tag `<form> ... </form>` and often contains various input fields, each defined with an input tag `<input>` and an attribute *type* for specifying the type of data to be collected. Common input types are *text*, which is a field accepting any text information, and *password*, which is a field specifically for password entries [110]. Others include *email* for email addresses, *tel* for telephone numbers, and *date* for day, month and year entries [110]. The form uses a URL assigned to its tag *action* to identify the URL to send the collected data to for processing tasks, such as saving the data into the database. The data processing webpage is also known as form handler. `<form action="/login" method="post">` is an example of a form tag indicating a form handler named *login* within the same host as the PDC webpage.

Script based dialogue (pop-up) windows which also can be used to prompt for personal data are often designed using JavaScript or JQuery scripting languages. With JavaScript, the `window.prompt()` command displays a prompting message and captures the inputs [111, 112]. Alternatively, both JavaScript and JQuery can incorporate the HTML form to prompt for inputs using input fields [113, 114].

Not every webpage with the HTML form or a dialogue window collects personal data. For instance, Google's search webpage collects user's search keywords. What differentiates PDC webpages from others is that they usually contain words or phrases that are related to the specific data being collected. These phrases, we refer to them as *PDC phrases*, can be within the form text [110], in the label tags `<label></label>` of input fields [115], as default values to *value* attribute of input fields [116] or elsewhere in the HTML document. Some of these phrases, for instance, *login*, *log in* and *sign in* can also be used as names of a submit button of the form. In this case, they are assigned as values to the input type *submit*.

2.3.1.3. Structural Characteristics of Phishing Websites

Phishing websites tend to imitate the URL, layout and contents of legitimate websites as much as possible in order to lure users as well as evading detection. The degree of similarity between phishing and legitimate websites varies depending on the phisher’s skills in producing the replica. Phishing websites can be categorized into three types depending on their levels of look and feel relative to the legitimate websites they imitate [117]. These are simple, advanced and extreme phishing websites. Three main criteria to describe the categories are summarized in Table 2.1. The metrics are level of visual similarities, the number of replicated webpages in a phishing website along with the number of original links maintained and whether the website supports dynamic user interactions like the original one. For instance, a simple phishing website is not visually close to the original website, usually has one imitated webpage with very few unmodified links and does not support any form of user interactions with the server. At the other end of the scale, an extreme phishing website is exactly similar to the original website visually, has the same number of webpages and links as the original website and supports the same user interactions with the server as the original website.

Figure 2.6 provides an example of simple phishing website in which Figure 2.6b shows a phishing website that looks somewhat similar to PayPal’s legitimate log in webpage, shown in Figure 2.6a. Using the phishing toolkits, which often consist of loaded templates of the targeted legitimate websites, phishers can easily produce phishing websites that looks closely similar to the legitimate ones by performing a few modifications on the structure and contents of the templates. For instance, they have to change the form handler to point to the URL of a machine that they will use to collect the phished data. Given the popularity of the toolkits among the phishers, the advanced and extreme phishing websites are likely to be the majority of the phishing websites created today.

Type of Phishing Websites	Similarity Metrics		
	Visual Appearance	Page Depth	Supports User Dynamic Interaction
<i>Simple phishing</i>	Somewhat similar	One webpage with few similar links	No

<i>Advanced phishing</i>	Mostly similar	Limited number of pages with few similar links	No
<i>Extreme phishing</i>	Similar in every way	Unlimited number of pages with completely similar links	Yes

Table 2.1. Types of phishing websites categorized based on the three similarity metrics [117].

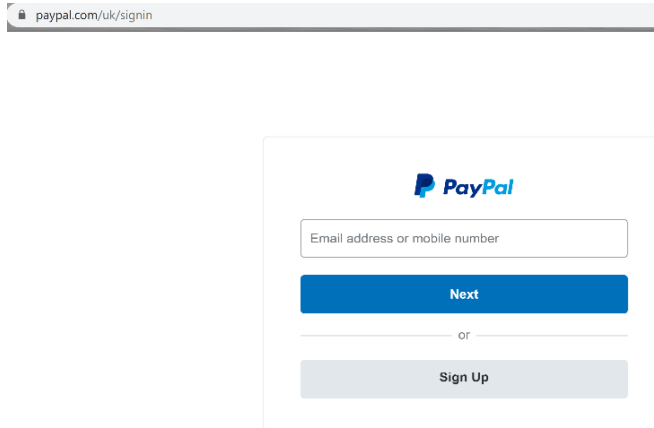


Figure 2.6a. A legitimate PayPal webpage.

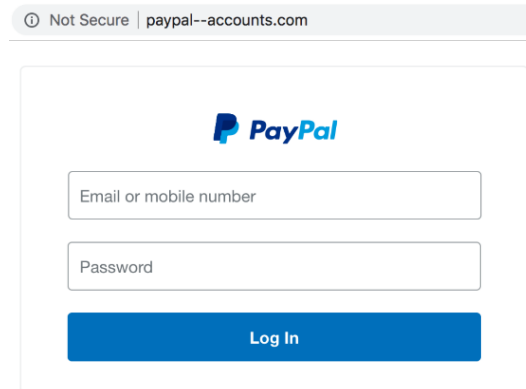


Figure 2.6b. A PayPal phishing webpage.

It is however impossible for a phishing webpage to use the same URL as that of the legitimate webpage. This is because the website’s domain name is a unique registered information for every website in the internet space. In an attempt to imitate the original URLs or to hide true identity of their suspicious URLs, phishers use two approaches. The first one is to compromise a webserver hosting a legitimate website and add a phishing webpage or website in the folder containing the legitimate website. The phishing webpage/website, in this case, will be running as webpage(s) of the legitimate website, thus, using the same domain name registered with the legitimate website but with different URL path(s) assigned by the phisher.

The second approach is to use their own registered domain names but make them look similar to the legitimate ones or mask them using other unsuspecting characters. Various techniques are used to achieve this. These include the use of legitimate domain names in non-standard positions in their URLs [118], replacing domain names with numerical digits or IP addresses [118], encoding the domains names with other string presentation formats (e.g ASCII characters) [119], addition of various obfuscation characters (e.g “-“, “_”, “=”) [120], replacing their original long URLs with shortened ones [100] and the use of domain names provided by free web hosting services [23]. These practices are not commonly associated with legitimate

URLs especially the PDC webpages. Due to the addition of more characters by some of these techniques, the resulting phishing URLs tend to be longer in length than those of legitimate websites. Also, in most legitimate websites, the domain names often represent brand names or names of the organizations owning the websites. These names usually appear multiple times in the contents or structural components of the webpages. Since in this approach the phishers' own domain names are different from the legitimate ones, they are less likely to relate to the contents or structural components of the phishing webpages, which are often copies of those of legitimate webpages.

Based on the structural designs of the phishing websites described above and the current trends in the operations of the phishing websites described in section 2.2.4, a number of key structural differences between phishing webpages/websites and those of legitimate ones can be noticed. We summarize them in Table 2.2. These can be useful in determining potential features for differentiating the two thus predicting phishing PDC webpages.

Comparison Factors	Legitimate Webpages/Websites	Phishing Webpages/Websites
<i>Originality of webpage structure and contents</i>	Use their original structure and contents	Copy most of the structural components and contents of the target legitimate webpages including webpage links. Few links may be modified such as that of a form handler
<i>Relationship between domain name and the brand/organization name of the website</i>	Domain names often relates with the brand names	Domain names registered by phishers do not relate with the brand names
<i>Presence of a website's domain name in non-standard positions in the URL</i>	It is not a standard practice	This is a common practice in phishing URLs
<i>Use of numerical digits or IP addresses for a domain name</i>	It is not a common practice	It is a common practice
<i>Encoding domain names</i>	It is not a common practice	It is a common practice
<i>URL length</i>	Often short	Often long

<i>Use of domain names offered by free web hosting services</i>	Those owned by the established organizations are not expected to use free domain names	There is a growing number of phishing websites using free domain names
<i>Use of shortened URLs</i>	PDC webpages are not expected to use shortened URLs	There is a growing trend of phishing PDC webpage using shortened URLs
<i>Domain name life span</i>	Most of the established organizations are expected to have been using the domain name for long time	Phishers often use their registered domain for short period to avoid being tracked
<i>Use of digital certificates</i>	Most of domain names are expected to use digital certificates especially the EV certificates.	Most of the domain names still do not use the certificates. For those using them, OV and DV types are the common ones.
<i>Number of websites sharing a host</i>	Sharing of a host is less expected to the majority of websites	A large number of phishing websites sharing a host is a normal practice

Table 2.2. A summary of key structural differences between legitimate and phishing websites.

2.3.2. Domain Name System (DNS)

The DNS is a distributed database containing resource records which map human-readable names of web services to IP addresses [121, 122]. Its purpose is to enable users to access the services without the need to memorize complex IP addresses. The DNS name space is organised hierarchically and forms an inverted tree with nodes representing a component of the entire domain name, also referred to as Full Qualified Domain Name (FQDN). The FQDN can be obtained by concatenating names of the nodes on the path from the subtree to the root using dot as a separator. The root domain is represented by ‘.’, but this may be omitted in many contexts. Figure 2.7 shows an example of domain tree in which *cs.barkkeley.edu* and *barkkeley.edu* are FQDN (often referred to as hostname) and Second level Domain Name (SLD) (often referred to as domain name) respectively.

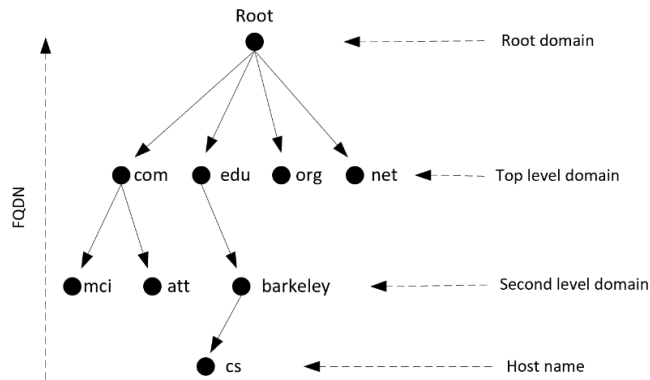


Figure 2.7. An example of a domain tree for the FQDN *cs.barkeley.edu*. [121].

The DNS domain name space is partitioned into zones, each of which is managed by a zone administrator. Information about the name space is held in name servers (NSs) in the form of Resource Records (RRs). An NS that has a complete information about a specific zone is said to be an authoritative one for that zone. An NS may be authoritative for multiple zones. Examples of RRs stored in NSs are A records (records of domain names against IP addresses), NS records (records of names of NSs against their IP addresses) and MX records (records of names of mail exchange servers against IP addresses) [122].

In a typical web browsing scenario, an FQDN is extracted from a URL by the web browser, which sends an A record query to a DNS client (resolver) in the user's local machine. The resolver looks, recursively, for an answer (mapped IP address(es)) in various intermediate NSs of the FQDN's zonal DNS hierarchy [122] as illustrated in Figure 2.8. If the matching record is not found in the intermediate nodes, an authoritative NS returns the answer. The answer may contain one or multiple IP addresses depending on the number of machines hosting the web service, where each machine corresponds to one IP address. The browser then sends an http request (say) to one of the IP addresses to download the requested webpage. Answers for DNS queries are cached by intermediate NSs (i.e. NSs that are not authoritative for the relevant zone) for specific periods. The time remaining until a cached record is deleted is known as the Time to Live (TTL) [122]. The TTL value associated with an authoritative RR is a recommendation for the maximum time for which the record should be cached. Caching improves the efficiency of the DNS when conducting subsequent related searches by reducing the number of queries, and hence time, required to yield answers from authoritative NSs.

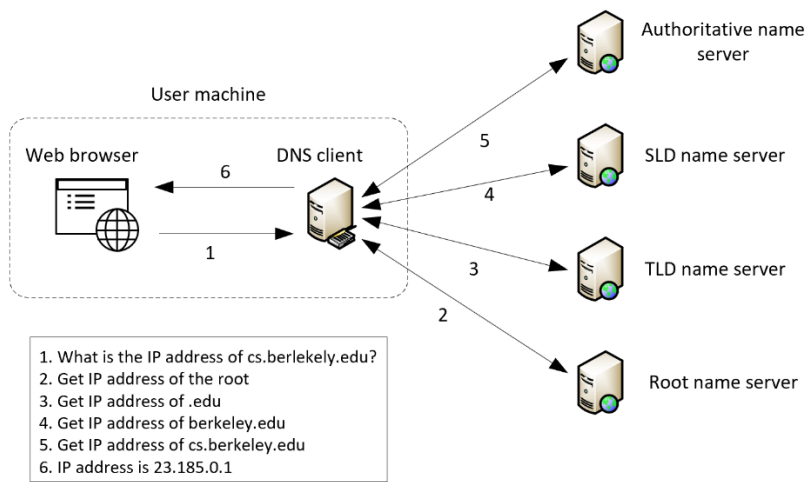


Figure 2.8. An example of a DNS recursive query operation.

As mentioned above, a single FQDN may be associated with, and hence resolved to, multiple addresses. The purpose of hosting a service in multiple hosts may, for example, be to provide redundancy for resilience purposes or to share processing load across the hosts. To achieve the latter, a round robin mechanism is used by default. Suppose the NS returns to the DNS client all the IP addresses mapped to the queried FQDN arranged in a particular order. The client picks one or a few top ranked addresses to serve the user’s request. The NS then re-arranges the order in a loop manner by sending the top addresses to the end of the loop. The client thus selects new top address(es) when the next query for the same FQDN is made [122]. Figure 2.9 illustrates this mechanism.

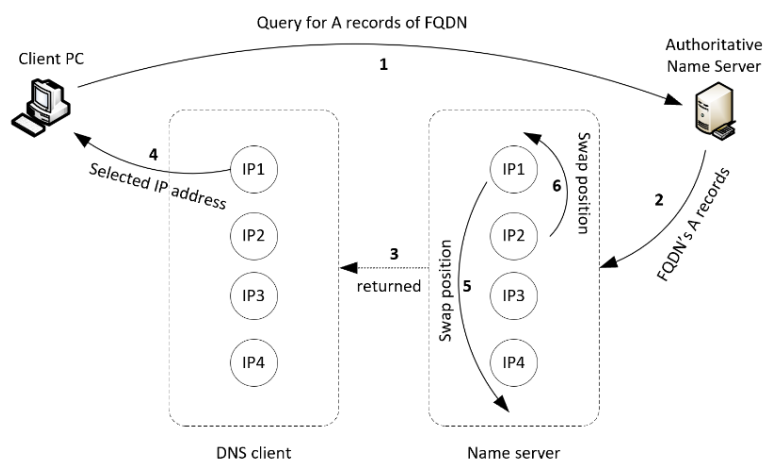


Figure 2.9. A round robin mechanism.

2.3.3. Structural Characteristics of Networks Hosting Phishing and Legitimate Websites

2.3.3.1. Host and Name Server Non-Flux Networks

Many legitimate and phishing websites are still hosted in traditional setups in which a hostname of a website is resolved consistently to the same IP address(es) of one or a few servers. If multiple servers are used, they are often located in a few locations and hosted in the same network/subnet or a small number of different networks/subnets. Except for occasional and often temporary changes due to events such as maintenance or upgrading of services, the A records of the servers do not change, resulting in repeated queries returning the same set of addresses. Since the records rarely change, the TTL is set relatively high. The TTL value for an A record that changes infrequently is often between one and three days [123, 124]. We refer to a network with such a behaviour as a *non-fluxing network*.

To avoid being easily tracked, attackers are increasingly hosting their web services in the compromised web servers, which also host legitimate services. Up to 51% of all phishing websites, for instance, are hosted in the compromised hosts while the remaining 49% are hosted in servers owned and managed by attackers [39]. As a means of optimizing their limited resources, attackers usually host multiple malicious web services in one machine [25, 41]. Other attackers use proxies to hide identities of their hosting machines [125]. In this case, when A records of hostnames of web services are queried, IP addresses of the proxies are returned. Upon receiving a data access request from a user, the proxy contacts the actual phishing server to retrieve the content and passes it to the user. Though there are legitimate uses of proxies including protecting hosts from being probed by hackers, and caching and traffic filtering [126], proxies are more likely to be deployed by attackers.

In legitimate NS non-flux networks, a network is comprised of one or a small number of authoritative NSs with fixed (static) IP addresses. The addresses are made static in order to maintain high reliability of the services. The authoritative NSs of a zone keep RRs for the zone in dedicated high-performance servers. The servers are usually not used to run other services so as to ensure all computing resources are utilized to serve large volumes of requests of the NS records at a high speed. To enhance high availability and load balancing of NS services, the NSs are often setup in a redundant network of servers which keep the same records at all

times. In a such setup, each zone has a primary NS, which keeps the master RRs and at least one secondary NS which keeps a copy of the master RRs [122]. The servers are often installed in different locations such as in different buildings, area, cities or even countries in order to avoid all servers being out of service simultaneously as a result of incidents such as natural disasters. Changes in the IP addresses associated with an NS name can happen on rare occasions due to maintenance operations or up-scaling of the services. Frequent variations of IP addresses are not expected in legitimate authoritative NSs. Legitimate authoritative NSs are managed by domain registrars or organizations such as Microsoft which often have a large number of domains hosting a wide range of their web services.

Malicious NS non-flux networks behave similar to the legitimate ones in most of the characteristics. In such networks in which the NSs are owned and managed by attackers, the hosts, however, are often less powerful machines and are used to host multiple malicious web services including other NSs and websites as a means of optimizing attackers' limited resources.

2.3.3.2. Content Delivery Networks (CDNs)

Content Delivery Networks are networks of web server farms in dispersed geographical locations often in different cities, countries or continents [127]. Their purpose is to deliver web contents efficiently to users scattered over a wider geographical area. Copies of the same contents are cached in multiple farms and a user's request is served from the farm that can provide a copy most efficiently [128]. An A record query citing the hostname of a hosted web content will return IP addresses belonging to one or multiple servers hosting the content. CDNs use sophisticated techniques, based on factors such as geographical distance, network topology and link health, to select and return the IP address(es) of the most efficient server(s). In most cases, the closest server farm is selected from which one of its servers, based on the configured metrics such as server load level and/or round robin mechanism, is chosen to deliver the content to the user. As servers are chosen dynamically, short TTLs are set for domains hosted by CDNs [128]. By delivering contents from the farm closest to the user and using redundant setups, CDNs improve significantly data access speed while ensuring high availability of services [127, 128].

CDNs are owned and operated by a small number of organizations for the purpose of hosting and distributing their own contents or leasing the infrastructure to other content providers as a service [127]. For example, large organizations with complex web service ecosystems such as Google and Netflix own their specialized CDNs while others including Akamai and Limelight provide a content delivery service to mid-sized content providers using unspecialized CDNs [127]. Their server farms are usually established in several locations across the globe to serve local users. Content is hosted and accessed directly from the servers, with servers being assigned static IP addresses. In order to deliver large volumes of contents to many users with high efficiency, high performance specialized servers and web server software are used [127]. Servers and software used in data centres of a given CDN are largely of a uniform standard, with an aim of facilitating smooth technical management of the infrastructure. Due to intensive high-performance operations performed 24/7, maintenance operations to replace the worn or faulty servers are frequent. Figure 2.10 below shows a general CDN architecture and a high-level description of operations.

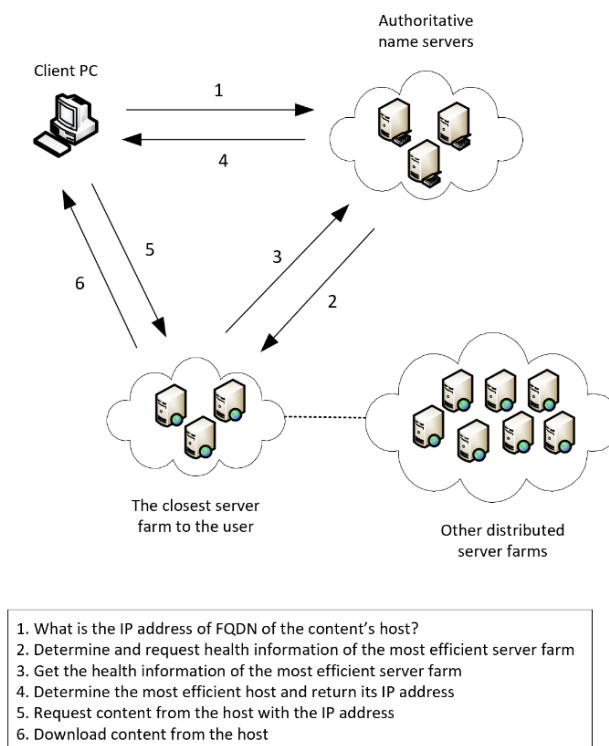


Figure 2.10. Architecture of CDNs.

2.3.3.3. Fast Flux Service Networks (FFSNs)

Fast Flux Service Networks are essentially networks of compromised machines which are managed by attackers to host and distribute malicious web services such as phishing websites, spams, malware and denial of services [129]. Typically, an FFSN consists of one or a few servers hosting the malicious content, one or a few command-and-control servers (also known as motherships) and a large number of distributed compromised machines (also referred to as flux agents) [130]. The motherships, in some cases, double up as content servers. FFSNs recruit flux agents by randomly infecting vulnerable machines from various networks with a malware distributed by the motherships. Due to the random process of malware infection, agents in FFSNs are usually from many different networks across the globe [123, 131, 132]. After infecting the machines, the malware creates backdoors which are then exploited by the motherships to remotely control and coordinate the machines to undertake malicious operations [123, 131]. A records mapping hostnames of the web services hosted by an FFSN to IP addresses of the agents it controls are registered in the DNS. Such hostnames are referred as *Fast Flux (FF) hostnames* for reasons that will become apparent. The A records of the agents are registered with authoritative NSs managed by domain registrars with weak validation checks or the ones owned by attackers managing the FFSNs. A typical FFSN may consist of a few hundreds to millions of agents [130, 133].

When an A record of the hosted hostname is queried, the IP addresses a subset of the agents are returned. Using a round robin DNS mechanism as well as instant availability checks of the agents, similar consecutive queries on the same hostname return different subsets of IP addresses, thus different agents [130]. It is this rapid change of IP address that a hostname resolves to that leads to the term fast flux. Users then request contents from the returned agents which forward the requests to the content servers. The contents from the servers are sent back to the users through the agents. The agents, therefore, act as proxies in communications between users and the servers. By so doing, they hide visibility of the servers from the user side, thus protecting the servers and the motherships from forensic investigations and blacklisting [130]. Only a subset of flux agents is exposed at a time so that even if they are detected and taken down, the FFSN can continue operation. Furthermore, since the motherships will continue to compromise and recruit new machines, the losses will quickly be replenished [129]. Use of FFSNs, therefore, aids attackers to maintain their web services for longer

compared to normal (non-flux) networks, making them more harmful. Figure 2.11 illustrates the general architecture and operations of an FFSN.

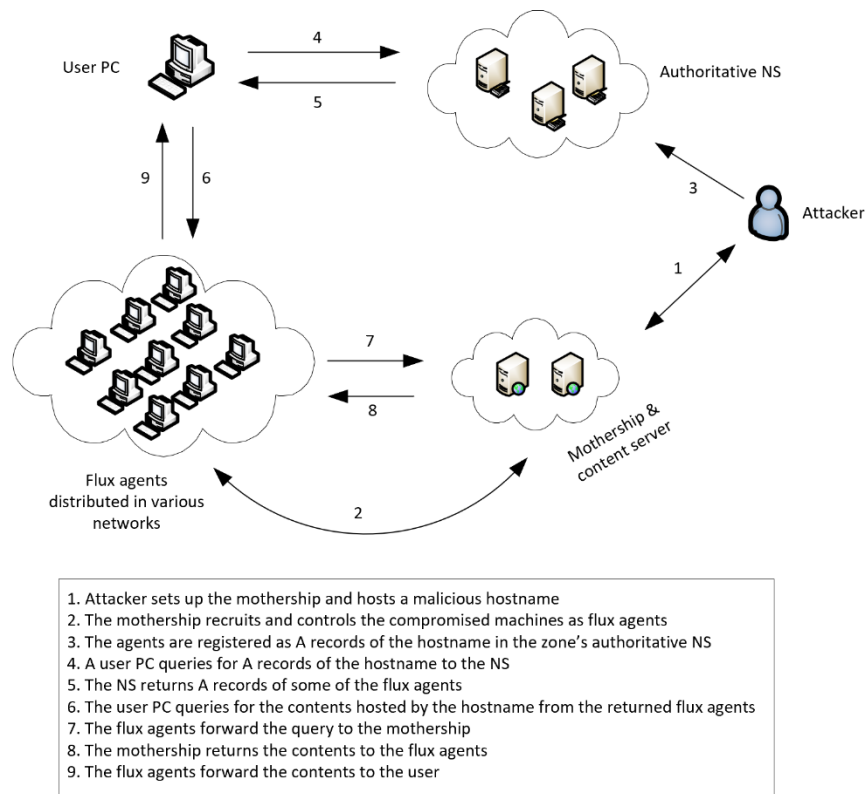


Figure 2.11. Architecture of FFSNs.

Flux agents are mostly standard computers and Internet of Things (IoT) devices in homes and small office networks [134-136]. This is because they are often less well secured and maintained, thus have many security vulnerabilities which can easily be exploited by FFSN malware [137]. Since most of the agents are machines for personal use, they are often switched off when not in use [130]. Thus, their availability to FFSNs at any time is unpredictable. FFSNs constantly check the availability of the agents and remove from their records those which are no longer reachable or have been disinfected [129, 131]. To maintain the size of the networks or to scale up, FFSNs constantly recruit new agents. Due to the constant fluctuations of the agents' availability states, attackers assign short TTLs to the A records of the hosted hostnames in order to force IP addresses of the currently-available agents to be retrieved from the authoritative NSs and returned to users [130]. To further protect their networks against detection and take down, other FFSNs conceal identities of their flux agents by using proxy applications in the agents [138]. Due to the effectiveness of FFSNs, non-technical attackers are

increasingly renting FFSNs as services to increase their productivity, resulting in the rise of number of unique attacks (phishing websites in this context) operated by one FFSN provider [59, 139].

Given the distinctive operational and structural characteristics of the four types of website hosting networks described above, we summarize their expected differences in Table 2.3. These will be useful in identifying potential features for distinguishing the hostnames hosted in these networks (i.e FFNSs, CDNs, legitimate non-flux networks and phishing non-flux networks) as described in detail in Chapter 5.

Comparison Criterion	Legitimate Non-Flux Networks	Malicious Non-Flux Networks	CDNs	FFSNs
<i>How long is TTL</i>	Long	Long	Short	Very short
<i>Number of IP addresses in the A records</i>	One or small	One or small	Moderately large	Very large
<i>Likelihood that hosts use proxy applications</i>	Likely	Very likely	Likely	Very likely
<i>Computing power and specialization of hosts</i>	Hosts are powerful specialized servers with the same operating system (OS)	Hosts are powerful standard machines or compromised specialized servers often with the same OS	Hosts are powerful specialized servers often with the same OS	Most flux agents have the least computing power and are for generic basic uses. They often have different OSs
<i>Co-hosted web services</i>	Each server often hosts a dedicated web service	Servers often co-host a large number of different malicious web services	Each server often hosts a dedicated web service	Each flux agent may host/proxy a large number of different malicious web services
<i>Server uptime</i>	Very long	Short or long	Moderately long	Very Short
<i>Host ownership</i>	The hosts are owned by owner of a network or service	The hosts are owned by attackers or owners of the	The hosts are owned by owner of the CDN	Flux agents are owned by owners of the compromised machines

		compromised machines		
<i>Geographical distribution of hosts</i>	Hosts are in one or a few locations	Hosts are in one or a few locations	Hosts are dispersed over several strategic locations	Flux agents are widely and randomly dispersed across the globe
<i>Distribution of hosts in networks</i>	Hosts often are in one network administered by one organization	Hosts often are in one network administered by one attacker	Hosts are in one or a few networks often administered by one organization	Flux agents are spread over many networks administered by different individuals or organizations
<i>Control over scalability of the network</i>	The network's owner has a total control over scalability of the network	The network's owner has a total control over scalability of the network	The CDN's owner has a total control over scalability of the network	FFSN's owner does not have a total control of network scalability
<i>Network size</i>	Pre-determined and mostly fixed	Pre-determined and mostly fixed	Pre-determined but occasionally varies	Not pre-determined and constantly change over time
<i>Matching of IP addresses with hosts known to host blacklisted phishing websites</i>	Small to large counts of IP addresses matched	Large counts of IP addresses matched	None or small counts of IP addresses matched	Large counts of IP addresses matched

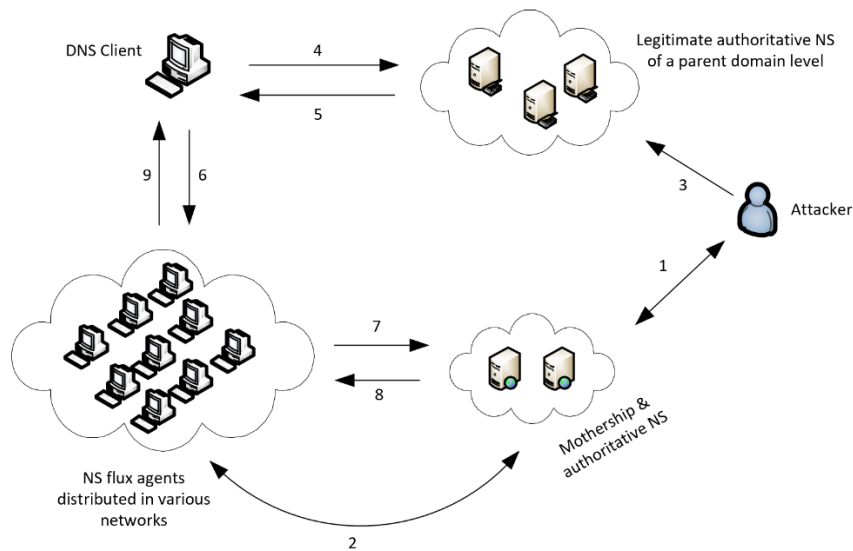
Table 2.3. A summary of the expected differences in the structural and operational characteristics of the four types of the website hosting networks.

2.3.3.4. Name Server Flux Networks

Studies including Pa, et al. [140], Metcalf and Spring [141], Kadir, et al. [142], Konte, et al. [143] and Salusky and Danford [136] observed that some of the authoritative NSs registering RRs of malicious web services of a particular zone exhibit a fluxing behaviour in which their IP addresses or names change at a rapid rate. We refer to these phenomena as NS IP flux (NSIF) and NS name flux (NSNF) respectively. In NSIF networks (NSIFNs), similar to FFSNs, the NS infrastructure consists of a mothership NS and a group of NSs which behave as flux agents

of the network. The former, which is owned and managed by an attacker, is the actual authoritative NS for one or more zones of malicious web services while the latter acts as proxies to the mothership by forwarding NS queries to the mothership and returning the responses to the DNS clients. The NS flux agents are the compromised machines, often from a wide range of global networks, which are recruited by a malware managed by the attacker in the mothership. The A records mapping names of the malicious NSs to IP addresses of the flux agents are registered by the attacker in the authoritative NS of a parent zone. The flux agents are rapidly rotated to avoid blacklisting of IP addresses of the agents, thus the entire network. To make this approach possible, as in the case of FFSNs, attackers often assign shorter TTLs to NS records of their malicious web services.

Similar to NSs in malicious NS non-flux networks, the agents and the motherships are often used to host multiple malicious NSs and websites. Figure 2.12 shows the general architecture of NSIFs. Based on the descriptions provided above, Table 2.4 summarizes the expected operational and structural differences between the three types of networks hosting authoritative NSs (i.e phishing NS flux networks, phishing NS non-flux networks and legitimate NS non-flux networks). The differences in the characteristics will be used to derive features for distinguishing hostnames hosted in each of these networks in Chapter 5.



1. Attacker sets up the mothership and registers DNS records of hostnames he owns in it
2. The mothership recruits and controls the compromised machines as NS flux agents
3. The agents are registered as authoritative NSs of malicious hostnames in the NS of a parent domain level
4. A DNS client queries for A records of an authoritative NS of a malicious hostname
5. The NS of the parent domain level returns A records of some of the agents
6. The client queries for A records of the hostname from the returned flux agents
7. The flux agents forward the query to the mothership
8. The mothership returns the answer to the flux agents
9. The flux agents return the answer to the client

Figure 2.12. Architecture of NSIFs.

Key Differences	Legitimate NS Non-flux Networks	Phishing NS flux Networks	Phishing NS Non-flux Networks
<i>Number of A records per NS record</i>	Medium	Large	Small
<i>TTL of NS records</i>	Long	Short	Long
<i>Network distribution of NSs</i>	Small/medium number of unique networks or one network	Large number of unique networks	Small number of unique networks or one network
<i>Geographical distribution of NSs</i>	Small/medium distances between locations of NSs	Large distances between locations of NSs	Small distances between locations of NSs
<i>Co-hosted web services</i>	Do not co-host other web services.	Large number of co-hosted malicious web services	May host small to large number of malicious web services.
<i>NS hosts</i>	High performance servers with the same OS	Standard machines with different OSs	Standard or medium performance machines with the same OS
<i>Server uptime</i>	Long	Very short	Short or long

<i>Matching of IP addresses with hosts known to host blacklisted phishing websites</i>	No matching of IP addresses is expected	Large counts of IP addresses matched	Small to large counts of addresses matched
--	---	--------------------------------------	--

Table 2.4. The expected structural and operational differences between the three types of networks hosting authoritative NSs.

NSNFs were observed by Kadir, et al. [142] and Konte, et al. [143] in which, instead of using NS flux agents, the attacker frequently changes a name of an authoritative NS registered at the domain registrar. This approach, however, is not within the scope of this study. Since there is no any reason for legitimate NSs to change their IP addresses or names frequently, domain registrars could adopt monitoring and auditing practices that would detect such behaviours and report or refuse to provide services to the NS owners. Many domain registrars do not do this, however, thus encouraging the growth of NS flux networks for malicious operations.

Konte, et al. [143] and Salusky and Danford [136] observed that some of the malicious hostnames are hosted in networks which exhibit both fast flux and NS IP flux behaviours. In these networks, A records of a hostname and an authoritative NS of a website are frequently changed as independent operations. Such networks are also referred as *double flux networks*. In addition to the two behaviours, Kadir, et al. [142]'s study also observed that some of the hostnames are also hosted in networks with an added layer of a NS name flux behaviour.

2.4. Summary

This chapter has provided an overview of phishing website attacks. It has described the evolution of phishing websites, the six common techniques that attackers use to distribute the websites to their targets, the five main stages of executing the phishing website attacks and the recent developments in the attacks. The chapter has also described the general anatomy of legitimate and phishing webpages that collect personal data from users. Finally, the chapter has provided an explanation of the structure and operation of DNS and the networks hosting malicious websites and their authoritative NSs.

In this chapter, we have learnt that phishers are increasingly using sophisticated phishing toolkits and evolving tactics to develop and distribute complex phishing websites. This has led to growth in the number of non-technical phishers, the number of zero-day phishing websites created on daily basis and the difficulty of detecting the websites. The type of information being stolen is also diversifying, from basic credentials to new categories such as family and employment records, in order to extend new avenues of malicious operations. In addition, phishers are shifting to target data that is commonly used for authentication in many applications through third party authentication services (e.g Facebook and Google), a move which is likely to multiply the impact of the attacks to the online community. These developments are making phishing one of the most concerning security threats to the internet community. However, we have observed that phishing websites have differences in some of their structural characteristics from those of legitimate websites. The differences are based on the URL structure, webpage structure and contents, and the third-party information related to the website. Also, we have learnt about some of the differences in the way flux networks hosting phishing websites and those hosting authoritative name servers of the websites are structured and operate from non-flux networks. Using these differences, we are going to derive features that can predict phishing PDC webpages (in Chapter 4), and phishing hostnames hosted in FFSNs and NSIFNs (in Chapter 5) in order to protect users from falling to phishing website attacks. In the next chapter, ML techniques that will be used in this thesis are reviewed.

Chapter 3

Machine Learning Techniques for Phishing Website Detection

3.1. Introduction

Machine Learning (ML) is a branch of Artificial Intelligence (AI) which enables computer applications to learn from data about historical events in order to predict outcomes of future similar events [144]. The learning is performed automatically by statistical algorithms whereby relationships between dependent and independent data variables are extracted by the algorithms to develop rules for prediction. Given sufficient amount of quality data and computing power, the algorithms are able to produce, at a fast speed, accurate predictions in complex problems without human intervention.

The deployment of ML to solve problems across various domains has increased sharply in recent years due to the availability of large datasets and powerful computing machines. One of the domains that has been a beneficiary of ML is cybersecurity. Specific areas of cybersecurity which have seen the significant use of ML include network intrusion detection, botnet detection, malware detection and analysis, detection of malicious websites, Internet of Things (IoT) security and credit card fraud detection [145-149]. Classification is one of the most popular branches of ML used in cybersecurity especially in the detection of phishing websites. In classification, an ML algorithm learns the relationships between data inputs and class labels of previous events to predict class labels of future events. A focused review of the application of classification-based ML for the detection of phishing websites is provided in Chapters 4 and 5. In this chapter, we aim at providing background on ML and classification as a foundation for the explanation of our ML approaches for predicting phishing websites in Chapters 4 and 5.

This chapter is divided into six sections. Section 3.2 provides an introduction to ML and describes different types of ML. In section 3.3, classification-based ML is described. The section also explains the two main types of ML classifications (binary and multi-class classifications) and the concepts of flat and hierarchical classifications for implementing binary and multi-class classification problems. Section 3.4 describes the standard workflow for undertaking of an ML task. Section 3.5 describes some of the most popular ML algorithms

used for classification tasks. We describe the algorithms into two groups namely traditional ML algorithms and Deep Learning (DL) algorithms. Section 3.6 summarizes of the chapter.

3.2.Overview of Machine Learning

Artificial intelligence refers to the creation of human-like intelligence, through computer systems, which can learn, reason, plan and solve complex tasks [150]. AI allows automation of repeated tasks, and evaluation of contexts and criteria to make best decisions at a high speed and accuracy [151]. With AI, complex problems which cannot be tackled by traditional human methods can be solved with high efficiency. Complexities of the problems are due to large sizes of data often consisting of different types and of mixed shapes (structured and unstructured), forming many patterns and correlations that can hardly be learned by humans [152].

Predictive AI, also referred as *machine learning*, is the most applied form of AI today. It is used to predict future outcomes of a problem by learning historical data of the similar problem [150, 153-155]. It explores the historical data, also known as training data, to discover hidden patterns and correlations among data fields and use them to automatically develop rules to predict future events [150, 151]. The rules are generated by specialized statistical algorithms though probability computations, in which the predicted outcomes are determined by the highest probability of occurrence of events [156]. ML uses the most relevant variables (features) of the training dataset to establish the prediction rules. The accuracy of prediction outcomes depends on the relevancy of the features used and the sizes of training data [156]. The more relevant the features and the larger the data, the higher the prediction accuracy is achieved. ML has been found to be an ideal approach in solving complex problems involving large datasets with many variables of different data types and structures [150, 157].

According to Internet Society [150] and Somvanshi and Chavan [154], ML is categorized into three main types as described below (Figure 3.1 illustrates the categorization);

- *Supervised Learning (SL)*

In this category, the training data consists of instances with known outcomes (class labels) [156]. An ML algorithm learns patterns of the mapped instance-outcome sets of previous

instances to develop an approximation function for predicting class labels of future instances. SL is useful in solving classification and regression related problems [152].

- *Unsupervised Learning (UL)*

UL is useful in solving problems in which outcomes of input instances are not known (unlabelled data). UL learns the data and develops similarities among the instances, based on certain features, to map the related instances. Prediction outcomes are presented as groups of previous instances which the new instances are most related with [152, 154]. Clustering and association methods are examples of applications of UL [156, 158].

- *Reinforcement Learning (RL)*

RL learns historical instances and make predictions of class labels of new instances through trials and errors. The consequences of actions due to the predicted results are then graded with success or failure scores and are fed back to update the algorithm to learn predictions of other instances [150, 152, 159]. Using feedbacks as historical experiences, RL systems continuously re-learn to improve their prediction accuracies while they are open to discover new unlearned experiences. However, RL requires large data to improve prediction rules over time and a high level of expertise to implement it.

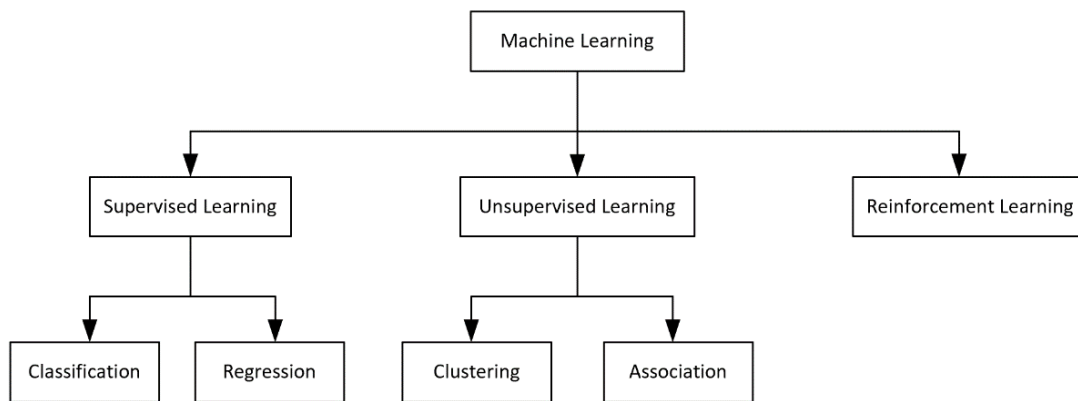


Figure 3.1. A chart summarizing the categorization of ML.

3.3. Machine Learning for Classification Tasks

Classification is a supervised ML approach in which an ML algorithm predicts class labels of new instances based on the mapped data inputs and class labels of previous similar instances. The algorithm learns from previous instances to generate a mapping (approximation) function $f(x) = y$ which maps input variables (x) to a class label y . Given a new instance, its predicted

outcome is first computed by the function as a predicted probability of an instance belonging to each output class. A predicted probability is then translated into a class value by selecting the class label with the highest probability [160]. The algorithm which classifies instances of a specific problem is known as *classifier*.

3.3.1. Binary and Multi-class Classification

There are two main types of classification tasks namely *binary classification* and *multi-class classification* [161]. In binary classification, a class label of the predicted instance can be only one of the two possible class labels at a time. For example, in a spam email detection problem, the predicted email can be either spam or non-spam. Binary classification is the simplest form of classification and requires only one classifier to separate the two classes. All ML algorithms for classification tasks supports binary classification.

Multi-class classification refers to the task in which a problem has more than two class labels whereby only one class label is predicted at a time. An example of this is a facial recognition problem in which one's face is identified out of many possible faces. This is a more complex and difficult task to address. Several ML algorithms for multi-class classification have been proposed in the literature which work by constructing a decision function that predict the class of a new point out of N different classes. Examples of such algorithms are k -Nearest Neighbours, Naïve Bayes, Decision Tree and Random Forest [161]. However, multi-class classification problems can also be solved by transforming the problems into multiple binary classification tasks whereby the tasks are solved using binary classification-based algorithms. In this case, multiple binary classifiers are implemented through either *one-vs-all (OVA)* or *one-vs-one (OVO)* methods [162, 163]. Logistic Regression and Support Vector Machines are examples of such binary classification algorithms [161].

3.3.2. Flat and Hierarchical Classification

In most cases, binary and multi-class classification problems are addressed with an assumption that target classes are not related to each other [164]. In such problems, also known as *flat classification* (illustrated in Figure 3.2a), any existing relationships between classes are ignored and the classes are regarded as the same level entities. One binary or multi-class classifier is developed and run once to classify instances into their respective classes. However, in many

real-world problems, target classes are related in a tree like hierarchy, in which some of classes share common features. Such classes can be grouped in a hierarchical structure to form parent-child class relationships [165]. This approach is also known as *hierarchical classification* (illustrated in Figure 3.2b). It is used to perform multi-class classification tasks.

Hierarchical classification involves a combination of binary or multi-class classifiers at different nodes or levels of a hierarchy, from top to bottom, to deliver the overall prediction of the tree leaf classes. An advantage of modelling the problem using this approach is that the structure gives detailed understanding of the problem and modularizes the classification process into separable components of nodes or levels which can independently be managed and optimized [165, 166]. Also, various research works have shown that hierarchical classification approaches, in some problems, can perform better than flat classification approaches [165]. Therefore, it is important that for problems with related target classes, both types of classification approaches are considered and compared in order to determine optimal prediction performances.

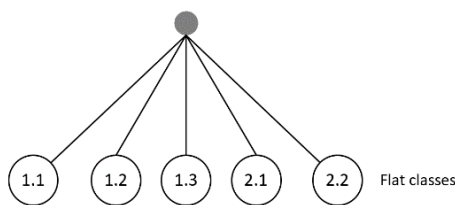


Figure 3.2a. No parent-child relationships in flat classification.

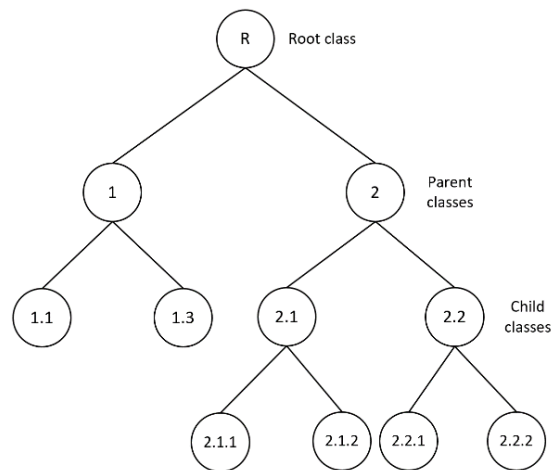


Figure 3.2b. Parent-child relationships in hierarchical classification.

One of the most recommended techniques in implementing hierarchical classification is Local Classifier per Parent Node (LCPN) [165, 167]. Others are Local Classifier per Node (LCN) and Local Classifier per Level (LCL). The advantages of LCPN over the other techniques are (1) It is simple to build, visualize and understand the parent-child relationships as the number of classifiers is always equal to number of nodes [167], (2) It also avoids class prediction

inconsistence problem when performing top to bottom prediction by limiting training dataset of the parent classifiers to instances of their child classes only [165] and (3) LCPN has performed well against flat classification approach in some problems [168].

In LCPN, at each parent node, a binary or multi-class classifier is built to classify the parent's child nodes. During training, different datasets and selected features are used to train the node classifiers, making the classifiers in the same hierarchy to produce different prediction performances. In predicting a new instance, the instance is predicted in a series of the classifiers from top to bottom in which the results of the parent classifiers become the inputs of their child classifiers [165]. The approach of using different ML algorithms to implement node classifiers, instead of using the same algorithm throughout the hierarchy, has shown to produce better results in many scenarios [169, 170]. In this approach, also referred to as *selective classifier*, training of datasets, feature selection and evaluation of the best performing ML algorithm are done at each node independently. Given the differences in the composition of datasets and best selected features at the nodes, different best performing algorithms are likely to be determined, resulting in optimizing each node thus the entire hierarchy.

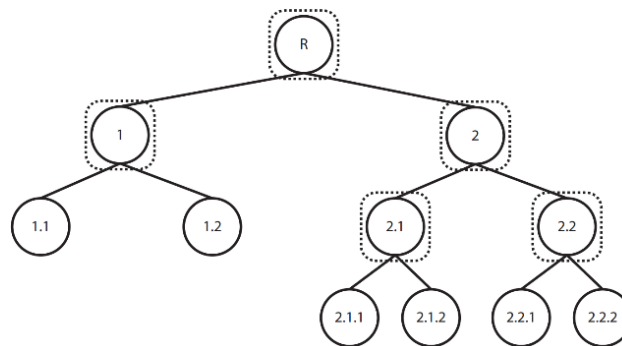


Figure 3.3. LCPN approach (dashed squares represent binary or multi-class classifiers).

The downside of LCPN is that the accumulative prediction performance is degraded as the classification is performed from parent to child classifiers [165]. Kowsari, et al. [168] suggested that accuracy of the child classifier is the fraction of the accuracy of its parent classifier. For instance, from Figure 3.3, accumulative accuracy at node 2.2.2 equals to accuracy of classifier 2.2 multiplied by accuracy of classifier 2. The final classification errors are the results of accumulation of errors of the individual classifiers from top to bottom and are obtained by summing up their errors [165].

3.4. A Standard Workflow for Machine Learning Tasks

This section describes important steps that are used to guide the design and implementation of a ML prediction model for a given problem. Figure 3.4 summarizes the steps which are discussed in the next sections.

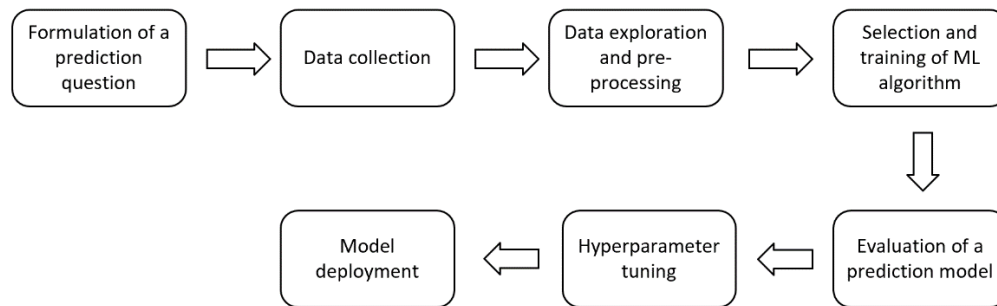


Figure 3.4. A summary of key steps for developing an ML prediction model.

3.4.1. Formulation of a Prediction Hypothesis/Question

In this step, a scientific guess or a question on a particular business problem that ML can help to predict an answer (outcome) is defined [171]. Measurable variables (prediction features) that are likely to affect the target outcome are identified and defined. The problem is categorized in terms of a type of ML task (described in section 3.2) in order to determine a suitable model architecture and an algorithm to use for the task. At this point, performance metrics to measure success or failure rates of the prediction model for the problem are also identified [172].

3.4.2. Data Collection

At this step, data based on the variables determined in the previous step is collected. In order to ensure high quality of prediction results, only data of variables that are likely to influence the target outcome and of sufficient size should be collected for training the algorithm. After determining the type of data needed and its sources, suitable data collection mechanisms are designed and built for the process.

3.4.3. Data Exploration and Pre-processing

Typically, real-world data is incomplete, inconsistent and inaccurate. This step plays a role of transforming raw data into a quality shape through cleaning, formatting, and organizing it so that ML algorithms can efficiently learn it to build an accurate prediction model [173, 174]. This section highlights key data pre-processing activities one is expected to perform when undertaking a ML task.

3.4.3.1. Exploratory Data Analysis (EDA)

This process involves visualizing, summarizing and interpreting the information that is hidden in rows and columns of data. The goal of EDA is to identify errors, outliers and anomalies as well providing insights of patterns and relationships between variables that can inform more about the problem [175]. EDA can be performed through non-graphical and graphical approaches. The former involves data manipulation and analysis such as data summarization and aggregation to generate various informative statistical measures. The latter uses graphical tools to present the statistical distribution of data. Various plots such as heatmaps, scatter plots, box plots and boxen plots are often used for this task. In this thesis, boxen plots have been mostly used for data analysis (in Chapters 4 and 5). Originally proposed by Heike, et al. [176], the plots show a distribution of all observations of a variable, including the outliers, using a large number of unique quartiles (boxes). The density of observations in each quartile is presented by the width of the quartile. The wider the quartile, the higher the density and vice versa. The horizontal black line represents the median. EDA is usually performed in most of the data pre-processing tasks (described below).

3.4.3.2. Data Cleaning

The goal of data cleaning is to provide accurate, consistent and complete sets of data instances for developing an accurate prediction model [173]. To achieve the objective, the process involves identification and handling of missing data, errors and meaningless data [177]. Handling missing data is important as many ML algorithms do not support such data. Most of the algorithms either perform very poorly or do not learn at all from the data. To handle the missing data, the following techniques are recommended;

- Drop the entire feature column or row if the percentage of missing values in the column or row is more than 50% [173].
- For features containing continuous values, replace the missing values with the imputed mean, median or mode values of their columns [178]. For features with categorical values, the missing values are often replaced with either the most frequent category or a new unique category [179].

It is quite often to find outliers in the real-world data. These are observation points that are distant from other observations. Since they significantly affect the statistical distributions of data, thus performances of the models, it is imperative that they are keenly investigated and treated accordingly. If they are actual events, they should be kept otherwise they should be corrected or deleted. Various techniques can be used to spot the outliers including normal data inspection by a domain expert, data visualization tools such as box and scatter plots, and a mathematical computation by z-score [180]. Methods that are often used to correct them including cap the values at a certain reasonable limit, replace the values with new imputed ones and transform the values into other forms [180].

3.4.3.3. Feature Engineering

This is a process of converting raw data into features that better represent a problem to the ML algorithms in order to optimize a prediction performance [181]. To achieve this objective, the process involves identification and the use of most relevant features for the problem [181]. The number of features (dimensionality) have a significant impact on the accuracy and computation time of prediction models. With high dimensionality, the models tend to learn patterns of the specific samples in the training data such that they become less accurate in learning new samples, a phenomenon also known as *overfitting*. Also, the computation time is increased drastically. In addition, some of the ML algorithms do not work with a large number of features [181]. To reduce the dimensionality, the following techniques are commonly applied;

- *Removing meaningless features.* These are features which do not influence the prediction results. Examples are the features with a zero or near zero variance and those with a high correlation coefficient [177]. Features which are highly correlated to each other tend to carry similar information, thus have the same impact to the prediction. For the prediction task, therefore, only one feature of each pair of features whose pairwise

correlation exceeds a given threshold is retained. Pearson's correlation coefficient is usually used to measure an extend of correlation between numerical features [181, 182].

- *Feature selection.* These are techniques that use scoring or statistical methods to remove irrelevant features from the prediction process. One of the popular methods is Backward Feature Elimination. In this method, the selected ML algorithm is trained on all n features. Then one feature is removed at a time and train the same algorithm on $n-1$ features. The feature whose elimination yields the smallest increase in the errors is removed, thus remaining with $n-1$ features. The training is then repeated using $n-2$ features and so forth. Each iteration k yields a model trained on $n-k$ features and an error rate $e(k)$. The smallest number of features which produces the largest tolerable errors is then selected as the best feature set for the prediction [182].

3.4.3.4. Data Transformations

The process involves conversion of data representations into the ones ML algorithms can learn efficiently. Several methods can be deployed for this process, the main ones include;

- *Encoding of Categorical Data.* With an exception of a few ML algorithms, most algorithms do not work or perform inefficiently with categorical (non-numerical) data [183]. Conversion of categorical data to numerical ones is therefore vital for the optimization of prediction results. Label encoding is one of the common methods used for encoding where a unique numerical value replaces each unique categorical value [184].
- *Scaling.* Most ML algorithms are sensitive to the magnitude and range of numerical data across various features [185]. Features with large magnitudes of values and those with a wide range of values often cause drastic changes to the learning coefficients, causing large prediction errors. Algorithms work well with small magnitudes of data, distributed within a small and similar range across various features [186]. Scaling is an example of the methods to transform the data range whereby data in each feature is converted into a range between 0 to 1 [187].

3.4.3.5. Handling of Imbalanced Data

Most ML algorithms were designed with an assumption that classes are equally balanced in terms of number of samples. However, the number of samples of different classes are usually imbalanced in most problems. With a severe imbalanced data, for instance a ratio of 1:4 or more, the algorithms tend to lean more towards the majority classes, therefore producing high accuracies for predicting the majority classes but poor accuracies for the minority classes. In many problems, however, it is common to find that the minority classes consist of the most important instances desired for prediction. To address the issue, two approaches are often used. One is to rely on many other performance measures besides accuracy, such as confusion matrix, precision and recall, to interpret the true performance. Alternatively, oversampling or under sampling techniques can be used to balance the number of samples for all the classes [183, 188]. In oversampling, samples of the minority classes are duplicated or synthetic samples are algorithmically generated to balance the number of samples of all the classes. Synthetic Minority Over-sampling Technique (SMOTE) is one of the most popular algorithms used for the latter [189]. In this technique, a random sample from the minority class is first chosen. Using distance measures, k nearest neighbours for that sample are selected (typically $k=5$). A random neighbour from the set of neighbours is chosen and a synthetic sample is created by an algorithm at a randomly selected point between the two samples in the feature space. Other synthetic samples are generated using the same approach [189]. In under sampling, some samples of the majority classes are removed. A combination of under sampling and oversampling techniques has also been observed to yield good results.

3.4.4. Selection and Training of ML Algorithms

ML algorithms were optimally designed for prediction based on the type of a prediction task, problem domain, data types of input variables and patterns of data distributions. For instance, deep learning algorithms such as CNN and LSTM (described in section 3.5.1) are optimized for problems involving image and sequential data respectively. An appropriate ML algorithm should be selected for a given problem, based on these factors, in order to achieve optimal prediction results.

3.4.5. Evaluation of a Prediction Model

The success of a prediction model is measured by its performance on unseen data. To achieve this, the training dataset is often divided into two sets, the training and testing data, for creating the model and evaluating it on unseen data respectively. Stratified k -fold cross validation is one of the most efficient methods used for this process. In this method, the entire data is divided into k equal sized subsets (folds) in which $k-1$ subsets are used for training and the k^{th} fold is used for testing [183]. This splitting is repeated k times such that each fold is used either for training or testing at different times. The overall average performance is obtained by computing a mean of scores of each rotation of k splitting. For modest sized data, typical values of k often selected are 3, 5 or 10 [187].

There are several standard performance metrics that are used to evaluate a prediction performance of a model. The types of metrics to use in a particular application depend on the business goals behind the deployment of a prediction model and the distribution of data [183]. Below are some of the common performance metrics [183, 190].

- *Accuracy*. This is a baseline metric that measures a ratio of number of correct predictions given the total number of predictions made. Mathematically, accuracy is described with the expression below;

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad 1$$

Where:

TP – True Positive: an outcome correctly predicted as a positive class.

TN - True Negative: an outcome correctly predicted as a negative class.

FP – False Positive: an outcome incorrectly predicted as a positive class.

FN – False Negative: an outcome incorrectly predicted as a negative class.

- *False Positive Rate (FPR)*. FPR is the proportion of samples of negative class incorrectly predicted as positive class. FPR is expressed by the formula;

$$FPR = \frac{FP}{FP + TN} \quad 2$$

- *False Negative Rate (FNR)*. FNR is the proportion of samples of positive class incorrectly predicted as negative class. FNR is expressed by the formula;

$$FNR = \frac{FN}{FN + TP} \quad 3$$

- *Precision.* Precision is a metric that measures the model's exactness, that is, how many samples which were predicted as positive are actually positive. The metric is useful when the business goal is to limit the number of false positives. It is expressed by the following formula;

$$Precision = \frac{TP}{TP + FP} \quad 4$$

- *Recall.* Recall is a measure of the model's completeness, that is, how many positive samples are captured by the positive predictions. The metric is important when the goal is to avoid false negatives. Below is its mathematical expression;

$$Recall = \frac{TP}{TP + FN} \quad 5$$

- *F1-score.* The metric combines precision and recall metrics into one value. It is a useful metric when dealing with imbalanced data or when we desire least possible false positives and negatives. It is computed as follows;

$$F1 - score = \frac{2 * Precision * Recall}{Precision + Recall} \quad 6$$

- *Area Under ROC Curve (ROC).* Receiver Operating Characteristics (ROC) curve is a probability curve that demonstrates the ability of a model in separating two classes at various possible threshold values [191, 192]. The curve is a plot of True Positive Rate (TPR) against False Positive Rate (FPR) values for each of the model's threshold value ranging from 0.0 to 1.0, as shown in Figure 3.5. TPR and FPR are defined as recall and (1 – precision) respectively. The more accurate the model is the closer the curve is towards the TPR axis thus the larger the Area Under Curve (AUC). AUC, therefore, is also useful in determining the prediction strength of a model. For a perfect model, the value of AUC is 1, meaning it's probability to separate the classes is 100%. For the worst model, the value of AUC is 0.

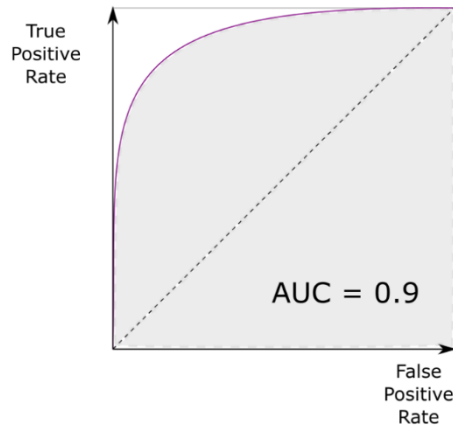


Figure 3.5. ROC curve of a model showing its area under a curve in a shaded part.

3.4.6. Hyperparameter Tuning

This is a process of evaluating a set of values of hyperparameters of a model in order to identify those which yield an optimal prediction performance. Hyperparameters are configuration arguments specified by the developer to guide the learning process of an ML algorithm for a specific data [193]. Tuning can be performed manually, in which the selected values of each hyperparameter are evaluated one by one by the developer and then their results are compared. Another approach is the use of automated techniques in which a combination of values of different hyperparameters are evaluated by an algorithm and the ones with the highest performance are identified. An example of a technique preferred for this task is Random Search [194]. In this technique, a set of values for each hyperparameter is identified by the developer and each value in every set is combined with each value in other sets of hyperparameters to form a grid of hyperparameter search space. From the resulting grid, random combinations are selected from which each one is algorithmically evaluated and the one yielding the best performance is identified as the best hyperparameters for the model [194].

3.4.7. Prediction Model Deployment

After passing the success metrics of the business goals, the model is integrated into the existing applications in order to be used with live data. At first, trial deployments should be executed to assess the model's performance on live data and detect any errors before a production deployment is staged [195].

3.5. Machine Learning Algorithms for Classification Tasks

In this section, we describe the structures, assumptions on the data distribution and the learning process of some of the ML algorithms for classification tasks used in this thesis. The first section describes traditional ML algorithms and the second one describes DL algorithms. DL is an advanced form of ML algorithms that extends the number of computational layers of Artificial Neural Network (one of the traditional ML algorithm described below) to extract minimum but most relevant predictive features from the raw input [196].

3.5.1. Traditional Machine Learning Algorithms for Classification Tasks

3.5.1.1. Logistic Regression

Logistic Regression (LR) is an algorithm which uses a statistical linear function to create a decision boundary for classifying instances to their respective classes (see Figure 3.6). It is used to predict a binary outcome based on a set of independent variables. The algorithm assumes a linear relationship between the input variables and the output variable. The relationship is represented by a linear function with each variable having a coefficient value learnt from the data by the algorithm. The function computes a probability score as a sum of all weighted input variables to measure the chance that an instance belongs to a particular class [197]. Using the set threshold, the score is estimated to the nearest numeric number (for example 0 or 1 in a binary classification task) representing the predicted class. The score above the threshold is estimated to 1 while that below the threshold is estimated to 0 [183, 197]. The linear function is defined by the following expression [183];

$$\tilde{y} = w[0] * x[0] + w[1] * x[1] + \dots + w[p] * x[p] + b \quad 7$$

Where p is the number of variables of a single data instance, w and b are coefficient and intercept respectively and \hat{y} returns the predicted class (from the estimated probability score).

The algorithm is popular because it is very fast to train and interpretable. It performs well on large sized data with no or little correlations between the variables and there is linearity between variables and an outcome. However, it performs poorly on small sized data, data with high correlations between variables and data with weak or no linearity between variables and the outcome [183, 198].

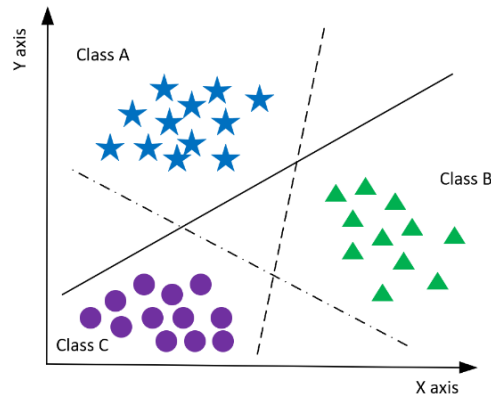


Figure 3.6. An example of linear decision boundaries created by a linear function for separating instances of three classes.

3.5.1.2. Support Vector Machines

Support Vector Machines (SVMs) are algorithms which use linear hyperplanes to form decision boundaries for separating instances based on their classes. For data with instances which cannot be separated by linear hyperplanes, the algorithm uses a process known as kernel trick to reshape the data into a form that instances can be linearly separated. The reshaping is made possible by an SVM function (kernel) transforming low dimensional input space into a higher dimensional space through specific polynomial-based computations. The algorithm then generates possible hyperplanes to separate the instances. To find the hyperplane which can best separate the instances with the smallest misclassification error, the algorithm uses a small number of instances of both classes closest to the hyperplanes (also known as support vectors) to determine the hyperplane with the largest distance margins from the vectors (see Figure 3.7) [183, 199, 200]. The distances are measured using the Gaussian kernel expression [183];

$$k_{rbf}(x_1 - x_2) = \exp(-\gamma \|x_1 - x_2\|^2) \quad 8$$

Where x_1 and x_2 are data instances, $\|x_1 - x_2\|$ denotes Euclidean distance and γ is a parameter that controls the width of the Gaussian kernel.

A class of a new instance is determined by the class of the side of the decision boundary the instance is located. Because SVMs are affected by few instances (support vectors), they are memory efficient and work very well with data having both small and large number of features,

even in cases where the number of features is larger than the number of samples. The disadvantage of SVMs is that they compute probability estimates using a computationally expensive five-fold cross-validation technique, making them not ideal for large data. They are also sensitive to data scale therefore careful data pre-processing is required. They do not perform very well in data which has many noises, that is, target classes are overlapping [200].

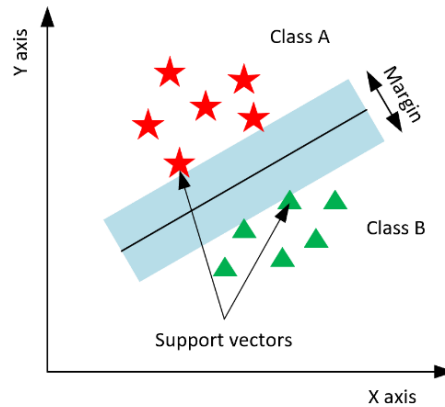


Figure 3.7. An SVM classifier using support vectors to determine a decision boundary for a binary classification task.

3.5.1.3. Decision Tree

Decision trees (DTs) are algorithms which identify ways to split data based on different features-based conditions to predict the outcome of an instance. The prediction process follows a hierarchical (a tree-like structure) approach in which various decisions based on important features are generated at each node at different hierarchy levels, from the root to leaf. Node is a point in a tree in which one feature best splits data into subsets based on the condition being tested (threshold-based or Boolean condition). Leaf is a data subset that is inseparable by any feature, with its majority class representing a prediction outcome. The algorithms follow the following process to make a prediction [201, 202];

1. It begins with an original dataset at the first node (root). The algorithm determines one feature and its threshold-based condition that can best split the dataset into different classes, that is, each resulting data subset will have the largest number of instances of one unique class. In order to do that, the algorithm computes Information Gain (IG) of each conditional feature on the dataset at the node. The feature with the highest IG is regarded as the best splitter and is selected at the node to split the dataset.

2. For each data subset obtained in the previous step, the algorithm determines another conditional feature from the set of unused features, using the same technique, that can best split the subset into other data subsets. This process is repeated in the subsequent data subsets until a data subset that can longer be split is found. At this point, a class consisting of the majority of samples in the data subset is determined as an outcome of the prediction.

To obtain IG at a given node, an entropy of a dataset at the node is computed. Entropy, in this case, measures the randomness in the distribution of number of instances of each class in the dataset. The entropy at the current state (S) is computed as [201];

$$Entropy(S) = \sum_{x=1}^c P(x) \log_2 \left(\frac{1}{P(x)} \right) \quad 9$$

Where $P(x)$ is the probability of each class in the dataset at the node. IG is the measurement of changes in entropy after the splitting of a dataset based on a feature. It measures how much information a feature provides us about a class. IG computes the difference between entropy before split and average entropy after the split by a feature. A large value of IG means the feature is effective in splitting the data and vice versa. IG is computed as [201];

$$IG = Entropy (before) - \sum_{j=1}^k Entropy (j, after) \quad 10$$

Where *before* is the dataset before the split, k is the number of data subsets generated by the split and $(j, after)$ is the subset j after the split.

The advantage of DTs is that they are easy to understand as their decision-making process is the same as the human one. Also, they can handle both categorical and numerical data, are resistant to outliers and are invariant to scaling, thus less data pre-processing is required. On the hand, they are prone to overfitting and therefore require proper tuning. Also, they can create biased learned trees if some classes dominate others [183, 203].

3.5.1.4. Random Forest

Random Forest (RF) is a type of an ensemble learning in which multiple classical ML algorithms of the same type are combined in a parallel manner to optimize prediction performances of the individual algorithms. The common building algorithm for RF is a Decision Tree. The idea behind RF, in this case, is to reduce prediction errors of the individual DTs (due to overfitting) by combining their individual predictions to obtain the most accurate and stable overall prediction result. RF uses a bagging technique to build the trees. In this technique, given a specified number of DTs, the algorithm generates different data subsets from the same original dataset to match with the number of DTs. Each equally sized unique subset consists of samples which are randomly selected from the main dataset, in addition to the replacement samples. Each tree is then fed with the unique data subset for learning. For each tree at each node, a random subset of unused features is selected from which the feature that best split the node is determined and applied at the node (as described in section 3.5.1.3).

In contrast to DTs, RFs allow trees to build all their branches to leaf nodes without pruning. With this way, along with the use of randomized training data subset and subset of features for splitting data at each node, the RFs ensure that each tree is unique thus yielding a different prediction result. In determining the overall result, the class which is predicted by most trees is voted as the overall predicted class [204, 205]. RFs offer the same advantages as those of DTs. In addition, they perform well on large data and prevent the overfitting issue which is common in DTs. On the flip side, they do not perform well on high dimensional sparse data such as text data. They are also slow in training and prediction when working on large data [183].

3.5.1.5. Artificial Neural Network

Artificial Neural Networks (ANNs) are algorithms consisting of nodes (neurons), as a processing unit, which are structured in interconnected layers to learn data for predicting an outcome. Typically, an ANN consists of three layers namely input, hidden and output layers. Input and hidden layers consist of multiple neurons whereas an output layer consists of one or multiple layers depending on the number of classes needs to be predicted. The input layer feeds an input data into the network. Hidden layer adds a layer of data learning process while the output layer produces the prediction outcome. The neurons between layers are fully connected, that is, each neuron in one layer is connected to each neuron in the next layer.

A neuron is where the learning computations happen. It combines data from each of the input variable with a set of coefficients (weights), generated by an algorithm, to either amplify or dampen the input in order to assign significance of the variables to a prediction task. Using a linear function, these input-weight products are summed and the sum is passed through an activation function. The function applies a non-linear computation to produce a neuron's output. The function uses a threshold to estimate the output which determines whether the signal should progress further through the algorithm's network to affect the ultimate outcome. The output of each neuron from one layer is passed to each neuron in the subsequent layer as an input. The outputs of all neurons of the layer prior to the output layer are combined using a linear function to produce the final outcome [183, 206, 207]. The outcome is produced as a probability of an instance belonging to a particular class. Using a set threshold, the probability is estimated and mapped to a class label as the predicted class. Figure 3.8 demonstrates the structure of a neuron.

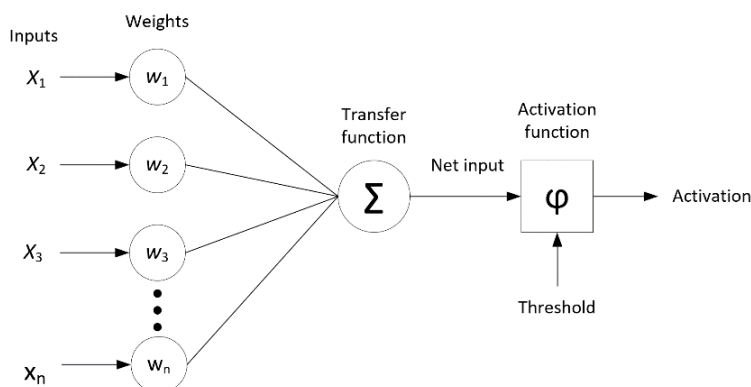


Figure 3.8. The structure of a neuron.

During the training process, the predicted outcomes are compared with the actual (labelled) outcomes of the training samples. Given the difference between the two outcomes (prediction error), the algorithm uses its back propagation function to adjust its initial randomly assigned weights and then repeats the same learning process from the first layer to the last one with the new weights (forward propagation). The weights affect the outcome of each neuron, thus the predicted outcome. The learning process is iterated until a minimum error is attained, thus determining the most accurate weights for the prediction [206]. Figure 3.9 illustrates the

structure of an ANN. Using this ANN as an example, the predicted outcome is computed as follows [183];

$$h[0] = \tanh (w[0,0] * x[0] + w[1,0] * x[1] + w[2,0] * x[2] + w[3,0] * x[3]) \quad 11$$

$$h[1] = \tanh (w[0,0] * x[0] + w[1,0] * x[1] + w[2,0] * x[2] + w[3,0] * x[3]) \quad 12$$

$$h[2] = \tanh (w[0,0] * x[0] + w[1,0] * x[1] + w[2,0] * x[2] + w[3,0] * x[3]) \quad 13$$

$$y = v[0] * h[0] + v[1] * h[1] + v[2] * h[2] \quad 14$$

Where w are the weights between the input x and the hidden layer, $\tanh()$ is an activation function, h and v are the weights between the hidden layer h and the output \hat{y} . The weights v and w are learned from data, x are the input features, y is the computed output and h is an intermediate computation.

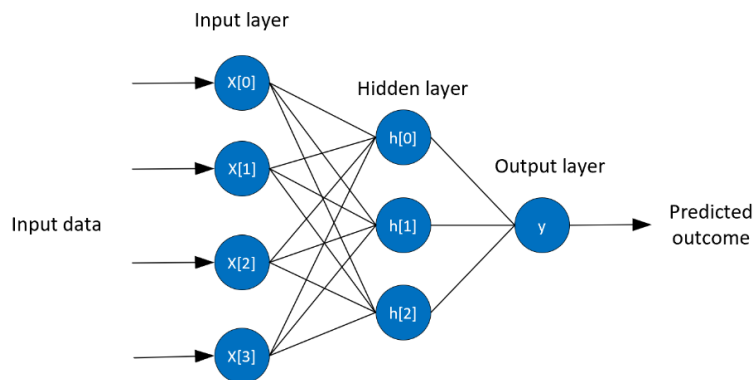


Figure 3.9. The network structure of a simple ANN with three input variables and three neurons of a hidden layer.

The advantage of ANNs over other traditional ML algorithms is that they perform better on complex and unstructured data such as images, audio and text, thus saving time for structuring the data. By regulating weights internally, the algorithms perform automatic feature extraction and selection of the most relevant features, saving time and efforts for the manual tasks as in the case of other algorithms [147, 148]. Nevertheless, ANNs have some limitations. One of them is that they require a large amount of data to extract relevant features in order to produce good accuracy. This also make them computing intensive. In some problems, data may be inadequate and therefore ANNs may not be useful [207].

3.5.2. Deep Learning Algorithms for Classification Tasks

3.5.2.1. Fully Connected Feedforward Deep Neural Networks (FC-DNNs)

FC-DNN extends the basic architecture of ANN (described in section 3.5.1.6) by adding more hidden layers between the input and output layers, resulting in the term “deep” neural network [196, 208]. Input variables are connected to multiple hidden layers and an output layer at the end. The neurons between the layers are fully connected. Computations in neurons, forward propagation, estimation of an outcome by the output layer and back propagation are performed similar to ANNs. With a larger number of hidden layers compared with ANNs, FC-DNNs perform better in large data than ANNs. These properties allow them to be more effective in learning complex data such as images and texts. However, the increase in the number of hidden layers make them more computing intensive and time consuming in both training and prediction.

3.5.2.2. Long Short-Term Memory (LSTM)

LSTM is a type of Recurrent Neural Network (RNN) which extends the implementation of FC-DNN to make a prediction from an order dependence sequential data. RNNs have a similar network structure to that of FC-DNNs, the main difference is that in RNNs, memory cells replace neurons in hidden layers. In RNN, the prediction is performed as a result of combining a current instance with the previous instances stored in memory cells of hidden layers [149]. LSTM overcomes RNN’s limitation of storing a small number of previous instances for prediction.

In LSTM, hidden memory cells are modified by implementing gates which combines current and previous instances, using an activation function, to control what relevant information to keep and pass to the next cells for prediction. By eliminating insignificant information, LSTMs are able to memorize a larger amount of previous significant information for learning, thus can predict longer sequences of instances than RNNs [209, 210]. The output of each cell is forwarded to memory cells in the next hidden layer where similar operations are repeated, thus shaping more the relevant information for prediction. The outputs of memory cells in the last hidden layer are combined and translated into a prediction result by one or multiple fully connected layers of neurons positioned after the hidden layers. Similar to ANNs, a back

propagation method is applied in multiple iterations to minimize the prediction error by regulating weights on the input data.

Figure 3.10 below illustrates the structure of a memory cell in the hidden layer of LSTM in which $c(t-1)$, $h(t-1)$ and $x(t)$ are the previous cell state, previous hidden state and input vector respectively. The f_t , i_t and O_t are the cell's control gates namely forget, input and output gates respectively. The forget gate combines which $c(t-1)$ and $h(t-1)$ through a sigmoid function to decide which relevant information from previous instances to keep. Given W and b as the linear coefficients generated by the algorithm, the output of the gate is computed with the following expression;

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f) \quad 15$$

The input gate combines $h(t-1)$ and $x(t)$ using the sigmoid function to generate relevant information to update the current state of the cell. Below are the expressions to update the state;

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i) \quad 16$$

$$\tilde{c}_t = \tanh(W_c * [h_{t-1}, x_t] + b_c) \quad 17$$

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t \quad 18$$

Using sigmoid and tanh functions, the output gate combines $h(t-1)$, $x(t)$ and the computed current cell state $c(t)$ to produce the current hidden state $h(t)$ which is the information used for prediction. Both $c(t)$ and $h(t)$ are forwarded to the memory cells of the next hidden layer [210-212]. Below are the expressions to compute the output;

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o) \quad 19$$

$$h_t = o_t * \tanh(c_t) \quad 20$$

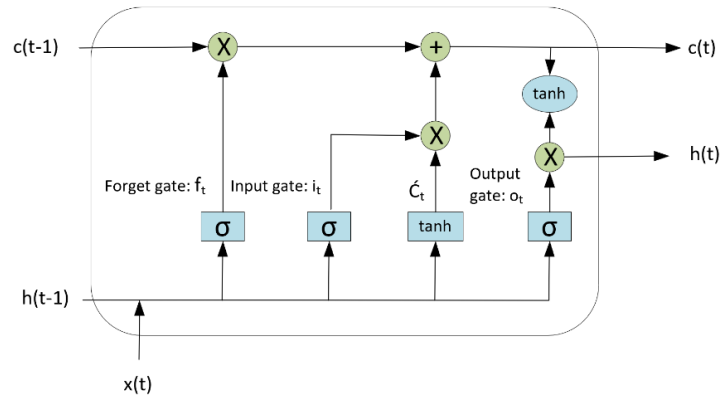


Figure 3.10. The structure of a memory cell of LSTM.

3.5.2.3. One Dimension Convolutional Neural Network (1D CNN)

Convolutional Neural Network (CNN) is the modification of FC-DNN in which each fully connected hidden layer is replaced with another layer known as convolution. The layer often consists of convolution and pooling sublayers. The layer is the main building block of CNNs in which all main computations are performed. In contrast to popular 2D and 3D CNNs, which process 2- and 3-dimensional input data respectively, 1D CNNs processes one dimensional data such as time series data [213]. In 1D CNNs, the convolution layer mainly transforms the one-dimensional input features, using multiple filters, into a mapped form of data, referred as feature map. This process is performed as follows; a specified small window of component known as filter, which is set to be shorter in height than a string of input features slides over the input data at different locations to extract a set of all feature values at each location. The element-wise matrix multiplication is then applied to each set and the results are summed to generate one value. A non-linear function is applied to the sums and the results go into the feature map. The feature map is then forwarded to the pooling sublayer [214].

The pooling sublayer selects a small number of informative mapped features and forward them to another hidden layer for the repeated task. The selection of features is performed by either max or average pooling methods. By so doing, the pooling sublayer reduces the size of feature map by picking and forwarding information of the most predictive features to the next convolution layer [149]. The process is repeated in the next convolution layers resulting in the minimum but most important information at the end of all hidden layers. The output of convolution layers is then fed into one or multiple fully connected layers of neurons, which, in

turn, compute and output a prediction result, as described in the ANNs. Similar to ANNs, the backpropagation method is also applied for several iterations to minimize the prediction error. In each convolution layer, 1D forward propagation is mathematically expressed as [215];

$$x_k^l = b_k^l + \sum_{i=1}^{N_{l-1}} Conv1D (w_{ik}^{l-1} - s_1^{l-1}) \quad 21$$

Where x_k^l is defined as the input, b_k^l is defined as the bias of the k^{th} neuron at layer l , s_1^{l-1} is the output of the i^{th} neuron at layer $l-1$ and w_{ik}^{l-1} is the kernel from the i^{th} neuron at layer $l-1$ to the k^{th} neuron at layer l . $conv1D(\dots)$ performs a 1D convolution. The dimension of the input array, x_k^l is less than the dimension of the output arrays, s_1^{l-1} .

Unlike FC-DNNs, the convolution process significantly reduces dimensionality of features, making 1D CNNs more accurate, compact and lesser computing intensive [149]. Also, due to this process, they manage to perform well in spatial data compared to other DL algorithms [148, 213]. Figure 3.11 below illustrates the structure of a simple 1D CNN.

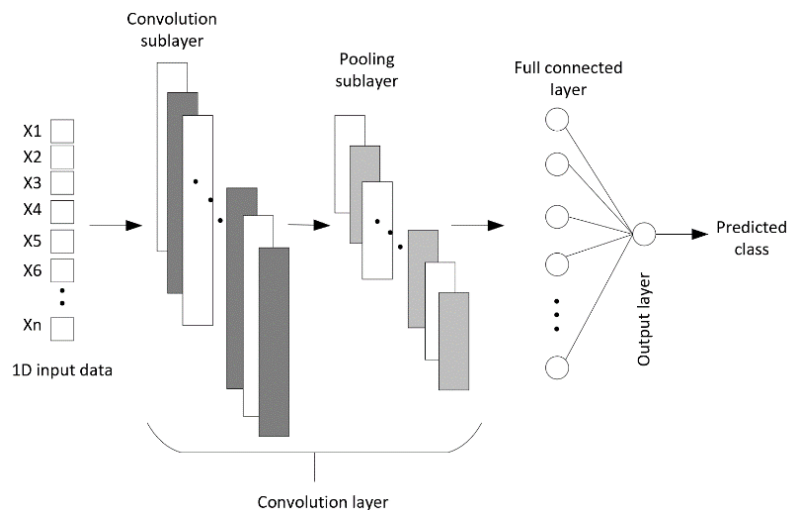


Figure 3.11. The network structure of 1D CNN with one convolution layer and one fully connected layer of neuron.

3.6.Summary

In this chapter, we have provided an overview of ML including the learning algorithms commonly used in the cybersecurity domain. First, ML and its three major types have been

described. Then, we shed light on ML for classification tasks and its main categories (binary and multi-class classifications), which are the most popular techniques for addressing cybersecurity-related problems including the one that is studied in this thesis. The hierarchical classification method has been explained in more detail as it has been utilised in this thesis to address multi-class classification problems. We have also described a standard workflow that guides the design and implementation of ML prediction models. Lastly, we have described ML algorithms for classification tasks with a particular focus on the five traditional ML and three DL algorithms.

In this chapter, we have learned that ML has been successfully used to address various cybersecurity problems. Hierarchical classification is an effective and informative alternative to flat classification for addressing multi-class classification problems. The performance of ML algorithms depends on the quantity and quality of data. The algorithms generally perform well on large datasets. The quality of raw real-world data is often not suitable for the algorithms to produce good prediction results. To improve the quality, various data pre-processing tasks need to be performed. The algorithms also vary in performance depending on the type and distribution of data. Given data on a specific problem, an appropriate algorithm that fits the data type and distribution should be selected in order to achieve the best performance.

Given the success of ML in addressing the cybersecurity related problems, we will use the approach to address the problem defined in this research. Since our input data is labelled, classification-based ML algorithms will be used for this task. As the performance of the algorithms depends on the nature of data, this research will evaluate the data using a number of common classification algorithms to determine the algorithm that can learn best the data, thus, achieve the best prediction performance.

Chapter 4

Detection of Zero-Day Phishing Webpages Using Machine Learning

4.1. Introduction

In the previous chapters, we have seen that attackers are increasingly using sophisticated phishing toolkits to create a large number of high-quality phishing websites within a short period, resulting in an increase in the number of zero-day phishing websites. In addition, mechanisms are incorporated into the websites to allow them to evade various existing detection solutions. The toolkits are updated frequently with new evasion mechanisms as their authors continue to discover features used by existing detection techniques. As a result, over time, the detection solutions are likely to become less effective in protecting users from phishing. The sustained rise in the number of phishing websites in recent years despite phishing website detection solutions being deployed by users and organizations suggests that the solutions are not entirely effective. To keep pace with phishers, solutions that will use novel set of features to provide real time and accurate detection of zero-day phishing websites are required. The solutions are also required to be more resistant to detection evasion tactics.

Given sufficient data with relevant features, ML has demonstrated the ability to solve complex prediction problems in various domains, including cybersecurity, in real time and with high accuracy. The review of phishing websites in Chapter 2 revealed that phishing PDC webpages exhibit some distinctive structural characteristics that can be useful in distinguishing them from the legitimate ones. Therefore, to counter phishing website attacks, we leverage the ML approach to detect zero-day phishing PDC webpages using features based on these characteristics. The role of ML in this case is to provide fast and highly accurate prediction of phishing webpages in order to ensure that they are detected as soon as they become accessible to users. The proposed features, most of which are novel, are taken from five categories, thus providing a level of feature diversification which is likely to make it difficult for phishers to subvert most or all the features in order to evade the entire solution. The significance of using novel features is that it will take time for phishers to discover them and develop their evasion mechanisms, increasing the chance of the solution to be effective longer than the current ones.

Part of the work in this chapter was published in a paper titled “A Framework of New Hybrid Features for Intelligent Detection of Zero Hour Phishing Websites” which was presented at the 12th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2019) in Seville (Spain) in 13th of May 2019. The paper can be accessed through doi: 10.1007/978-3-030-20005- 3_4. The paper proposes 20 features, based on URL and webpage structure, for accurate and instant prediction of zero-hour (day) phishing websites using a supervised ML approach. The features were evaluated using eight different traditional ML algorithms. The work in this chapter extends the paper by proposing 26 features based on URL and webpage structure. The features are evaluated using eight and three traditional and deep learning supervised algorithms respectively.

This chapter is divided into six sections. In section 4.2, we review related works and discuss their limitations. Section 4.4 we describe our proposed method for identifying PDC webpages, the proposed prediction features for the detection of phishing PDC webpages and the system architecture of the prediction model. Section 4.5 describes the experiments conducted to evaluate the features, performance results and our analysis on the results. In section 4.6, we compare our work with other related works and identify possible ways the proposed model can be applied to protect users from the attacks. Lastly, section 4.7 summarises the chapter.

4.2. Related Works

Current approaches for detecting phishing webpages can be grouped into five categories. These are blacklists and whitelists, visual similarity, offensive defence, rule based and hybrid.

4.2.1. Blacklist and Whitelist Based Approach

In this category, several web browser filters and anti-malware suites are available to protect users from accessing phishing PDC webpages. The web browser filters are incorporated in web browsers as a built-in or installable component (also known as plug-in). The anti-malware suites, on the other hand, are software that can either be installed in a user’s machines as standalone applications or as client applications of cloud-based software. The suites scan the websites as they are being accessed by users to detect malicious behaviours. Most filters and the suites are based on a URL blacklist approach. Google’s Safe Browsing, for instance, uses a database of hashed URLs of malicious webpages, including phishing ones, to protect users

of Chrome, Firefox, Safari and Opera web browsers. It warns users when they access phishing webpages whose URLs are in the list [216-218]. TrustWatch is a web browser plug-in that employs a blacklist of phishing URLs, in combination with validity check of the website's TLS certificate, to indicate to a user the level of suspicion of a webpage for phishing using three coloured indicators [219]. Other popular plug-ins using blacklists are Bitdefender TrafficLight and PhishTank SiteChecker [220, 221].

Similarly, anti-malware suites including Trend Micro, ESET and Kaspersky employ their own blacklists of phishing URLs to detect phishing webpages being accessed by users [222-224]. At the research level, Dong, et al. [225] proposed a detection system based on two whitelists; one consisting of websites the user has already visited, and the other consisting of domain names of the websites the user has accounts with and their corresponding credentials. If a visited website is not in both lists, a phishing score is computed for the webpage and it is flagged as phishing if the score is above the set threshold. The system achieved an accuracy of 99.14% and an FPR of 0.14%.

4.2.2. Visual Similarity Approach

This approach compares images of suspicious webpages or in combination with webpage structure and contents against those of pre-collected legitimate webpages to detect phishing webpages based on the computed visual similarity scores. Medvet, et al. [42], for instance, proposed a visual similarity-based technique which uses webpage text, individual images within a webpage and the entire webpage image (screenshot) to compare a suspicious webpage against a list of texts and images of pre-collected legitimate webpages. The webpage is flagged as phishing if its overall similarity score of the three features against the score of any of the legitimate webpages in the list exceeds the pre-computed threshold score. The technique attained an FPR of 0% and an FNR of 7.4%. Hara, et al. [226] proposed a technique in which a domain name and image of a webpage (screenshot) is compared against a database of domain names and images of pre-collected legitimate and phishing webpages. A new webpage is flagged as a phishing webpage if its visual similarity score against one of the webpages in the database is higher than the pre-calculated threshold and their domain names are not matching. Phishing webpages were used in the database to help detecting other phishing webpages

targeting the same legitimate webpage as they are likely to be similar in their visual looks. The technique achieved an accuracy of 80% and FPR of 17.5%.

A work by Chen, et al. [44] proposed a technique based on a similarity check of key image features extracted from all images of a potential phishing webpage against those from a list of the most targeted legitimate webpages. The study adopted a Contrast Context Histogram (CCH) technique to compute a similarity degree between the tested webpage and each webpage in the list. If the degree is above the set threshold, the webpage is flagged as phishing otherwise it is not. The work achieved an accuracy ranging between 95% and 98%, and FPR and FNR ranging between 0.0% and 2.1% when tested against phishing webpages targeting different legitimate webpages. Zhang, et al. [43]'s study proposed a technique which combines comparison of texts and images of a webpage against those of the common targeted legitimate webpages. The text and image classifiers use Bayesian probabilities to determine the classification thresholds. The results of the two classifiers are combined, using a Bayesian fusion approach, to generate the overall classification result. An accuracy ranging between 99.68% and 100%, and FNR ranging between 0% and 1.96% were obtained when tested against phishing webpages targeting different legitimate webpages.

4.2.3. Offense Defensive Approach

In this approach, fake login credentials are fed into a suspected phishing webpage and a response from the webpage is compared against an expected response from the corresponding legitimate webpage. A dissimilar response indicates that the webpage is a phishing one. Knickerbocker, et al. [45]'s system, for instance, generates a list of fake credentials for each of the suspected phishing webpages and send the credentials to the webpages as if they are coming from the phishing victims. The system then monitors the targeted legitimate webpages over a certain period to count the number of fake credentials from the list that have been used to attempt to login to the targets. A probability of the used fake credentials is then computed by the system. If the probability exceeds the set threshold, the system flags the website to which the fake credentials were submitted to as a phishing one. Shahriar and Zulkernine [46] proposed a system which detects a phishing website by checking its response after being provided with random data input and comparing it with the expected behaviours of the pre-collected phishing and legitimate websites. The detection outcome is determined by eight heuristic rules based on

the state sequence of inputs, their correspondences as well as contents of the responses. The system achieved 0 FPR and 0 FNR.

4.2.4. Rule Based Approach

This is one of the most popular approaches in the research works because of its capability to instantly predict both known and unknown phishing webpages with high accuracy rates. This category uses rules that are set manually or determined automatically by ML algorithms to distinguish phishing webpages from the legitimate ones. Aburrous, et al. [227], for instance, proposed a classifier based on a fuzzy data mining algorithm to detect phishing webpages with an accuracy of 86.3% and FPR of 13.6%. The classifier used 28 features based on webpage contents, WHOIS domain records, URL structure and social human factors. Ma, et al. [120] used an SVM based classifier with 16 heuristic features related to WHOIS domain records, URL structure, network and geolocation, and IP address and domain name checks against seven blacklists and one whitelist. The lists used included SORBS, URIBL, SURBL, Spamhaus and a list from SpamAssassin botnet plugin. The classifier yielded classification error rates ranging between 0.90% to 44.02% when different training and testing datasets collected from different sources were used. Xiang, et al. [118] developed a CANTINA+ system, based on a Bayesian Network classifier, to detect phishing webpages. The classifier used 15 features related to webpage contents, WHOIS domain records, URL structure and search engine reputation to detect phishing webpages to achieve an accuracy, FPR and F1 score of 92.25%, 1.375% and 0.95 respectively. Lakshmi and Vijaya [58]'s work proposed a Decision Tree-based classifier which combined 14 webpage contents, URL structure, WHOIS domain records and search engine reputation related features, and a URL check against a blacklist of phishing URLs. An accuracy of 98.5% and classification error rate of 0.15% were achieved by the classifier. Mohammad, et al. [228] proposed a self-structuring DL classifier to detect phishing webpages. The classifier was constructed using 17 features related to URL structure, webpage structure and contents and WHOIS domain records. An accuracy of 92.18% was achieved by the classifier. Zuhair, et al. [47] developed an SVM based classifier to detect phishing webpages. The classifier used 48 webpage structure and 10 URL based features to yield an FPR of 1.17% and FNR of 0.03%.

Shirazi H. [48]’s work compared several DL algorithms and SVMs to detect phishing webpages. Using 30 features related to URL structure, webpage structure, WHOIS domain records, Alexa’s webpage reputation and Google’s search engine reputation, one of the DL algorithms achieved the best results with Area Under ROC Curve (AUC) of 0.897, True Negative Rate (TNR) of 90.27%, and True Positive Rate (TPR) of 89.33%. An ensemble classifier of Linear Combination and Extreme Learning Machine algorithms was proposed by Zhang, et al. [229] to detect phishing webpages based on 12 features of URL structure, webpage structure and contents, WHOIS domain records and Alexa’s website reputation. An accuracy of 99.04% and FPR of 0.53% were obtained for English based webpages whereas an accuracy of 97.5%, FPR of 2% and FNR of 3% were achieved for Chinese based webpages. Jain and Gupta [49]’s Random Forest based classifier detected phishing webpages with an accuracy of 99.09% and FPR of 1.25%. A total of 19 features based on URL structure, and webpage structure and contents were used by the classifier. A Random Forest based classifier proposed by Sahingoz, et al. [50] used word vectors, natural language processing of URL characters as well as website’s ranking in the Alexa top websites list to detect phishing webpages. An accuracy, precision, sensitivity and f-measure of 97.97%, 0.97, 0.99 and 0.98 respectively were achieved by the classifier. Li, et al. [51] proposed a classifier composed from a stack of GB, Extreme GB (XGBoost) and Light GB (LightGBM) algorithms to detect phishing webpages. The classifiers used 20 features based on URL structure, and webpage structure and contents to yield an accuracy of 97.3%, FPR of 4.46% and FNR of 1.61%.

4.2.5. Hybrid Approach

This approach involves combinations of more than one approach described above. For instance, Gowtham and Krishnamurthi [230] proposed a phishing webpage detection system which combined a pre-filtering whitelist maintained by a user and an SVM based classifier. The classifier was built using webpage contents, WHOIS domain records, URL structure and Google’s PageRanking reputation related features. The classifier achieved TPR of 99.65%, FPR of 0.42% and F1 score of 99.7%. Table in appendix 1 summarizes significant works of all the mentioned categories.

4.3. Limitations of Related Works

The techniques described above have various limitations as explained below;

1. The blacklists often depend on users reporting phishing webpages in order to build their records. Websites could already have been alive for several hours, days or weeks by the time they are seen and reported. Once reported, managers of the blacklists usually validate the reports manually before confirming and recording the websites in the database. This process takes time, leading to delays in updating the lists. The blacklists, therefore, are less effective in instantly detecting zero-day phishing webpages. This was also demonstrated by various studies. Barraclough, et al. [38], for instance, observed that 83% of phishing websites take at least 12 hours to appear in a blacklist after they were launched. Wenyin, et al. [37]'s experiment showed that PhishTank identified 89.2% of phishing websites within an average of 16.4 hours per each website, Google's Safe Browsing identified 65.7% of the websites in 10 hours on average and Microsoft's web browser detection identified 40.4% of the websites in 24.5 hours on average. In addition, AV-Comparatives [231] tested all major consumer anti-malware suites with anti-phishing capabilities against newly discovered phishing webpages. They found out that only three of them had an accuracy of above 90%, the highest being 94%.

The proposed whitelists require prior collection of records of specific legitimate websites that each user has visited or intends to visit regularly. First, the process involves intensive manual work of collecting details of many such websites. Second, the technique protects users of the specified legitimate websites only against their corresponding phishing websites. False positive alarms will be raised for new legitimate websites that have not yet been recorded in the lists. Third, whitelists which are based on saving user credentials of legitimate websites are not ideal for websites whose credentials change from a one log in process to another such as those using one time passwords.

2. As observed by Medvet, et al. [42], visual similarity methods are computational intensive and therefore are expected to have longer detection times compared to other approaches. Given that a large number of legitimate websites needs to be compared against every time a user attempts to access an PDC webpage, the approach is likely to create intolerable detection overheads. In addition, the technique provides a limited protection, that is, against phishing websites of the legitimate websites which have been recorded in the database.

3. Similar to the whitelists, offensive defence techniques detect only specific phishing websites whose target legitimate websites have been recorded in the database. Those whose target legitimate websites have not been recorded will not be detected.
4. Most of the rule-based approaches achieved low or moderate prediction performances with respect to various metrics. For instance, Aburrous, et al. [227] achieved an accuracy of 86.3%, Whittaker, et al. [232]'s accuracy was 90% and Shirazi H. [48]'s AUC was 0.897.
5. Most of the rule-based approaches used sets of features with low diversity. For instance, Pan and Ding [233] used features from 2 features categories, He, et al. [234] used features from 2 feature categories, Jain and Gupta [49]'s features are of 2 different categories and Li, et al. [51]'s features are from 2 feature categories. Phishers tend to learn features of the detection techniques to find effective ways to bypass them [27]. With a small set of feature categories, attackers can learn most or all the features belonging to each category and require one or a few evasion mechanisms to bypass each category, thus the entire solution. For instance, most or all the webpage content-based features can simply be neutralized by creating a phishing webpage as an exact duplicate of the target legitimate webpage. In this case, approaches such as Zuhair, et al. [47], Jain and Gupta [49] and Li, et al. [51], which are based mainly on webpage content features, will produce high FNRs, allowing a significant number of phishing webpages to go undetected.
6. A number of rule based approaches such as Li, et al. [51] and Sahingoz, et al. [50] intentionally did not use features involving third party services, such as WHOIS database, for the reason that extraction of the features would cause significant overheads, resulting in longer detection times. Instead, they used features locally extracted from the URLs and webpages. Such features, however, can easily be emulated by phishers by ensuring the phishing webpages and their URLs are as similar as their targets, thus neutralizing at least most of the distinguishing features between phishing and legitimate webpages. Development of high-quality duplicates of legitimate webpages is possible with the sophisticated phishing toolkits as described in section 2.2.3.5. Nevertheless, we argue that it is extremely difficult for phishers to manipulate third party services because the services are highly secured. To manipulate the services, the phishers would require high technical skills, longer time and other resources that very few attackers are likely to possess or willing to invest in.

From this review, we learn that blacklist-based solutions are the most common ones being deployed by users to detect phishing webpages. Despite them being accurate in detecting known phishing webpages in real time, they are less so in detecting zero-day phishing webpages. Given the significance and prevalence of zero-day phishing webpages across the globe, a different approach is needed to detect them in an effective and efficient way. The ideal approach would be the one that does not rely on the lists of known phishing websites and can accurately detect both known and unknown phishing websites in real time. The reviewed ML based solutions have suggested that the ML approach has the potential to meet these design goals if used with appropriate prediction features. However, as attackers investigate the prediction features used in existing solutions and devise corresponding evasion mechanisms, the ideal solution should also make evasion as difficult as possible and for as long as possible. This would be through, for instance, the use of a set of novel features which are also highly diversified and include features involving information from third party services. The significance of these feature characteristics in this aspect has been described in the sections above. In this study, therefore, we aim at investigating, identifying and evaluating a set of highly diversified novel features that can accurately predicts zero-day phishing webpages in real time using an ML approach.

4.4. Prediction Model Design

In this section, we describe how we derived our proposed features for predicting phishing PDC webpages and provide summarized descriptions of some of the features. We also describe and illustrate the proposed system architecture of our prediction model.

4.4.1. Phishing Webpage Prediction Features

First, we describe how we identified PDC webpages, the webpages that collects users' personal data. As explained in section 2.3.1.2, not every webpage with an HTML form or a dialogue window collects personal data. From our observations, PDC webpages usually consist of at least one word or phrase (we refer to it as PDC phrase) in their structure and contents which is related to the specific personal data being collected. To determine common PDC phrases used by PDC webpages, we investigated 100 samples of phishing and legitimate webpages capturing the data from which we obtained a list of 43 PDC phrases (indicated in Table 4.1). The importance of differentiating PDC from non-PDC webpages is that we avoid predicting

webpages which do not pose any phishing threat. This will avoid degrading of user's experience when accessing the non-PDC webpages and the potential false positives on these webpages which will prevent users from accessing them, causing significant implications to users and websites' owners (e.g denial of services and losses of revenues). The list, however, is not exhaustive as there could be other PDC webpages which capture personal data not related to the PDC phrases in the list. In this case, such webpages will be regarded by our model as non-PDC webpages, thus will not be considered for the prediction analysis. To expand the list, a larger set of PDC webpages can be used to extract the phrases. For instance, one can collect a comprehensive set of known phishing PDC webpages from various phishing blacklists and algorithmically extracts label and default values of all input fields, and name attribute of a submit button in each webpage to create the list.

Username	Login	Forgotten your password	Customer number	Log in with Facebook
User	Password	Reset password	Membership number	Log in with Twitter
Email	PIN	Debit card number	Billing information	Log in with Google
Account	Secret key	Credit card number	Billing address	Sign in with Facebook
Account number	Security code	Card number	Cardholder	Sign in with Twitter
ID	Security key	Account number	Expiry date	Sign in with Google
Sign in	Security number	Security number	Date of birth	Create an account
Sign up	Forgot password	Passcode	Birth date	
Log in	Forgot	Remember me	Phone	

Table 4.1. Common PDC phrases used in PDC webpages.

Based on the differences in the structural characteristics between phishing and legitimate websites described in section 2.3.1.3, we derived various potential features for distinguishing phishing PDC webpages from the legitimate ones. We also studied features used by previous works addressing the same problem and identified those which can be extended or improved, based on the mentioned characteristics, to add to our set of features. In addition, we adopted some of the features in our proposed set of potential features those which were used and defined

as strong predictors in several works. In order to identify the features which are strongly exhibited by the current PDC webpages, thus are likely to be useful predictors, we investigated them in the same 100 samples of phishing and legitimate PDC webpages mentioned above. This was done by manually analysing the occurrence patterns of values of each feature across the two sets of webpages. The features whose patterns of values were more consistent in one set compared to the other were considered to be potential predictors. For example, we counted the number of URLs from each set that contained the character @ in their strings (feature #10 in Table 4.2). We found that the character appeared in almost 18% of all the phishing URLs while none in the legitimate URLs. From this investigation, 35 of such features were identified. We categorize the features in five groups namely webpage structure and contents, URL structure, WHOIS records, TLS certificate and webpage reputation. The categorization is based on the similarity of sources of the features. For instance, all features which were derived from the character composition of a URL string are grouped as URL structure. Of the 35 features, 24 are new ones introduced by this study and 11 features are adopted from previous works. In this section, we describe some of the features which were observed to be among the best features for this problem. Descriptions of all the proposed features can be found in appendix II. Table 4.2 below summarizes all the proposed features. The Table indicates whether each feature is extracted locally (from the URL or webpage structure and contents) or from information obtained from an external source. It also indicates whether the features are novel or adopted from existing works.

Feature #	Feature Category	Feature	Local/ 3rd Party	Novel or adopted?
1	Webpage structure and contents	Domain identity in a webpage	Local	Novel
2		Domain identity in copyright		Novel
3		Domain in canonical URL		Novel
4		Domain in alternate URL		Novel
5		Foreign domains in links		Novel
6		Proportion of void and same webpage links		Novel
7		Foreign form handler		Novel
8	URL structure	Encoded hostname		Adopted
9		Encoded URL path		Adopted

10		Use of @ character		Adopted
11		Domain out of position		Novel
12		# of dots in hostname		Novel
13		# of dots in the URL path		Novel
14		Non-standard port number		Adopted
15		# of obfuscation characters in hostname		Novel
16		# of obfuscation characters in URL path		Novel
17		# of forward slashes		Adopted
18		# of characters in hostname		Adopted
19		# of characters in URL path		Novel
20		IP address in a hostname		Adopted
21		Numeric in a hostname		Novel
22		Numeric in a URL path		Novel
23		Shortened URLs		Adopted
24		Free domain services		Novel
25	WHOIS records	Domain validity		Novel
26		Form handler's domain validity		Adopted
27		Domain age		Adopted
28		Form handler domain's age		Adopted
29	TLS certificate	Type of TLS certificate		Novel
30		Domain, certificate and geolocation country matching		Novel
31	Webpage reputation	URL ranking in search engines	3 rd party	Novel
32		Hostname ranking in search engines		Novel
33		Domain ranking in search engines		Novel
34		Counts of matched hostname's IP address against IP addresses of blacklisted phishing websites		Novel
35		Counts of matched domain's IP address against IP addresses of blacklisted phishing websites		Novel

Table 4.2. Summary of the proposed features.

4.4.1.1. Webpage Structure and Contents

This category consists of 7 features derived from texts of various components of the webpage's HTML structure and contents. Four of the features are described below.

Feature 1: Domain Identity in a Webpage

Legitimate webpages often contain words in their structure or contents which are related to their brands. Similarly, the second level domain (for generic Top-Level Domains (gTLDs)) or the third level domain (for country code Top-Level Domains (ccTLDs)) of a URL often represents a brand of an organization owning the website. We refer the brand name appearing in the domain as *domain identity*. Phishing webpages hosted in attackers' own domains have different domains from those of the targeted legitimate websites. Their domain identities, therefore, are less likely to appear in their webpages which mimic the structures and contents of the targeted legitimate webpages. In this feature, we extract the second or third level domain from a URL as the domain identity and search it throughout the webpage to count how many times it has appeared in the webpage. It is to be expected that the lower the counts, the higher the chances that the webpage is a phishing one and vice versa.

Algorithm: Domain identity in the PDC webpage

Input: Webpage

Output: Counts of the appearances of the domain identity in the PDC webpage

Retrieve the second level domain (for gTLDs) or the third level domain (for ccTLDs) from the URL's FQDN as the domain identity.

Search for the domain identity in the webpage's HTML structure and contents.

Count the number of appearances of the identity.

Feature 5: Foreign Domains in Links

Except for links to objects of legitimate websites such as images and stylesheets, which are often hosted in external legitimate domains, our observation of sampled legitimate PDC webpages has shown that the rest of the links (those defined with *href* tags) are usually hosted in the same domain (second level domain for gTLDs or the third level domain for ccTLDs) as the URL's domain of the PDC webpage. Since most of the phishing PDC webpages maintain original links from their targeted legitimate webpages, their URLs' malicious domains are

likely to be different from most of the domains in the links. In this feature, we extract all non-object links (with *href* tags), determine the most common domain and compare it with the webpage's domain. If the two are mismatching, we flag the webpage as phishing one, otherwise we label it as legitimate.

Algorithm: Foreign domains in links

Input: Webpage

Output: 1: Phishing webpage

0: Legitimate webpage

-1: Unknown

Find all webpage links.

If links with URLs exist,

If the links are not of objects,

Identify unique domains in all links,

Count number of appearances of each domain and retrieve a domain with the highest count,

Compare webpage's domain against the domain with the highest count,

If there is a matching

Output → 0

Else

Output → 1

Else if all links have no URLs OR no links exists

Output → -1

Feature 6: Proportion of Void and Same Webpage Links

Described in section 2.3.1.2, links to different webpages and those directing to various sections of the same webpage are commonly used in legitimate webpages whereas void links (those with no URL assigned to them) are less expected. A phisher may modify links copied from a target webpage to point to the same phishing webpage or to create void links to force users to focus on the phishing webpage [233]. We therefore argue that the two types of links, especially the latter, are more likely to be common in phishing webpages than in legitimate ones. To measure the presence of the types of links in a webpage, we take a ratio of the sum of numbers of void and same-page links to the total number of non-object links. We expect that the higher the ratio, the higher the probability that the webpage is a phishing one.

Algorithm: Proportion of void and same webpage links

Input: Webpage

Output: Ratio of void and the same webpage links.

Retrieve all non-object links.

Identify and count the numbers of void and same webpage links.

Divide the sum of the numbers of void and the same webpage links by the total number of links.

Feature 7: Foreign Form Handler

In most legitimate websites, data collected by the HTML forms are often processed by form handlers (FHs) (described in section 2.3.1.2) hosted in the same website. A second or third level domain of an FH of the webpage, in this case, is likely to be the same as both domain of the webpage and the dominant domain of its non-object links. When designing phishing webpages, phishers change FHs to point to their own servers to collect the data. Domains of FHs of phishing webpages are therefore likely to be different from either the domains of the webpages or the dominant domains in non-object links (depending on whether the webpage is hosted in a compromised or attacker's registered domain).

Algorithm: Foreign form handler

Input: Webpage

Output: 1: Phishing webpage.

0: Legitimate webpage.

Retrieve the hostname.

Retrieve all webpage links except those of objects.

Identify the most dominant domain in all links.

Compare the form handler's domain to both the webpage's domain and the most dominant domain in all links.

If there is a match to both,

Output \rightarrow 0

Else

Output → 1

4.4.1.2. URL Structure

This is category of 17 features which are derived from the URL's hostname and path. In this section, we describe five features.

Features 8-9: Encoded Hostname and URL Path

To hide identification of hostnames for spoofing purposes, hostnames of some of the phishing URLs are encoded using ASCII character conversion [119, 235]. Each character is represented by % followed by two hexadecimal digits (numbers or letters or combination of both) [236]. For instance, the domain *google.com* can be represented as *%67%6f%6f%67%6c%65%2e%63%6f%6d* [235]. This format is not a standard practice in the legitimate domains. In feature 8, we flag phishing if we detect at least one set of % followed by two hexadecimal digits in the hostnames. Encoding of some of the characters in URL paths (i.e. the strings following the hostnames) is commonly observed in legitimate URLs. More characters, however, are likely to be encoded in phishing URL paths to obfuscate the paths. We establish feature 9 by counting on the number of the encoded characters in the paths. The more encoded characters are found in the path, the more likely that the URL is a phishing one.

Algorithm: Encoded hostname

Input: URL

Output: 1: Phishing webpage.

0: Legitimate webpage.

Retrieve the hostname.

If at least one set of '%' followed by two hexadecimal digits exists in the hostname

Output → 1

Else

Output → 0

Algorithm: Encoded URL path

Input: URL

Output: Counts of encoded characters in the URL path

Retrieve the URL path.

Count sets of '%' followed by two hexadecimal digits in the path

Feature 10: Use of @ Character in a URL

Phishers use the character @ or %40 (in ASCII format) in the URL to redirect users to their phishing webpages [237]. In legitimate usage, an @ is used to prefix a hostname with user information (username and optionally a password). Thus, if a browser encounters the @ in a URL, it will assume that the string before @ is a username while after the character is the hostname [119, 237, 238]. For instance, in the URL

http://www.mozilla.org/&item%djk3354@example.com/bad/evil/fraud.htm,

contrary to the appearance, the hostname is *example.com* and the username is *mozilla.org&item%djk3354*. Thus, the character can be used to disguise the true destination of a link. The use of the character for redirection purpose is not recommended in legitimate websites [237].

The character has also been used in passing email information as a URL parameter for authenticating users through URLs [239, 240]. However, since this method makes the data visible, thus leaking the data, it is not recommended by security experts to be used for any legitimate purpose [240, 241]. With this recommendation, it is less likely to find a significant number of legitimate webpages using this approach. On the other hand, through our sampled phishing URLs, we observed that the approach is popular in phishing webpages, mostly in the webpages requiring users to recover their email accounts. In this feature, we flag the webpage as a phishing one if we find the character in its URL.

Algorithm: Use of @ character in a URL

Input: URL

Output: 1: Phishing webpage.

0: Legitimate webpage.

Retrieve a URL.

If '@' or '%40' exists in the URL

Output → 1

Else

Output → 0

Feature 11: Domain Out of Position

To obfuscate URLs, phishers, in some cases, include legitimate domains in non-standard positions within URLs of their phishing webpages so as to confuse users who are not knowledgeable about legitimate domains [118, 119, 242]. For example, in the URL below, user may be tricked to think the webpage is in a *paypal.com* domain but actually it is hosted in a *2ipphoto.cn* domain.

http://2ipphoto.cn/https://www.paypal.com/cgi-bin/webscr?cmd=_login-run

In this feature, we detect this technique by checking if *http://*, *https://* and *www* characters as well as gTLDs and ccTLDs have been used more than once in a URL. If not, their positions will be determined if are different from the standard ones, such as in the URL paths.

Algorithm: Domain out of position

Input: URL

Output: 1: Phishing webpage.

0: Legitimate webpage.

Retrieve a URL.

If any of TLD, http://, https:// or www exists more than once in the URL,

Output → 1

Else if

If any of the http, https, www or :// exists in a URL path,

Output → 1

Else if

If a TLD exists in a URL path,

Output → 1

Else

Output → 0

Else

Output → 0

Feature 24: Free Domain Services

As described section 2.2.3.3, the use of free domain services has been growing in hosting phishing websites. Given that domain is an important element of a brand, most organizations are expected to host their legitimate websites in their own domains. We therefore flag any PDC webpage hosted in any of the free domains as a suspect of phishing activities. To extract the feature, we check if the webpage's domain string ends with a string matching a list of domains of the most abused free domain services we compiled from the Anti-Phishing Working Group (APWG)'s reports⁹ on global phishing survey between 2008 and 2017. The list can be found in appendix IV.

Algorithm: Free domain services

Input: URL

Output: 1: Phishing webpage.
 0: Legitimate webpage.

Retrieve a URL's domain.

If the domain ends with a free domain service's domain in the list

Output → 1

Else

Output → 0

4.4.1.3. WHOIS Records

This is a category of 4 features based on the domain's information recorded in the domain registrar. Two of the features are described below.

Features 27-28: Domain Age

As described in section 2.2.3.3, the majority of malicious domains are registered for short periods for the purpose of launching specific attacking campaigns for specific periods. Their WHOIS records will show that they have been registered recently. On the other hand, the domains of most legitimate websites are well established and thus they are expected to have

⁹ <https://www.antiphishing.org/resources/apwg-reports/>

longer domain ages. We determine the domain age (in days) by taking a difference between the current date and the domain registration date obtained from the WHOIS records. We also determine domain age of the form handler as another feature.

Algorithm: Age of domains of a webpage and a form handler

Input: Webpage

Output: 1: Domain age

-1: Unknown

Retrieve webpage's domain and form handler's domain.

For each {webpage, form handler}

Query the domain in the WHOIS database.

If WHOIS records are found,

Compute domain age (current date minus domain registration date)

Else

Output → -1

4.4.1.4. TLS Certificates

Here we propose 2 features based on the properties of the TLS certificate used by the webpage. The features are described below.

Feature 29: Type of TLS Certificate

The majority of phishing websites with TLS certificates use DVs followed by OVs while very few use EVs (see section 2.2.3.2 detailed descriptions of the certificates). Though a research by APWG [93] observed that the majority of phishing websites use the certificates, our sampled data indicated that majority of them do not use any type of TLS certificates. The certificates are recommended by security experts to be used by legitimate websites especially those collecting sensitive data. Among the three types of certificates, EVs are highly recommended due to their strictest procedure in validating domain owners, thus, offering the highest level of trust to users. Therefore, the majority of the legitimate websites are expected to use EVs compared to other types. In this feature, we first determine if the webpage uses a certificate or not. If it does, we determine the type of certificate used by extracting the policy identifier found in the certificate of the webpage. A DV certificate has an identifier *2.23.140.1.2.1* while for an

OV certificate it is 2.23.140.1.2.2 and for an EV certificate it is 2.23.140.1.1 [243]. If the policies are not found, we check for an organization name and jurisdiction country recorded in the subject field of the certificate. DV does not have both records while OV has a record of the organization name only. EV has both records.

Algorithm: Type of TLS certificate

Input: Webpage

Output: Certificate type

Retrieve a webpage.

If the URL starts with https protocol,

Retrieve TLS certificate information

If the certificate policies exist,

Retrieve policy identifiers,

If any of the identifiers ends with number 2.1

The certificate type is DV

If any of the identifiers ends with number 2.2

The certificate type is OV

If any of the identifiers ends with number 1.1

The certificate type is EV

Else if,

If both organization name and jurisdiction country name are null,

The certificate type is DV

If organization name is known but jurisdiction country name is null,

The certificate type is OV

If both organization name and jurisdiction country name are known,

The certificate type is EV

Else

The certificate type is unknown

Else,

The domain does not use a certificate

Feature 30: Domain, Certificate and Geolocation Country Matching

For most legitimate domains with gTLDs, the registered country of a domain in the TLS certificate is often the same as the country hosting the IP address of its web server. For domains

with ccTLDs, the country code often matches the country of the domain and that of an IP address of its web server. PhishLabs [23] and PhishLabs [99] have suggested that a significant number of phishing websites are hosted in countries different from those hosting their target legitimate websites. Meanwhile, a significant number of EV and DV certificates used in phishing websites are those of legitimate websites stolen from the databases of certificate authorities [244, 245]. These contain information of their target legitimate websites, including the domain's registered country. As a result, countries of web servers of phishing websites are likely to be different from those in the certificates and country codes.

Algorithm: Domain, certificate and geolocation country matching
 Input: Webpage
 Output: 1: Country matching status
If the webpage has a certificate and its webserver's geolocation is known,
 If the domain is ccTLD,
 Compare cc, country name in the certificate and country geolocation,
 Determine the country matching status
 If the domain is gTLD,
 Compare country name in the certificate and country geolocation,
 Determine the country matching status
Else,
 The country matching status is 'unknown'

4.4.1.5. Webpage Reputation

Here, we propose 5 features on the reputation of a webpage based on the information in the search engines and a blacklist of phishing URLs.

Features 31 – 33: Webpage Reputation in Search Engines

Legitimate webpages of established websites rank highly in search engines when they are searched for using URLs, hostnames or domains (i.e for a URL *https://moodle.bcu.ac.uk/my/*, the hostname is *moodle.bcu.ac.uk* and the domain is *bcu.ac.uk*). Search engines such as Google takes between 4 days to 4 weeks to get a new website fully indexed [246]. Since phishing webpages have an average lifetime of less than this period (see in section 2.2.3.3), the webpages

may not be full indexed during the time they are active. We propose three features for each webpage. We search for a webpage using three queries (URL, hostname and domain) using two search engines (Google and Bing). We use two search engines because different search engines take different times to index new webpages, thus one webpage may be visible in one search engine before being indexed in the other. The top five results returned by each query are compared against its query term. For instance, results of the URL query are compared with webpage's URL to see if any of them matches it exactly. If there is no match for either engine, we flag the webpage as a phishing one.

Algorithm: URL/hostname/domain reputation in search engines

Input: URL.

Output: 1: Phishing webpage.

0: Legitimate webpage.

Retrieve URL/hostname/domain and search for it using Google and Bing search engines.

For each item in {URL, hostname and domain}

If item matches with any of the Google's or Bing's top 5 results,

Output \rightarrow 0

Else

Output \rightarrow 1

Features 34-35: Shared Phishing Blacklisted Hosts

Attackers tend to economize their resources by re-using a few machines they own or those they have compromised to host many other malicious websites concurrently or at different periods (see section 2.2.3.3). As a result, the host of a recent phishing website is likely to have hosted old phishing websites which have been blacklisted by various phishing blacklists. We generate two features in this case. In the first feature, we count the number of times an IP address of the webpage's hostname matches the IP addresses of blacklisted phishing websites. A 3-month-old blacklist of phishing URLs from PhishTank is used for this analysis. However, it is a common practice for website owners to host the hostname and the domain in different web servers. For instance, *moodle.bcu.ac.uk* can be hosted in machine A while *bcu.ac.uk* is hosted in machine B. For this reason, we generate a similar feature but counts on the number of times

an IP address of the webpage's domain matches with same list of the IP addresses of blacklisted phishing websites.

Algorithm: Counts of matched hostname/domain IP in a blacklist

Input: URL

Output: Counts of matched IP address of a hostname/domain against IP addresses of blacklisted phishing websites

Retrieve the hostname/domain and query for its IP address.

Count the number of times an IP address of the hostname/domain matches the IP addresses of IP addresses of blacklisted phishing websites.

4.4.2. System Architecture of the Prediction Model

Training Process

Our prediction model based on the proposed features described above is built using the following five-step process (illustrated in Figure 4.1 as steps 1 to 5);

1. *Collection of known phishing and legitimate PDC webpages*

A set of each type of webpages is collected from its respective database and then labelled as phishing or legitimate accordingly. In this study, we collected active phishing webpages from a phishing blacklist while legitimate webpages were collected from a ranked list of the most visited websites.

2. *PDC webpage filtering*

The model is aimed at analysing only PDC webpages. This module, therefore, determines if a webpage consists of an HTML form or a JavaScript pop-up window and at least one of the PDC phrases as described in section 4.4.1.

3. *URL redirections check*

Some of the webpages are designed to perform one or more URL redirections before landing at their actual URLs. We need to obtain the final redirected URL of each webpage in order to collect its relevant URL based features. Checks are carried out for redirections embedded in the webpage structure, those provided through URL shortening services and those configured at the webserver. The former is indicated by the presence of a URL in the meta tag's *refresh* attribute in the head section of the webpage or in the JavaScript's window location attribute [247]. The shortened URL based redirections are determined by comparing the webpage's

hostname against a list of known shortening URL providers we collected (indicated in appendix III). If a match is found, short to long URL conversion is performed by using Untiny's online converter¹⁰. The latter are configured by the website administrator in the web servers such as Apache and Internet Information Services (IIS).

4. Feature extraction

All the features (described in section 4.4.1) of a PDC webpage are extracted from various local and external sources to build a training dataset.

5. Training a classifier

Use the training dataset to train a ML algorithm in order to build the classifier.

Prediction Process

The process of predicting a new webpage requested by a user is shown as steps 2 to 8 in Figure 4.1. The webpage is retrieved from a webserver after being requested by user. A check (2) is performed to establish whether it is a PDC web page or not. If it is not, it is passed to the browser and displayed to the user. If it is a PDC webpage, any redirections are resolved (3). Its features are extracted (4) and passed as an input to the classifier which makes a prediction (7-8). If the webpage is classified as phishing, the user's access to the webpage is blocked or warned otherwise it is permitted. The designs of phishing webpages are likely to change over time as phishers adapt their methods to evade detection. We, therefore, propose periodic addition of new phishing webpages and re-training of the classifier to ensure the classifier always provides optimal performance.

¹⁰ <http://untiny.com/>

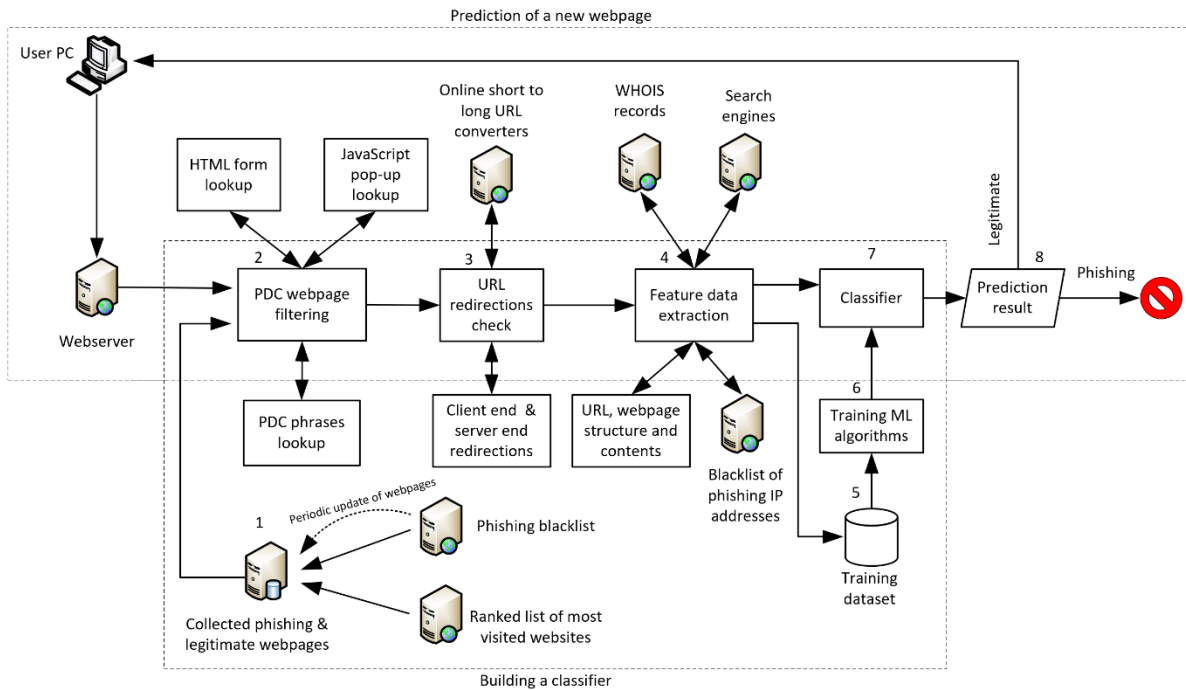


Figure 4.1. A system architecture of the proposed model for predicting zero-day phishing PDC webpages.

4.5. Experiments

We designed a number of experiments to build and evaluate a binary classifier that distinguishes phishing PDC webpages from the legitimate ones. We ran two sets of experiments with the aim of identifying the most accurate set of features and the best performing ML algorithm for the classifier. In the first set, eight traditional ML algorithms namely Linear Regression (LR), k-Nearest Neighbour (k-NN), Decision Tree (DT), Naive Bayes (NB), SVM, Artificial Neural Network (ANN), Random Forest (RF) and Gradient Boosting (GB), were used. In the second set of experiments, we evaluated using three DL algorithms namely Fully Connected Feedforward Deep Neural Networks (FC-DNN), Long Short-Term Memory (LSTM) and One Dimensional Convolutional Neural Network (1D CNN). Some of these algorithms have been described in section 3.5. We use eight standard ML performance metrics namely accuracy, FPR, FNR, precision, recall, F1-score, AUC and ROC (described in section 3.4.5) to compare performance results of the algorithms.

4.5.1. Experimental Setup

The ML experiments were run on a machine with MS Windows Home, 16GB memory and Intel's i7 processor specifications. DL based experiments were run on Jupyter notebooks hosted in Google's Collaboratory platform. We developed and used a Python application to extract and pre-process data, create a training dataset, and train and evaluate the algorithms. Python libraries Scikit-learn, Keras, Pandas, NumPy and Seaborn, amongst others, were used for data analysis and development of the classifier. Extracted data was stored in the MySQL database. We also used WEKA, an ML platform, to perform feature selection.

4.5.2. Data Collection

We collected 13,494 legitimate and 12,621 phishing PDC webpages between 22nd of September and 7th of October 2019. To obtain the legitimate webpages, we first collected more than 100,000 top websites from a ranked list of 1 million most visited websites from Tranco¹¹. Tranco filters the websites against a Google's Safe Browsing blacklist to remove all known malicious websites, thus the list is expected to contain legitimate websites only. Using a hostname of each website combined with each of the PDC phrases listed in Table 4.1 at a time, we searched for candidate PDC webpages related to these websites in the Google and Bing search engines. We extracted the maximum possible number of URLs returned by each search, downloaded the webpage of each URL and then checked (using the PDC webpage filtering procedure described in 4.4.2) whether it prompts for personal data or not. Finally, features of each of the confirmed PDC webpage were extracted using our application and added to the MySQL database.

We obtained a list of phishing PDC webpages from an online repository of confirmed phishing URLs managed by PhishTank¹². The database is one of the most reliable sources of blacklisted phishing URLs in the cybersecurity domain. Since the database is updated hourly, we downloaded its list four times a day over a span of 5 days. In each list, we retrieved each active URL, downloaded its webpage and then checked whether it is an PDC webpage or not. We then extracted the features of all the qualified webpages and recorded them in the database. In several cases, multiple URLs were observed to belong to the same hostname, hosting the same

¹¹ <https://tranco-list.eu>

¹² <https://www.phishtank.com>

or different PDC webpages. To avoid the possibility of excessive influence of a few hostnames leading to a biased model, we limited each hostname to at most 20 unique URLs.

PDC Webpage Type	Source	Size
Legitimate	Tranco’s list of most visited websites, Google and Bing search engines	13,494
Phishing	PhishTank online repository	12,621
Total number of webpages		26,115

Table 4.3. Summary of the collected data used to build the training dataset.

4.5.3. Data Pre-processing

We applied several standard data pre-processing techniques (described in section 3.4.3) as follows. We first identified features with missing values, as summarised in Table 4.4. There is no standard threshold percentage to determine whether a feature with the missing data should be used or not. For this study, we set a threshold of 50%, which is a commonly used by many practitioners [248, 249]. We therefore dropped features 4 and 3 (feature numbers indicated in Table 4.2) as they exceeded the threshold. We also analysed correlations between the features using Pearson’s correlation matrix (shown in appendix V) in order to determine redundant features. We dropped features 24, 25 and 30 because they have correlation values of 1.0 with features 22, 23 and 29 respectively. We then encoded all categorical features as unique numeric values, with missing values given their own unique labels.

Four imputation methods (mean, median, most frequent and k-NN (k=4), which are commonly used in replacing missing values in numerical features, were compared. We found that mean imputation produced the best performance when we ran one of the algorithms (RF) on the dataset and therefore it was used to replace the missing values. Finally, we applied a data scaling technique (described in section 3.4.3.4) to standardize data ranges of all the features by transforming the data in each feature such that its distribution has a mean value 0 and standard deviation of 1.

Feature #	Feature Name	% Missing Values
4	Domain in alternate URL	91.4
3	Domain in canonical URL	87.2
2	Domain identity in copyright	48.0
26	Type of TLS certificate	45.4
27	Domain, certificate and geolocation country matching	42.3
22	Domain validity	14.2
23	Domain age	14.1
24	Form handler's domain validity	14.1
25	Form handler domain's age	14.1
6	Ratio of void and same webpage links	1.9

Table 4.4. Features with missing values.

4.5.4. Performance Results

In this section, we present and compare prediction results of the algorithms for both traditional ML and DL based experiments.

4.5.4.1. Results of Traditional Machine Learning Algorithms

The 8 standard ML algorithms were run and their results were compared to identify the best performing algorithm for the classifier. First, feature selection using a backward feature elimination method (described in section 3.4.3.3) was performed which identified a subset of 26 features to be the best features for the classifier. Stratified cross validation technique (k -fold where k is 10) (described in 3.4.5) was applied to train and test the algorithms in order to obtain their average prediction scores. Table 4.5 summarizes results of the untuned algorithms and Figures 4.2 show the performances of ML algorithms across all threshold values in their ROC curves. The results indicate that RF has the best performance across all metrics except for FNR, thus identified as the best algorithm for the implementation of the classifier. The RF was then tuned using a Random Search method (described in section 3.4.6) to optimize its performance. The tuning of RF yielded an **accuracy of 98.56%**, **FPR of 1.12%** and **FNR of 1.17%**. Values of hyperparameters of the tuned RF are indicated in Table 4.6.

Algorithm	Accuracy (%)	FPR (%)	FNR (%)	Precision	Recall	F1 Score	AUC
LR	94.87	3.71	1.48	0.96	0.93	0.95	0.98
K-NN	95.72	2.91	5.74	0.97	0.94	0.96	0.98
DT	97.42	2.59	2.57	0.97	0.97	0.97	0.97
NB	78.48	1.33	43.11	0.98	0.57	0.72	0.96
SVM	96.00	2.48	5.63	0.97	0.94	0.96	0.99
ANN	96.94	2.18	3.99	0.98	0.96	0.97	0.99
RF	98.45	1.13	2.00	0.99	0.98	0.98	1.0
GB	97.89	1.48	2.77	0.98	0.97	0.98	1.0

Table 4.5. Performance results of the standard ML algorithms.

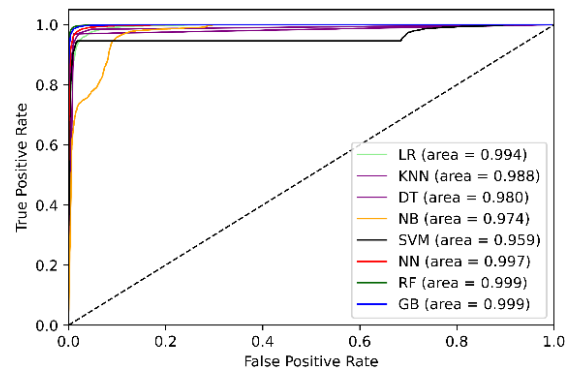


Figure 4.2. ROC curves of the trained traditional ML algorithms.

Hyperparameter	Description	Value
n_estimators	Number of trees	1200
max_features	Max number of features considered for splitting a node	log2
max_depth	Max number of levels in each decision tree	31
min_samples_split	Min number of data points placed in a node before the node is split	6
min_samples_leaf	Min number of data points allowed in a leaf node	1
bootstrap	Method for sampling data points	false

Table 4.6. Values of hyperparameters of the tuned RF.

4.5.4.2. Results of Deep Learning Algorithms

The three DL algorithms were run and their results were compared with the traditional ML algorithms to identify the best performing algorithm for the classifier. First, the training dataset for LSTM and 1D CNN was converted into a time series shape, a standard input data format for the two algorithms. Time step value was assigned to 1 since the features of each URL were collected and input into the algorithms at the same time instance. Each of the three algorithms was then tuned using a Random Search method. Table 4.7 indicates ranges of values of hyperparameters we selected for tuning. We also attempted to tune with multiple hidden/convolution layers. We found that only one layer was sufficient to produce optimal performance in all three algorithms. Additional layers did not improve the performances. The optimal values of the hyperparameters identified by the method were used to build the classifiers. The final result of each classifier was obtained by taking an average of the performances of five runs of the tuned classifier. Table 4.8 summarizes performance results of the tuned algorithms. Figures 4.3a – 4.3c show network architectures of the tuned DL algorithms along with the optimal values of the tuned hyperparameters.

Hyperparameter	Range of Evaluated Values
Number of : neurons in dense layers / memory units in a hidden layer of LSTM / filters in a convolution layer of CNN)	10, 30, 50, 80, 100, 150, 200, 300, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000, 2200, 2400, 2800, 3000
Activation functions	Relu, tanh, sigmoid, hard_sigmoid, linear, softmax, softplus, softsign
Optimization algorithms	SGD, RMSprop, Adagrad, Adadelata, Adam, Adamax, Nadam
Learning rates	0.001, 0.01, 0.1, 0.2, 0.3
Kernel initializers	Uniform, lecun_uniform, normal, zero, glorot_normal, glorot_uniform, he_normal, he_uniform
Dropout rates	0.1, 0.2, 0.3, 0.4, 0.5
Batches	15, 30, 50, 70, 90, 110, 130, 150
Epochs	10, 30, 60, 90, 120, 150, 180, 210, 240, 270, 300

Table 4.7. Hyperparameters and their ranges of values evaluated for tuning the three DL algorithms.

Algorithm	Accuracy (%)	FPR (%)	FNR (%)	Precision	Recall	F1 Score	AUC
FC-DNN	97.28	2.13	3.33	0.98	0.97	0.97	0.97
LSTM	95.71	3.50	5.11	0.96	0.95	0.96	0.99
CNN	95.66	3.14	5.61	0.97	0.94	0.95	0.98

Table 4.8. Performance results of the DL algorithms.

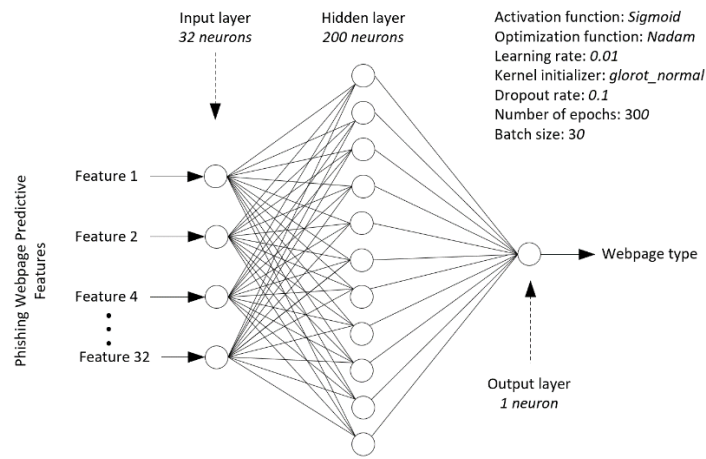


Figure 4.3a. FC-DNN architecture of the classifier.

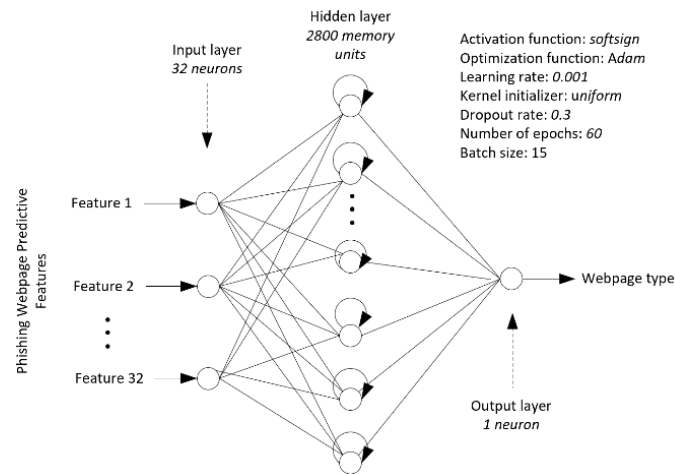


Figure 4.3b. LSTM architecture of the classifier.

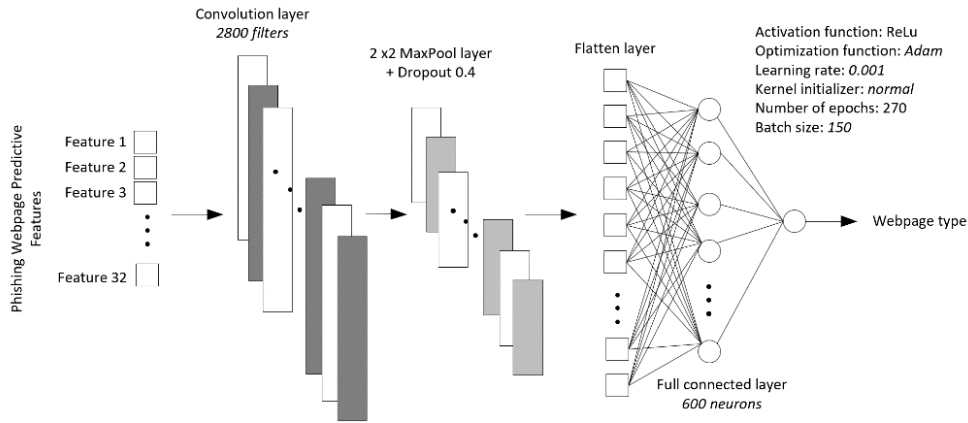


Figure 4.3c. 1D CNN architecture of the classifier.

4.5.4.3. Overall Results and Feature Analysis

We found that tuned RF achieved the highest performance of all the evaluated algorithms across most metrics. With the exception of the NB algorithm, all the algorithms, however, yielded good performance values in all metrics. Given that the algorithms use different assumptions to develop their prediction rules, the observed good results across the 10 algorithms show that our features are generally effective in predicting phishing webpages.

Table 4.9 breaks down the features in the full set (35 features) and best set (26 features) by category. There is a high representation of features from each category in the best set, with an exception of WHOIS records, which has only 1 out of 4 in the best set. This indicates that all the categories are important in the prediction although at different extents. Using the tuned RF, we evaluated the performances of the best feature set belonging to a specific category only. The results are shown in Figures 4.4a and 4.4b. Webpage reputation, URL structure, and webpage structure and contents categories produced the highest accuracy and lowest FPR rates whereas the TLS certificate category achieved the worst of the same, although with one of the lowest FNRs. The former, therefore, are the strongest predictors while the latter is the weakest. We also evaluated the performance of combinations of some of the high performing feature categories of the best features to determine the combinations that are more useful for prediction. As indicated in Table 4.10, the combination of feature categories 1, 2, 3 and 5 (see Table 4.9 for an explanation of the category numbers) produced the highest performance, which was closest to the overall performance obtained using all five categories.

Similarly, we evaluated the performance contributions of the novel and existing features in the set of best features and compared them against the overall set of best features (see Figures 4.5a and 4.5b). While our novel features achieved better results compared to the existing ones, a combination of the two increased the overall accuracy and lowered the error rates especially the FNR. This suggests that their combination is important for achieving optimal prediction performances, as for the case of feature categories. Increasing diversity of the features is also likely to have a benefit in terms of hardening the solution against detection evasion techniques.

Feature Category #	Feature Category	Tally of Features in the Full Set	Tally of Features in the Best Set	Best Features # (See Table 4.2 for feature #)
1	Webpage structure and contents	7	5	1-2, 5-7
2	URL structure	17	14	9-19, 22-24
3	WHOIS records	4	1	27
4	TLS certificate	2	2	29-30
5	Webpage reputation	5	4	31-32, 34-35

Table 4.9. Distribution of feature categories in the set of the 26 best features.

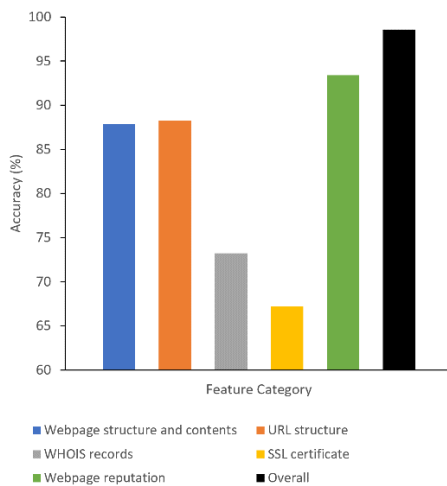


Figure 4.4a. Accuracy rates of the feature categories.

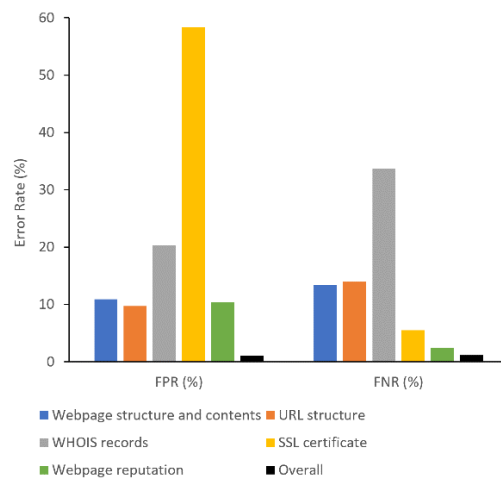


Figure 4.4b. Error rates of the feature categories.

Combination of Feature Categories	Accuracy (%)	FPR (%)	FNR (%)	F1 Score
1 + 2	94.21	4.16	7.54	0.94
1 + 3	92.26	7.21	8.31	0.92
1 + 5	97.24	2.44	3.10	0.97
1 + 2 + 3	95.28	3.41	6.12	0.95
1 + 2 + 3 + 5	98.45	1.42	1.85	0.98
2 + 3	91.77	6.71	9.85	0.91
2 + 5	97.73	2.10	2.46	0.98
2 + 3 + 5	98.04	1.82	2.12	0.98

Table 4.10. Performances of various combinations of the feature categories of the best features.

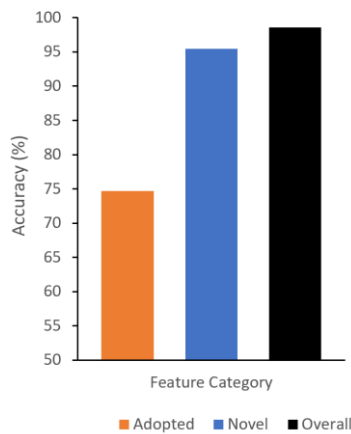


Figure 4.5a. Accuracy rates of the existing and novel features.

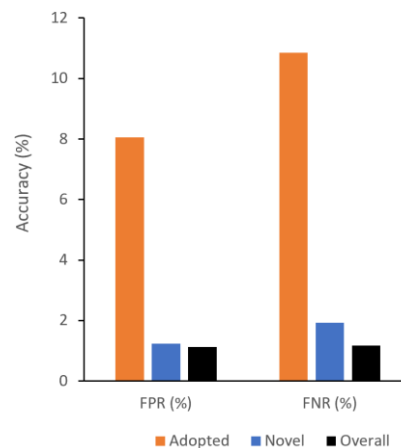


Figure 4.5b. Error rates of the existing and novel features.

To evaluate individual features, feature importance weights of the best features were computed using the tuned RF to determine the prediction strength of each feature relative to others. Figure 4.6 shows the ranking of the 20 novel and 6 existing features in terms of their importance weights. The feature with the largest weight indicates the strongest predictor while the one with the lowest weight is the weakest predictor. As a general observation, the novel features are observed to rank higher than to the adopted ones. The top 5 features are the novel ones while the least 3 are the adopted ones. Also, 6 of the 9 third party-based features are ranked higher compared to most of the 17 local features (those based on URL structure, and webpage structure and contents). 6 of the top 7 features are third party-based features. This indicates that the novel and third party-based features tend to be more effective for this prediction problem. Most

of the webpage reputation-based features are the strongest predictors while the weakest ones consist mostly of the URL structure-based features.

The highest ranked feature is feature 34 (refer to Table 4.2), which is related to the number of times the hostname's IP address matches with IP addresses of blacklisted phishing websites. The feature's data distribution shown in Figure 4.7a (presented using boxen plots - see the description of boxen plots in section 3.4.3.1) explains this by showing that hosts of unknown phishing hostnames match with a large number of hosts previously known to host backlisted phishing websites while only a small number of legitimate hostnames does the same. The distribution suggests that many phishers use a small number of machines to host multiple phishing websites at different times. Meanwhile, the small number of hosts of legitimate websites that were matched indicates that phishers also use compromised legitimate hosts to do the same. This feature and feature 35, which is related to the number of times the domain's IP address matches with IP addresses of blacklisted phishing websites, have a moderate correlation value of 0.6 (see appendix V), showing that there is medium level of correlation between them. This, combined with the difference in ranking between the two features, suggests that phishers, in some cases, host their hostnames and domains in different machines. We confirm this trend in our dataset by observing that some of the phishing webpages have different counts for features 34 and 35. We think phishers use the approach to limit the impact of blacklisting their services, that is, if a host of the hostname is taken down, the domain can still operate.

Hostname matched in a search engine's top 5 results (feature 32) is the second ranked feature. As shown in Figure 4.7b, about 90% of the phishing hostnames are not returned in the search results, suggesting either that they were not indexed because their webpages were recently created, or that the webpages did not meet the search engines' high ranking indexing criteria. Conversely, about 90% of the legitimate hostnames were returned in the search results. Counts of a domain identity appearing in a webpage structure and contents (feature1) appears in 3rd place and is the highest ranked feature based on webpage structure and contents. Figure 4.7c shows that a larger number of legitimate URLs contains domain identity which is appearing multiple times in a webpage comparing to those of phishing URLs. This confirms that many phishing webpages exhibit a mismatch between the domain names in their URLs and the

identities of the organizations the webpages appear to belong to, thus, most of them were being hosted in the domains registered by attackers.

Domain age (feature 27) takes the 6th position and it is the only feature from the WHOIS records category in the ranking. Its distribution of data (see Figure 4.7d) shows that phishing domains have shorter domain ages, with majority of them have ages between 1000 and 4000 days and with median value of just below 2000 days, while legitimate domains have longer domain ages, in which the majority have ages between 3500 to 8000 days and a median value of just below 6000 days. The observed domain ages of the phishing domains are still significantly longer than the expected ones reported in section 2.2.3.3. This suggests that attackers have generally increased the duration of their domains staying active, possibly for the purpose of evading detection techniques based on domain age.

The TLS certificate type (feature 29) is the 7th feature and the highest ranked of the two features in the TLS certificate category. Figure 4.7e shows the breakdown of certificate usage by type for the two webpage classes. The majority of phishing webpages still do not use any certificate. There are more of them using DV than OV and EV, suggesting that phishers are taking advantage of the least strict validation procedures in obtaining DV certificates to trying to make their webpages as more legitimate as possible. We had expected EV certificates to be popular among legitimate websites due to their high security and user trust but in fact it is the least popular category and a significant percentage of them do not use any certificate at all. This shows that most of the legitimate websites' owners are yet to understand the significance of using a digital certificate, more importantly the EV certificate, in their websites in improving security and users' trust.

The number of characters in the URL path (feature 19) takes the 8th position and is the highest ranked feature based on URL structure. From Figure 4.7f, phishing webpages tend to have longer URL paths. This is consistent with other features related to URL path including number of forward slashes (feature 17), number of numeric in a URL path (feature 22), number of dots in a URL path (feature 13), number of obfuscation characters in a URL path (feature 16), number of encoded characters in a URL path (feature 9) and the use of obfuscation characters in the hostname (feature 15) being among the best predictors (at 11th, 13th, 14th, 15th, 16th and 21st ranking positions respectively). This suggests that phishers obfuscate their true URLs

through the addition of various characters in URL paths, which increases length of the paths. The three URL based features (feature 14, 10 and 23), which were also adopted from other works, were the least ranked ones thus were the weakest ones among the best features.

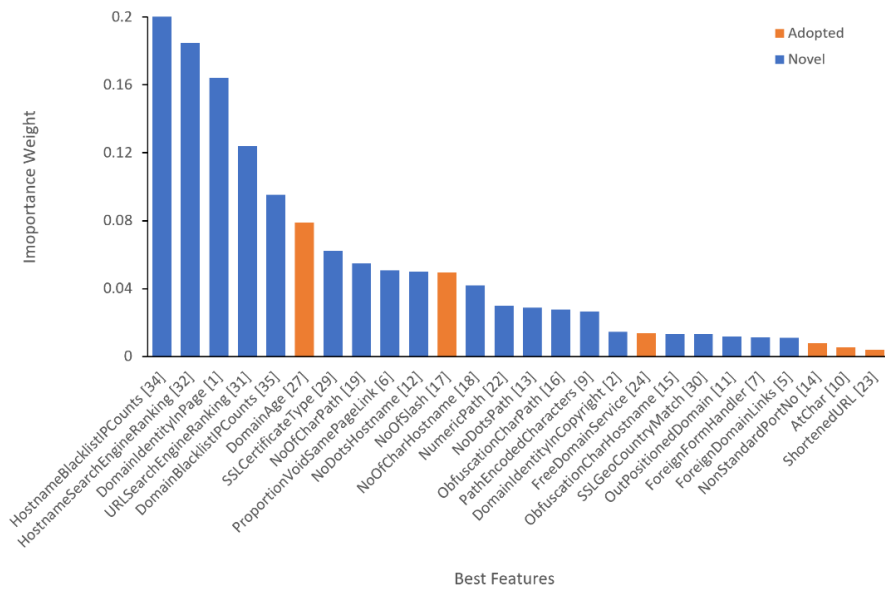


Figure 4.6. Ranking of the best features by importance weight. Numbers in the brackets represent numbers of the features as indicated in Table 4.2.

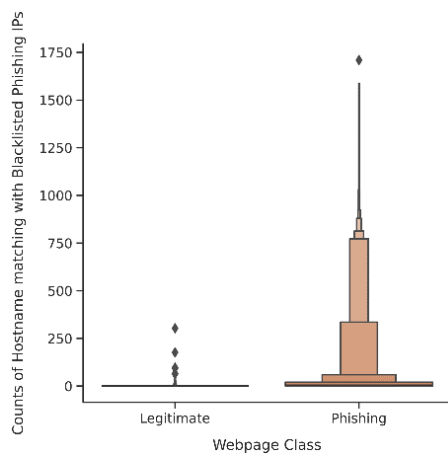


Figure 4.7a. Data distribution of counts of hostname’s IP address matching with phishing blacklisted IP addresses.

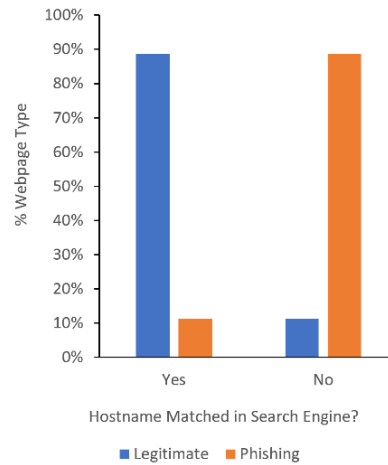


Figure 4.7b. Data distribution of the matching of hostnames in a search engine’s results.

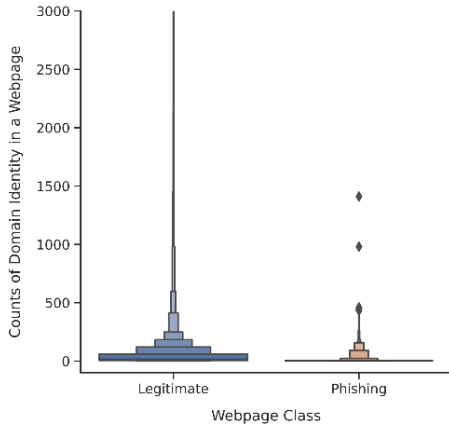


Figure 4.7c. Data distribution of counts of domain identify appearing in the webpage structure and contents.

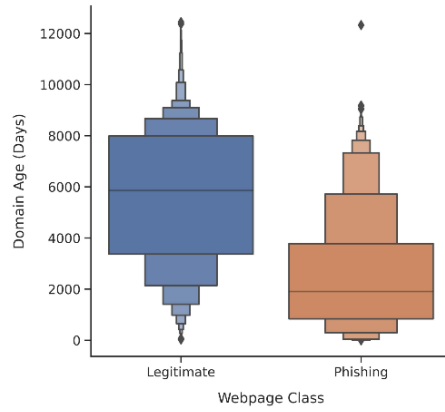


Figure 4.7d. Data distribution of domain age.

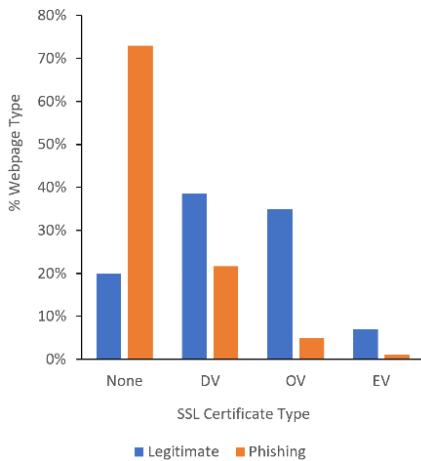


Figure 4.7e. Data distribution of types of TLS certificate.

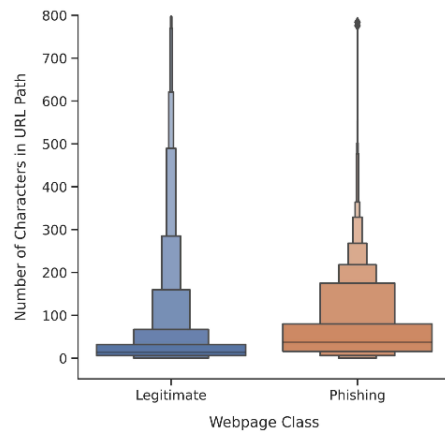


Figure 4.7f. Data distribution of counts of characters in a URL path.

4.5.5. Detection Time Analysis

We measured the runtimes of the model's stages (described in section 4.4.2) namely retrieval of a webpage from its server, PDC webpage filtering, URL redirections check, feature extraction, training the dataset and prediction analysis. Table 4.11 summarizes the average times. We only measured the feature extraction time for the 26 features in the best set. The sum of the average times of the stages in the classification process is a little under 7.2 seconds. Feature extraction is responsible for about 75% of the overall detection time. We observed that the extraction of 9 third party and 17 local based features take 3.05 and 2.31 seconds

respectively, so the average time for a third-party feature is 0.34 seconds and that for a local feature is 0.14 seconds. Overheads in data retrieval from the third parties' servers and network overheads are likely to be the main reasons for the difference. Comparing the extraction times for each of the third-party features with the average time for a local feature, Figure 4.8 shows that the blacklist and the search engine-based features have the longest retrieval times. They all take longer than the average local feature. It is important to note that the runtime of each activity could be improved through use of more efficient Python libraries and code optimisation. Also, the features were queried and generated sequentially and it is likely that overall speed could be improved by introducing some concurrency.

Module	Runtime (s)
PDC webpage loading	0.8430
PDC webpage filtering	0.0976
URL redirections check	1.2959
Feature extraction	5.3600
Prediction analysis	0.0002
Total prediction time per webpage	7.1537
Training a classifier	18.2300

Table 4.11. Runtimes of the model's modules.

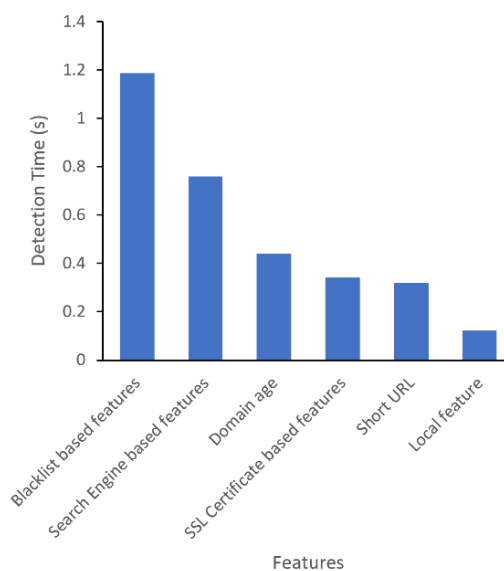


Figure 4.8. Comparison of extraction times of the best features.

4.5.6. Model Validation Using New Data

Phishers are likely to adopt new ways of designing their phishing webpages over time to evade some of the detection features. To check whether the performance of the model degraded with time, we collected a new testing dataset consisting of 2,736 legitimate and 2,498 phishing PDC webpages between 8th and 9th of February 2021, 14 months after we collected the training dataset. After performing similar data pre-processes as described in section 4.5.3, the processed data was tested against the tuned RF algorithm described in section 4.5.4.1 using the training dataset described in section 4.5.2. The classifier achieved an **accuracy of 98.87%**, **FPR of 1.13%** and **FNR of 1.79%**. Compared with the results reported in section 4.5.4.1, the model performed even better in terms of the accuracy but slightly low in terms of the error rates. The results confirm that the excellent performance achieved was not restricted to the specific training dataset and that it remained effective in detecting phishing webpages after over a year. The latter indicates that frequent retraining will not be required to adapt to new tactics employed by phishers.

4.6. Discussions

4.6.1. Comparison with Existing Works

We compare the performance of our work against works which have also used ML to predict phishing webpages as a binary classification task. Since dataset size has a significant impact on the prediction performance, we select only those which used a dataset size of at least 10,000 webpages, similar to ours in terms of the magnitude, for a fair comparison. We exclude those that used very small dataset sizes (in a magnitude of hundreds and thousands) (see appendix I). The comparison is done in terms of prediction performance, diversity of features, and number of ML algorithms and metrics used for evaluation. Table 4.12 provides a summarized comparison chart.

Work	Feature # and Categories	Data Size (URLs)	Evaluation Algorithms	Performance
Ma, et al. [120]	5 WHOIS records, 9 URL, 1 network, 1	35,500	NB, SVM, LR	Acc = 95% – 99%

	geolocation, 14 webpage reputation features			Error rates = 0.9% – 33.5%
Xiang, et al. [118]	7 URL, 4 webpage contents, 1 WHOIS record and 3 webpage reputation	13,064	SVM, LR, Bayesian Network , DT, RF, Adaboost	Acc = 92.3% FPR = 1.38% FNR = 0.95%
Shirazi H. [48]	30 URL, webpage structure, WHOIS records and webpage reputation features	12,000	SVM, FC-DNN	TP = 89% TN = 90.3% AUC = 0.9
Sahingoz, et al. [50]	1 webpage contents and 40 URL structure features	73,575	NB, RF , kNN, Adaboost, K-star, SMO and DT	Acc = 97.97% Prec = 0.97 Recall = 0.99 F1 = 0.98
Our Work	5 webpage structure and contents, 14 URL, 1 WHOIS record, 2 TLS certificate, 4 webpage reputation	26,115	LR, K-NN, DT, NB, SVM, ANN, RF , GB, FC-DNN, LSTM, 1D CNN	Acc = 98.56% FPR = 1.12% FNR = 1.17% Prec. = 0.99 Recall = 0.98 F1 = 0.99 AUC = 1.00

Table 4.12. Performance comparison of some of the related works with our work. The reported performances are of the best performing algorithms in bold.

4.6.1.1. Prediction Performance

Our work has produced better performance in most metrics compared to the other works. Note that Ma, et al. [120] evaluated their model against several datasets from different data sources with an accuracy ranging 95% and 99% and error rates between 0.9% and 33.5%. Although their best performance is superior to ours, their least good is significantly worse than ours. The performance of our model against two independent datasets, as described in sections 4.5.4.1 and 4.5.6, showed less variation than theirs, suggesting that our model is more robust. In addition, enlarging our dataset to a size similar to Ma, et al. [120]’s is likely to increase our work’s performance to close the gap or outperform their highest performance. It can be

observed that Sahingoz, et al. [50]’s work achieved a performance close to ours. However, our work has used a new set of features. This prolongs our model from the exposure of possible detection evasion techniques as it takes time for phishers to learn about existing features before attempting to evade the features, thus the solutions. Other key differences with our work are the use of large numbers of different categories of features and third-party features. The advantages of these are elaborated in the next sections. In addition, Sahingoz, et al. [50] used a dataset of three times the size of our dataset and still achieved a lesser performance than ours. As ML algorithms tend to increase their performance with an increase in the dataset sizes, our model is likely to perform better than the current performance if our dataset is increased to the same size as theirs.

4.6.1.2. Diversity of Features

Along with Ma, et al. [120], our model has used 5 different categories of both third party and local based features. Sahingoz, et al. [50] used 2 categories of local based features only while the other two works used 4 categories of third party and local based features. Using a large number of feature categories increases the difficulty faced by an attacker in evading detection. The attacker would need a wide range of knowledge and skills to develop techniques that are able to circumvent most or all types of features. Furthermore, local based features can more easily be evaded by attackers than the features derived from the information obtained from third party services. For instance, a phishing webpage created by copying a legitimate webpage and slightly modifying URL is difficult to distinguish from the legitimate webpage and is unlikely to be identified using the features proposed by Sahingoz, et al. [50]. Third party features, which are the strongest predictors in our model, are always difficult to emulate or forge for the reasons mentioned in section 4.3. We think that a mixture of third party and local based features is ideal for a more robust solution.

4.6.1.3. Number of ML Algorithms Used for Evaluation

While other works have used between 2- and 7-ML algorithms to evaluate their classifiers, our work has used 11. Our study is the first to evaluate LSTM and ID CNN algorithms for this problem. They both have outperformed accuracies reported by Xiang, et al. [118] and Shirazi H. [48], two of the studies in the table with the least performances. The advantage of exploring a large number of algorithms is that it increases a chance of finding an optimal performance of

a problem with features of mixed data distributions, such as this, as different algorithms learn the same data distribution differently. In addition, it allows us to draw conclusions on the general effectiveness of the features for a given prediction problem. From the table, two of the works (ours and that of Sahingoz, et al. [50]) achieved among the highest performances with a Random Forest algorithm. This suggests that the algorithm is likely to be more suitable for this classification task.

4.6.1.4. Performance Metrics

Our work has evaluated the classifier using 8 performance metrics while the other works have used between 2 and 4 metrics. Important measures such as precision, recall and F1 score were not reported by the first three studies in the table. FNR was not reported explicitly by three of the four studies. It is an important one as it measures the extent to which solutions misclassify true phishing websites, thus exposing end users to the attack. Evaluation using a small number of measures limits our understanding on the all-round effectiveness of the solutions.

4.6.2. Application of the Proposed Model

The webpage loading time affects the web browsing user experience which in turn determines the percentage of users that are likely to decline accessing the webpage. The percentage increases as the loading time increases. As indicated in Table 4.10, our model takes 0.84 seconds to load an PDC webpage in a desktop device and 6.31 seconds to predict it, giving a total prediction time of 7.2 seconds. According to MachMetrics [250], at least 30% of users are likely to abandon the webpage if the loading time exceeds 7 seconds. This is a significant loss to any website especially the commercial ones. This means our model, if implemented as it is, will be less than ideal for use as a real time application. However, the model's prediction time can be reduced in two ways; (1) using more efficient python libraries and coding style and (2) extracting most of the features in parallel. For instance, all the third-party features can be extracted concurrently thus reducing their total time of 3.05 seconds to the longest time to extract one of their features, which is 1.19 seconds. Similarly, local based features can be extracted in parallel. The average time to extract one such feature is 0.14 seconds. The total time to load the webpage and extract all the features in parallel would be 2.17 seconds ($0.84 + 1.19 + 0.14$). This time is less than 3 seconds, which is a range considered to be fast by many users according to MachMetrics [250]. With this improvement, our model can thus be

implemented as real time application to protect users at the web browser as a built-in functionality or as a plug in, for instance.

Alternatively, our model can be used to build a blacklist of phishing URLs by predicting phishing webpages from PDC webpages collected from various data sources such as emails and social media posts. The blacklist can then be used to defend users by integrating it with a web browser as a built-in functionality or as a plug in. The blacklist can also be used to complement existing general-purpose blacklists for research purposes.

4.7. Summary

In this chapter, we have proposed an ML based model that can instantly and accurately predict both known and zero-day phishing PDC webpages using a novel set of highly diversified features. First, we investigated and proposed 35 features derived from various distinctive structural characteristics of phishing PDC webpages. Of the 35 features, 26 of the features were found to be the most relevant features for the prediction task, producing an accuracy of 98.56%, FPR of 1.12% and FNR of 1.17% with a Random Forest algorithm. The 26 features, in which 20 of them are introduced by this study and the rest were adopted from previous studies, are grouped into five categories namely webpage structure and contents, URL structure, WHOIS records, TLS certificate and webpage reputation. 9 of the features are based on the third-party services while 17 of them were derived from the webpage's structure. Our feature analysis indicated that novel and third party-based features tend to be stronger predictors than the adopted and local based features. We also found that most of the features based on webpage reputation against blacklisted phishing IP addresses and search engines are the most influential ones for the prediction whereas URL based features are among the least influential ones. The prediction time of the model was measured at 7.2 seconds but the time could go as low as 2.17 seconds if the features were to be extracted concurrently. The time suggests that the model can be used for real-time protection of users from accessing phishing PDC webpages without degrading their web browsing experience. We also tested the model (without retraining) against a new dataset collected 14 months after collecting the first dataset. The results showed that the model performs consistently on different datasets, suggesting that the features are reliable for addressing the problem in long term. They also show that phishers do not vary their tactics in creating websites frequently.

To apply the model to protect users in a production environment, we recommend increasing the training dataset size to a magnitude of hundreds of thousands. This is likely to further improve the prediction performance given that some of the algorithms especially the DL algorithms continue to improve their performance as dataset size increases. With large datasets, we expect different data distributions which might affect the ranking of prediction strengths of the features. However, as illustrated in this study that the same feature set perform similarly on two datasets of different sizes, we do not expect to any change in the composition of the best feature set. In this case, however, it is likely to see a different algorithm apart from RF that performs the best.

Chapter 5

Detection of Zero-Day Phishing Hostnames Hosted in Fast Flux and Name Server IP Flux Networks Using Machine Learning

5.1. Introduction

In Chapter 4 we proposed a technique to predict phishing PDC webpages based on features derived from their structure and contents. In this chapter, we focus on the prediction of phishing PDC webpages, through their hostnames, based on the structural and operational characteristics of the networks hosting them. In Chapter 2 we learned that phishing websites can be hosted in non-flux networks or IP flux networks. In the former, cybersecurity experts have been proficient at taking down the websites by tracking and blacklisting the hosts through their consistent IP addresses. In order to evade the blacklisting approach, thus increasing longevity of the websites, attackers have been taking an approach of hosting their websites in FFSNs in which the content hosts are shielded by flux agents. Other attackers implement NS fluxing behaviour in the networks of their authoritative NSs in order to protect the NSs from tracking, in some cases combined with the fast-fluxing behaviour. As there are no legitimate applications of NSIFNs, any PDC webpage that keeps its A records in NSIFNs is a suspect of phishing activities. The use of such networks makes phishing websites difficult to shut down, thus they have greater impact. Effective and efficient solutions to detect the websites hosted in these networks are critical in order to address the phishing problem effectively.

Detecting hostnames hosted in FFSNs and NSIFNs can be useful in complementing the technique proposed in Chapter 4 to warn users that they are visiting phishing webpages. Furthermore, a database of such hostnames can be used by cybersecurity stakeholders such as Internet Services Providers (ISPs) to investigate and monitor flux networks in order to identify the compromised legitimate networks hosting the flux agents. This can help owners of the compromised networks to clean their machines and take precautions to prevent reinfection. Also, solutions such as those proposed by Gu, et al. [251] and Khattak, et al. [252], which monitor data traffic between flux agents in the local networks and their external motherships, can be used at the network gateways to track the motherships in order to target them for take down, thus shutting down the entire infrastructure of the phishing campaigns.

Chapter 2 described the distinctive DNS, network and host characteristics of FFSNs and NSIFNs which can be used to distinguish malicious hostnames, and thus websites, hosted in these networks from those hosted in non-flux networks. Based on these characteristics, this chapter proposes supervised ML approaches for fast and accurate prediction of zero-day phishing hostnames hosted in FFSNs and those hosted in NSIFNs using novel sets of predictive features. Due to the similarity of most characteristics of FFSNs to those of NSIFNs, as indicated in section 2.3.3, a number of predictive features we propose for the first prediction task are also proposed and evaluated for the second task.

Part of the work in this chapter (the first prediction task) was published in a paper titled “A Machine Learning Approach for Detecting Fast Flux Phishing Hostnames” at the Journal of Information Security and Applications in March 2022. The paper can be accessed through doi: 10.1016/j.jisa.2022.103125. The main contribution of the paper is the proposed 56 features, based on DNS, network and host characteristics, for instant and highly accurate detection of zero-day phishing hostnames hosted in FFSNs using a supervised ML approach.

This chapter is organized in four sections. Section 5.2 describes the development of an ML model for predicting phishing hostnames hosted in FFSNs. Section 5.2.1 describes related works. Section 5.2.3 describes the monitoring of phishing and legitimate hostnames in order to label the training dataset, and an analysis of the networks hosting the hostnames. Sections 5.2.4 and 5.2.5 explain the prediction features and the system architecture of the prediction model. Section 5.2.6 describes the experiments undertaken to build the model for evaluating the features. We then report performance results and present an analysis of the results. Discussions of the results is takes place in section 5.2.7. Section 5.3 describes the development of the ML model for predicting phishing hostnames hosted in NSIFNs. This has a similar structure to Section 5.2; we describe related works (section 5.3.1), the monitoring of name servers of phishing and legitimate hostnames, and the network analysis of NSIFNs (section 5.3.3). Sections 5.3.4 and 5.3.5 describes prediction features and the system architecture of the prediction model. Section 5.3.6 describes the experiments we performed to build the model and then presents the results. Section 5.3.7 presents discussions on the results. Section 5.4 summarizes the chapter.

5.2. Prediction of Phishing Hostnames Hosted in FFSNs

5.2.1. Related Works

Fast flux (FF) hostnames are hostnames of malicious web services hosted by FFSNs. FF phishing hostnames are FF hostnames hosting phishing websites. FF hostname detection approaches based on DNS, network and host related features can generally be categorized into two groups; those based on the monitoring of the features over an extended period of time and those based on the features extracted from data collected at one instant. Using the former approach, Passerini, et al. [129] developed the FLUXOR system based on a Naïve Bayes classifier that detects FF hostnames from data collected for at least three hours. The classifier used 9 features related to characteristics of hostnames and their networks queried from links of spam and non-spam emails. The classifier yielded zero FPR and FNR. Perdisci, et al. [131] passively monitored over 2.5 billion DNS queries per day for 45 days to cluster the queries based on related IP addresses. Using 12 passive and active features related to DNS answers, network, geo-location and up state of the host, a Decision Tree based model was developed to detect a cluster with the most FF hostnames, yielding a detection rate of 99.7%. For the model to work effectively, some of the features required data to be monitored for at least a day.

Kumar and Xu [253] proposed an SVM based classifier to detect FF hostnames using 7 DNS related features. The model was trained on passive DNS data to achieve an accuracy of 88.03%. Other important performance metrics including error rates were not reported. Some of the features required long term monitoring of hosts to obtain their values. For instance, the feature *MaxCount* counts the total number of visits to the domain over a particular period, typically 24 hours. Chen, et al. [254] proposed an LSTM based classifier to detect FF hostnames using three features from active hostnames. The classifier requires hostnames or CNAME and A records collected at five separate times as an input data. The technique achieved an accuracy of 95.39%. False alarm rates were not reported.

In order to shorten detection times in an attempt to achieve real-time detection, several works proposed methods based on detection features extracted at one point in time. The resulting detection times were reduced to a few seconds or minutes. Huang, et al. [255], for instance, proposed a classifier using 6 features based on time zones and geographical locations of websites' hosts and NSs. The study reported an accuracy, FPR and AUC of 98.16%, 0.398%

and 0.984 respectively. The classifier's detection time was 0.5 seconds. Hsu, et al. [137] proposed an SVM based classifier trained to detect FF hostnames using DNS traffic of known malware and legitimate domains. The classifier used 6 features based on network, processing and document fetch delays in the hosts. The key assumption used is that flux agents are standard computers in home networks thus have lower computing power and bandwidth than the hosts in non-FF legitimate networks, which often are servers. The classifier performed with an accuracy of 95% and AUC of 0.99. Wang, et al. [256] proposed a classifier to distinguish FF hostnames from benign hostnames using 6 features based on the number of IP addresses returned per hostname, the differences in time zones and the geographical distances between hosts of the same hostname. The classifier, based on a Bayesian network algorithm, achieved an accuracy of 96.7%, FPR of 4.4%, F1 score 0.971 and AUC of 0.982. Hsu, et al. [134] developed an FF hostname-detection classifier based on an FF score computed from the response time differences between hosts of the same hostname. The classifier achieved FPR and FNR of 0.3% and 2% respectively. Accuracy, precision and recall were not reported.

Lin, et al. [128] proposed an FF hostname-detection classifier based on a genetic algorithm. The classifier used four DNS and time related features to produce an accuracy of 98.24% and FPR of 1.78%. Detection time was found to be 18.54 seconds. A Random Forest based classifier built by Stevanovic, et al. [257] used 39 DNS, network, reputation related features to build a FF hostname detection method, achieving precision, recall and F1 scores of 0.90, 0.804 and 0.849 respectively. Jiang and Li [258] proposed an SVM based classifier that used 5 DNS, network, time and host geo-location related features to detect FF hostnames. The classifier produced an accuracy of 96.7%, precision of 0.965, recall of 0.981, F1 score of 0.973 and AUC of 0.989. Detection time was estimated at 400 seconds. Almomani [130] developed a Fast Flux Hunter system based on adaptive evolving fuzzy neural network-based classifier to detect FF hostnames. It used 11 and 3 DNS and time related features respectively. The classifier produced an accuracy of 98% and a range of error rates between 0% and 16% depending on the sample size of the testing dataset. Some of the works derived most or all of their features from other works. For instance, Martinez-Bea, et al. [259] adopted all 10 features from Passerini, et al. [129], Hsu, et al. [137] and Lin, et al. [128] to build an SVM based classifier. The classifier achieved an FPR of 1.23% and an FNR of 0%. The table in appendix VI summarizes significant works in this domain.

5.2.2. Limitations of Related Works

The approaches described above have the following limitations:

1. Some of the approaches require monitoring of the prediction features over an extended period of time in order to produce good prediction performance. For instance, the minimum time intervals in Passerini, et al. [129] and Kumar and Xu [253] were 3 and 24 hours respectively. However, as described in section 2.2.3.3, some of the phishing websites stay online for less than 24 hours, the time window in which they still manage to victimize many users. Therefore, during the monitoring phase, the monitored FF hostnames will continue to operate and cause significant damages.
2. They detect FF hostnames of all forms of malicious web services, including spam, malware and phishing websites. As studied by Caglayan, et al. [55], different types of FFSNs, depending on specific web services they host, are often designed differently thus have variations in some of their distinctive characteristics. For instance, the study observed that spam FFSNs have longer lifespans (between 30 and 90 days) while most of the phishing FFSNs live less than a week. Since the longer the FFSN exists, the more agents and hostnames are being recruited and hosted, spam FFSNs tend to have large network sizes as well as large numbers of hosted hostnames compared to phishing FFSNs. The study illustrated this difference by observing that the average number of hostnames hosted in spam FFSNs and phishing FFSNs were 71 and 19 respectively. Such differences in their structures result in differences in a number of characteristics that are often used in detecting the FF hostnames. The characteristics include network and geographical dispersion of hosts, and distribution of co-hosted websites. Therefore, solutions designed to detect all types of malicious web services hosted in generic FFSNs are likely to be less effective than the ones which detect specific services hosted in their dedicated FFSNs.
3. A number of approaches achieved low or moderate prediction performance. For instance, Kumar and Xu [253] obtained an accuracy of 88.03%, Stevanovic, et al. [257] achieved an F1 score of 0.85 and Almomani [130] attained misclassification rates of up to 16%.
4. Some of the approaches used feature sets with low diversity. For instance, Hsu, et al. [134] and Chen, et al. [254] used features belonging to one category only while Hsu, et al. [137], Lin, et al. [128] and Almomani [130] have used features from 2 categories.

Similar to the reason we pointed out in section 4.2.6, the use of features from a small number of categories increases the susceptibility of a solution to simple detection evasion mechanisms. For instance, Hsu, et al. [137]’s classifier can easily be bypassed by attackers recruiting and using as flux agents compromised servers that are as powerful as legitimate servers, leading to FF and non-FF hostnames having similar feature characteristics. In Stalmans, et al. [260] and Wang, et al. [256], the features used, which are all based on geolocation and time zone records of hosts, can be bypassed by attackers configuring the name server such that it returns IP addresses of flux agents closer to users in order to emulate the characteristics of CDNs, thus evading detection. Furthermore, attackers may command only agents from the same time zones and geolocations to serve users in certain regions, similar to how CDNs operate, to increase false alarms.

5. With an exception of Chen, et al. [254], all the studies addressed the problem of distinguishing FF hostnames from legitimate hostnames while ignoring non-FF malicious hostnames. However, according to our data (described in section 5.2.3.2), the majority of the malicious websites are still hosted by non-FF malicious hostnames. By not considering such hostnames, the current solutions are impractical as they do not reflect the real-world problem.
6. IP geolocation databases such as Maxmind¹³ and IP2location¹⁴ do not have geolocation and time zone records for all public IP addresses. Some of the works, including Huang, et al. [255], Stalmans, et al. [260] and Wang, et al. [256] have based their detection on features relying entirely on this data. Such techniques fail to detect websites whose hosts’ geo-IP data is not found in the databases.

In addition to these limitations, phishers continue to learn about the prediction features used by current solutions with the aim of subverting the solutions. To keep with the pace of attackers and be ahead of them, new features that will improve or maintain the detection effectiveness across different periods must be explored. To address these limitations, we base our solution for effective detection of FF phishing hostnames on the following design characteristics and goals:

¹³ https://dev.maxmind.com/geoip/geoip2/downloadable/#MaxMind_APIs

¹⁴ <https://www.ip2location.com/>

- It uses an ML approach in order to build accurate prediction rules from a high dimensional dataset we intend to use. The approach also provides for the possibility of improving the rules through data re-collection and re-training as the nature of attacks change.
- It focuses specifically on the detection of phishing FF hostnames in order to optimize the detection of the attacks.
- It reflects the real-world problem setting in which the input to the classifier is a mixture of legitimate and malicious hostnames of all sorts by distinguishing FF phishing hostnames from the other possible types (i.e non-FF phishing hostnames and legitimate hostnames). This will make the solution more practical.
- It monitors dynamic features over time in order to label the training dataset but uses prediction features that can be extracted instantly and evaluated by the prediction model in order to achieve a possible real time detection.
- It achieves high prediction accuracy and low misclassification rates.
- It uses highly diversified novel prediction features to make the solution more resistant to detection evasions.

Aiming to meet the design goals, this study investigates, proposes and evaluates a novel set of predictive features for accurate and fast detection of FF phishing hostnames using ML.

5.2.3. Monitoring and Analysis of Hostnames and Their Networks

5.2.3.1. Monitoring of Phishing and Legitimate Hostnames

In order to be able to label a training set of websites according to their fluxing behaviour, we collected 4,271 known and active phishing websites and 7,530 legitimate websites and monitored the IP addresses returned by DNS queries of their hostnames over an extended period. We obtained the legitimate URLs from a list of 1 million most visited websites maintained by Tranco¹⁵, and the phishing URLs from two major reputable online repositories namely PhishTank¹⁶ and OpenPhish¹⁷. The hostnames were then extracted from the URLs. For each hostname, a DNS A record query was sent to Google's public NS every 15 minutes for 5 weeks from 16th of June to 24th of July 2019. The IP addresses returned in consecutive queries

¹⁵ <https://tranco-list.eu>

¹⁶ <https://tranco-list.eu>

¹⁷ <https://openphish.com>

for the same hostname were compared and the number of times throughout the period that a change was observed was recorded. We adopted a threshold of number of changes of IP addresses used by Holz, et al. [132] to distinguish between FF and CDN hostnames whereby we labelled any phishing or legitimate hostname observed to have at least one change of IP address as a FF phishing or CDN hostname respectively. The phishing and legitimate hostnames observed without any IP change were labelled as phishing non-flux and legitimate non-flux hostname respectively. Figure 5.1 illustrates the monitoring of A records of URLs for class labelling.

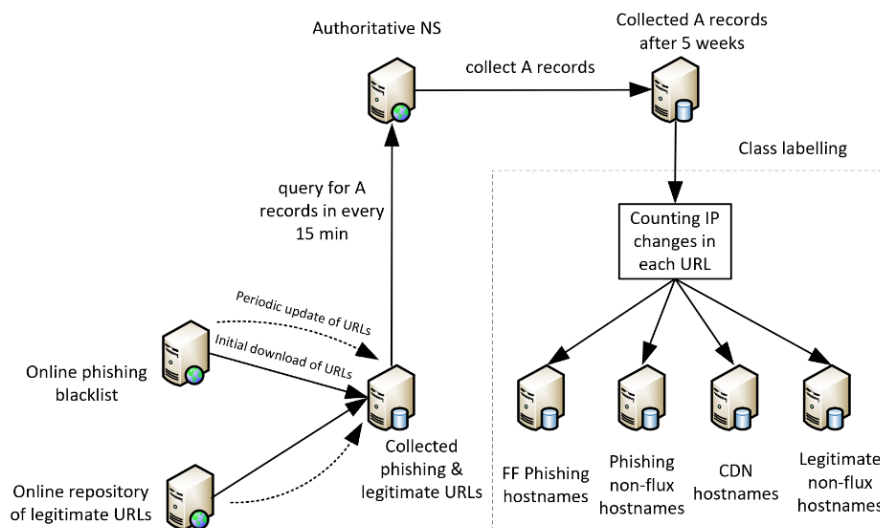


Figure 5.1. Monitoring of A records for 5 weeks for labelling classes of the hostnames.

5.2.3.2. Network Analysis of Flux and Non-Flux Networks

This section describes the analysis of some of the network characteristics of the hostnames observed based on the monitored A records from the previous section. This analysis gives an early insight on the distinctive structures and behaviours of the networks that can potentially be used for classification of hostnames they host. The behaviours can also be used by other researchers to further investigate the networks in order to extend the contributions made by this study. Furthermore, the analysis can be useful in determining whether the recent FFSNs have evolved in terms of their structures and operations compared to those appearing in previous studies.

Lifespan of Hostnames

During the monitoring process, we observed varying hostname lifespans. This is the period during which DNS returns A records for the hostname when queried. Figure 5.2 shows the lifespans of the phishing hostnames. 37% of the hostnames were active throughout the entire period (39 days). From this data, it is difficult to estimate the true lifespans of the hostnames from the first day they were registered as they could have existed longer time before we started monitoring them. The range of lifespan of most hostnames is much larger than those observed by other studies described in section 2.2.3.3. The reason for this could be that some of the collected hostnames correspond to the compromised legitimate sites and therefore have longer lifespans. Another reason could be that the attackers prolong the lifespans of some of the hostnames in order to avoid them appearing suspicious. As expected, the collected legitimate hostnames were active throughout the monitoring period.

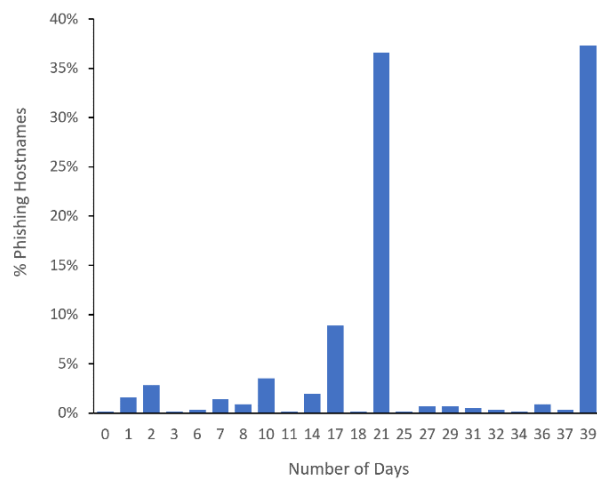


Figure 5.2. Lifespans of the collected phishing hostnames during the monitoring period.

Number of Changes of IP Address

Figure 5.3 shows the distribution of the number of changes of IP addresses of hosts observed for each of the monitored hostnames in the same period. Legitimate and phishing hostnames exhibit similar behaviours, with a high proportion of both types undergoing no change of IP address, but there is a long tail of hostnames with a high number of changes. Only 29% of the collected phishing URLs were observed to undergo at least one change and 17% of these underwent fewer than 10 changes. This suggests that most phishing websites are still hosted in

non-flux networks. At the other end of the scale, a small number of hostnames exhibited over 400 changes. 52% of the legitimate URLs experienced at least one change with 25% of these having fewer than 10 changes. Generally, legitimate URLs were observed to have larger numbers of changes than phishing ones. This may be because the legitimate URLs were obtained from a list of 1000 most visited websites, which are more likely to be hosted in CDNs than the lowly ranked websites [261]. Websites with a low but non-zero number of changes, for instance fewer than 5 changes in the monitoring period, may be due to activities such as routine maintenance of hosts and upgrading of networks.

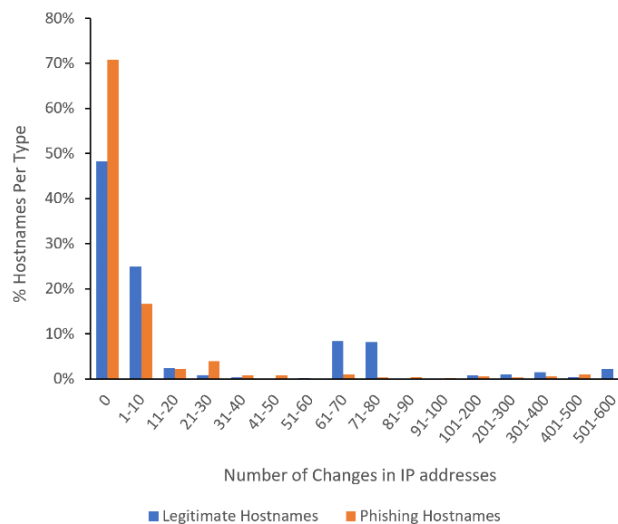


Figure 5.3. Distribution of number of changes of IP addresses of hosts observed per hostname type. Note that the data bins are not all of equal size.

Number of Cumulative Unique IP Addresses Over Time

Figure 5.4 shows the growth over time of the number of cumulative unique IP addresses of both FF phishing and CDN hostnames. Here, we observed the number of addresses after every 2 hours, instead of 15 minutes (as described in section 5.2.3.1), in order to observe only noticeable changes. This is because some hostnames were changing the addresses after few hours. The former increases rapidly at first but then the rate of growth decreases gradually over time until the curve becomes approximately linear. This suggests that FFSNs continue to acquire new agents over time. We stopped the graph at the 137th query because the pattern was interrupted by having a large number of phishing hostnames no longer active thus were not returning IP addresses. The CDN curve is less smooth, with short bursts of rapid growth

interleaved with longer intervals of gradual growth. From the 102th query, the CDN data maintained a nearly zero growth rate, indicating that the hostnames have exploited the finite pools of hosts of their networks after a certain number of queries. This observation concurs with our understanding that CDNs do not grow with time.

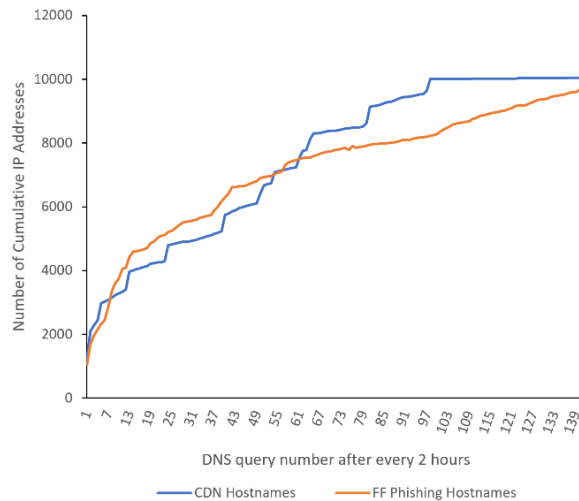


Figure 5.4. The number of cumulative unique IP addresses of FF phishing and CDN hostnames over the monitoring time.

Figures 5.5 and 5.6 show the distributions of number of changes of IP address versus number of unique IP address over the monitoring period for FF phishing and CDN hostnames respectively. Each vertical line represents a hostname. In the former distribution, the number of unique IP addresses tend to increase with the number of changes after the 15th mark. Five common numbers of IP addresses are observed as the flat steps in the graph, indicating that most of the hostnames use almost equal numbers of unique IP addresses within a small range of number of changes of IP addresses. It is likely that some of these hostnames are hosted in the same FFSNs or in different FFSNs which share a large pool of the same machines. The latter distribution is different whereby after the 37th number of changes, the hostnames use a small range of number of unique IP addresses but largely varying after the 77th number of changes. It is also likely that some of the hostnames between the 37th and 77th marks are hosted in the same CDNs sharing the same servers. However, it is clear that in both graphs the number of unique IP addresses is very low for hostnames which were observed with small changes.

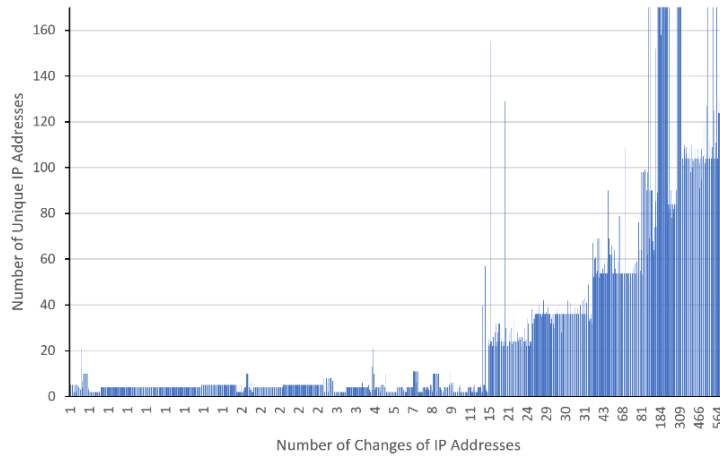


Figure 5.5. Distribution of number of changes of IP addresses versus number of unique IP addresses over the monitoring period for FF phishing hostnames.

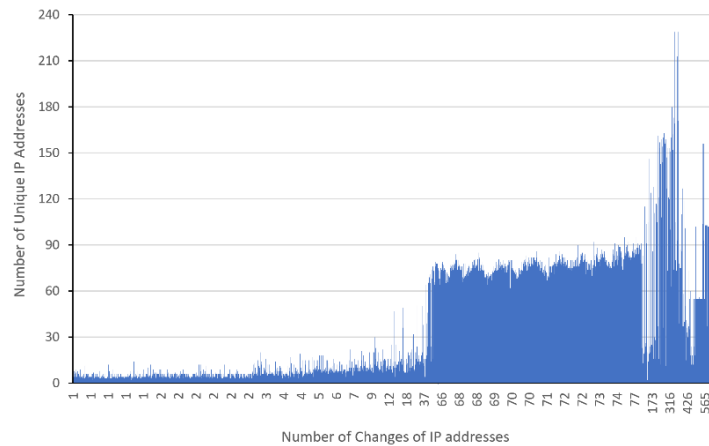


Figure 5.6. Distribution of number of changes of IP addresses versus number of unique IP addresses over the monitoring period for CDN hostnames.

Network Distribution of Hosts of Hostnames

Figure 5.7 shows the distribution of the number of unique subnets of hosts of CDN, legitimate non-flux FF phishing and phishing non-flux hostnames. Subnets are logical network segments of a large IP network. The hosts of most CDN hostnames are distributed over between 2 and 17 subnets, the most popular number being 3 subnets. The largest number of subnets observed is 32. The most common number of subnets amongst FF phishing hostnames, on the other hand, is 6. The phishing distribution is more spread out, with a long tail, although also more variable. Legitimate non-flux hostnames are distributed between 1 and 5 subnets while those of phishing are between 1 and 4, with most of both hostname types being hosted in one subnet. This is in

line with expectations that the hosts of FF phishing hostnames are located in more spread subnets than those of CDN and non-flux hostnames.

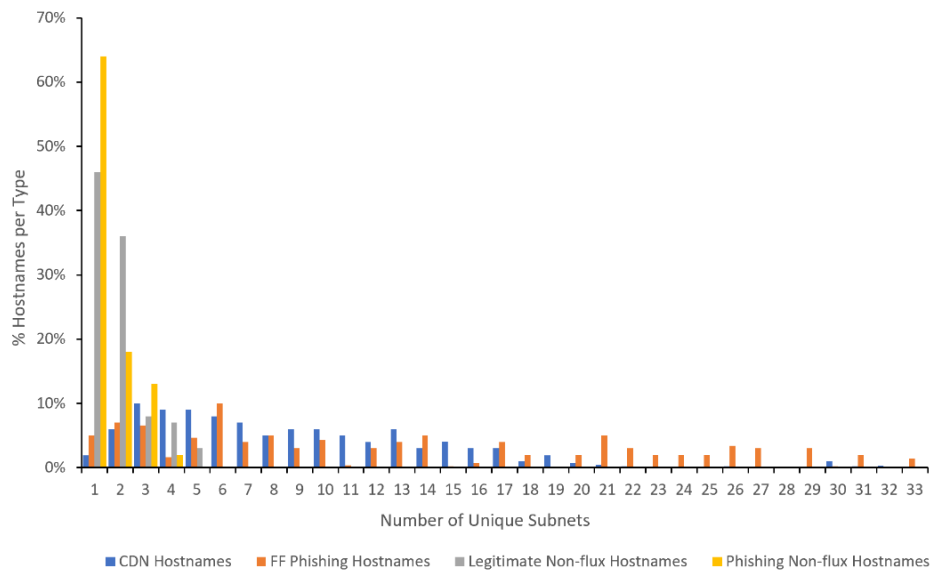


Figure 5.7. Distribution of unique number of subnets of flux and non-flux hostnames.

Similar patterns are observed for the distributions of unique networks and Autonomous Systems (ASs) of the hosts as shown in Figures 5.8 and 5.9 respectively. In the former, the majority of CDN hostnames are hosted between 1 and 8 networks, with the largest number of networks observed was 28. In contrast, FF phishing hostnames are hosted in a wider range of numbers of networks. In both cases, the largest percentage of hostnames are hosted in 2 networks. Legitimate non-flux hostnames, however, are hosted between 1 and 6 networks while the phishing ones are between 1 and 5 but most of them are hosted in one subnet.

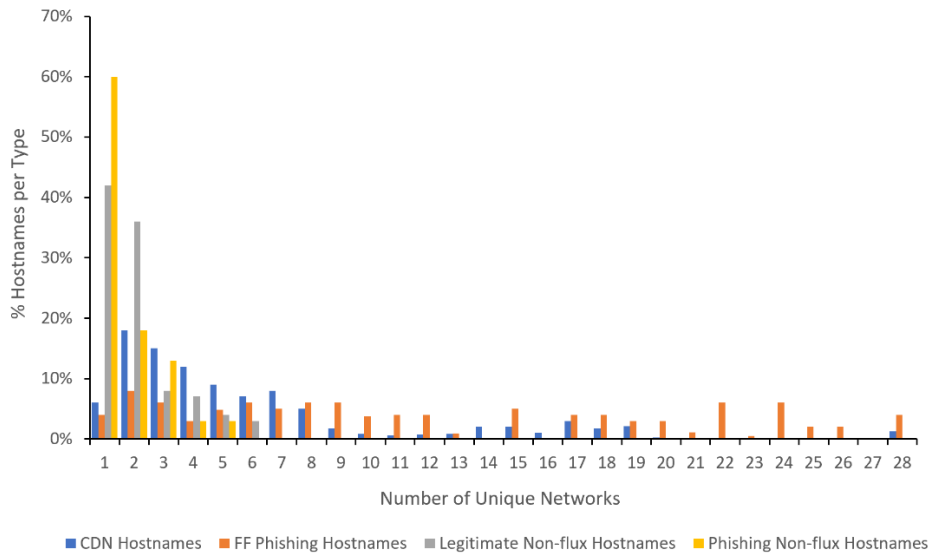


Figure 5.8. Distribution of number of unique networks of flux and non-flux hostnames.

For the case of ASs, the majority of all types of hostnames are hosted in 1 and 2 ASs while significant percentages of FF phishing hostnames are hosted in up to 15 unique ASs. A small percentage of CDN hostnames are hosted in 10 ASs but the rest are hosted in less than 7 ASs. All phishing and legitimate non-flux hostnames are hosted between 1 and 3 and between 1 and 4 ASs respectively, with one AS being the most popular for most of them. The observed wider network distribution of FF phishing hostnames compared to the others in this study is similar to the observations made by other studies as discussed in section 2.5.

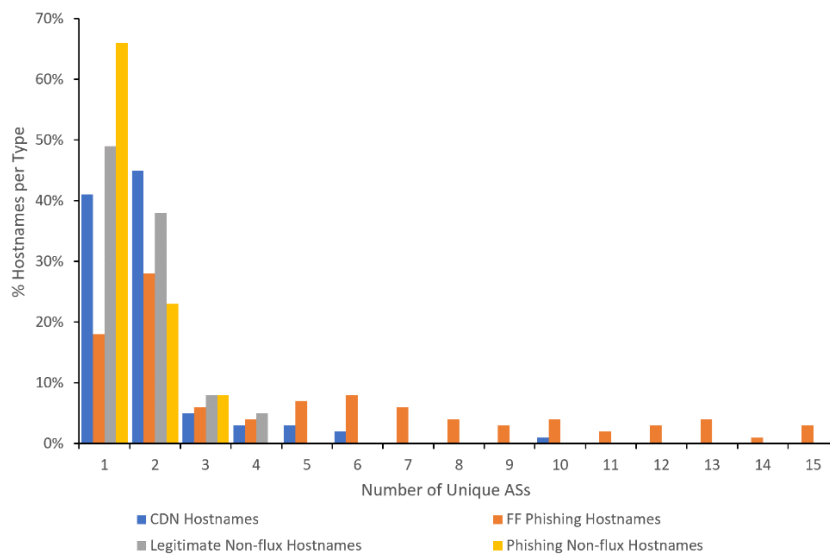


Figure 5.9. Distribution of number of unique ASs of flux and non-flux hostnames.

Figures 5.10 and 5.11 show top organizations managing the ASs which were observed to host servers of the phishing and legitimate hostnames respectively. Hosts of most phishing hostnames were found in ASs managed by Cloudflare and Incapsula, which are among the largest CDN providers in the market. Hosts of most CDN hostnames we collected appear to be hosted by the former while hosts of legitimate non-flux hostnames were fairly distributed among a larger number of organizations. In total, we found 34 organizations with their ASs hosting servers of phishing hostnames compared to 273 organizations hosting servers of legitimate hostnames. This indicates that only few organizations are targeted by attackers for hosting their services. Cloudflare, as observed by other researches, has been a leading platform of choice for attackers to host their servers especially those of botnets. This is because they do not have a strict policy of monitoring contents hosted in their platforms and takedown those which host malicious contents [262]. Beside their little efforts in battling abusers of their services, we think CDN providers have been a common target for hosting the phishing services also because of their widely distributed networks across different parts of the globe, in which attackers benefit from their high availability and performance of services to various locations.

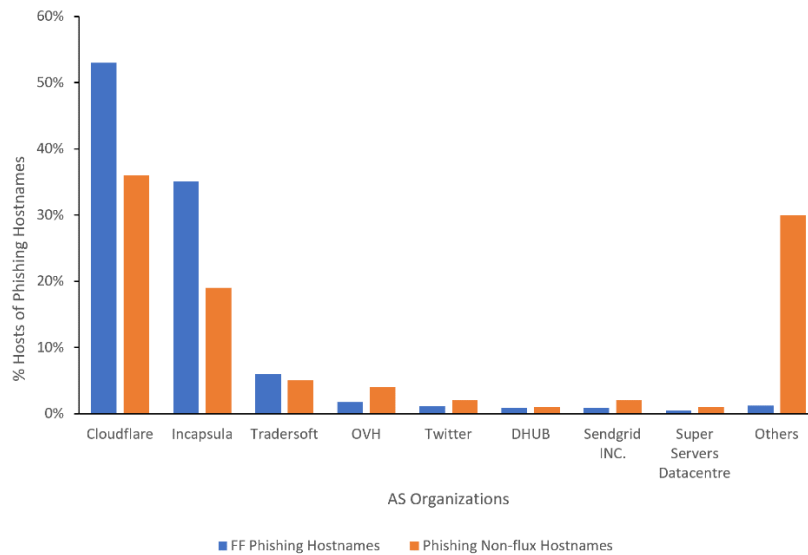


Figure 5.10. Top AS organizations hosting servers of phishing hostnames.

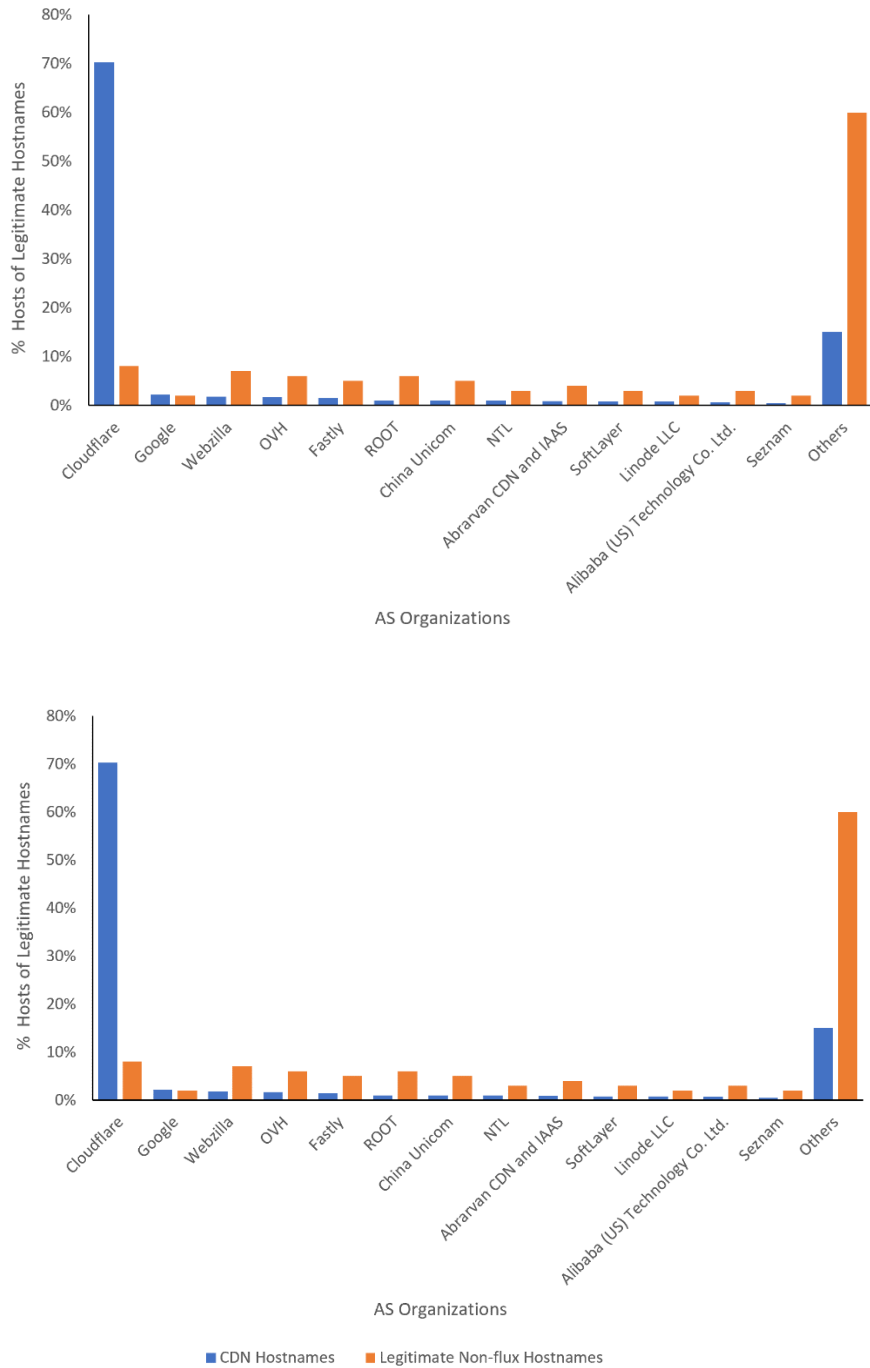


Figure 5.11. Top AS organizations hosting servers of legitimate hostnames.

Hosting Countries of Hostnames

We also observed the distributions of number of unique countries hosting the hostnames (Figure 5.12). The distributions show that hosts of most hostnames are located in one country with the next large percentage are located in two countries. However, a significant percentage of phishing hostnames are hosted in more countries (up to 8) compared to 6 and 3 of CDN and

non-flux hostnames respectively. Figures 5.13-14 show the distributions of top countries hosting the hostnames. The top countries hosting CDN hostnames are also among the hosts of the largest number of data centres [263] and the home of most of the largest CDN providers [264]. A number of countries in this list are also among those with the highest number of internet users and the largest e-commerce markets [265, 266], and those with the largest number of organizations targeted with phishing attacks [23]. In the latter, the list consists of mostly the same countries in addition to Russia, Lithuania and Ukraine. This suggests that some of the attackers host their phishing websites within the same countries of their targets while others host the websites in other countries different from the countries hosting their targets.

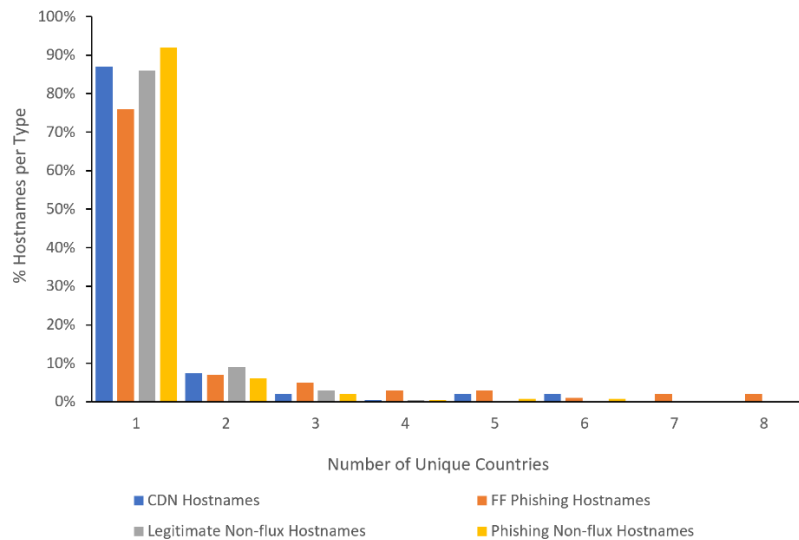


Figure 5.12. Distribution of number of unique countries of the observed hostnames.

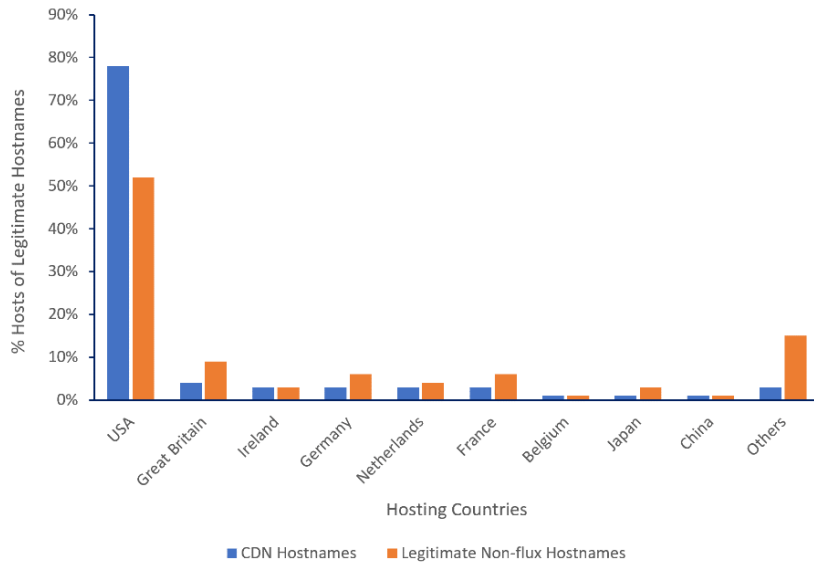


Figure 5.13. Top countries hosting servers of the legitimate hostnames.

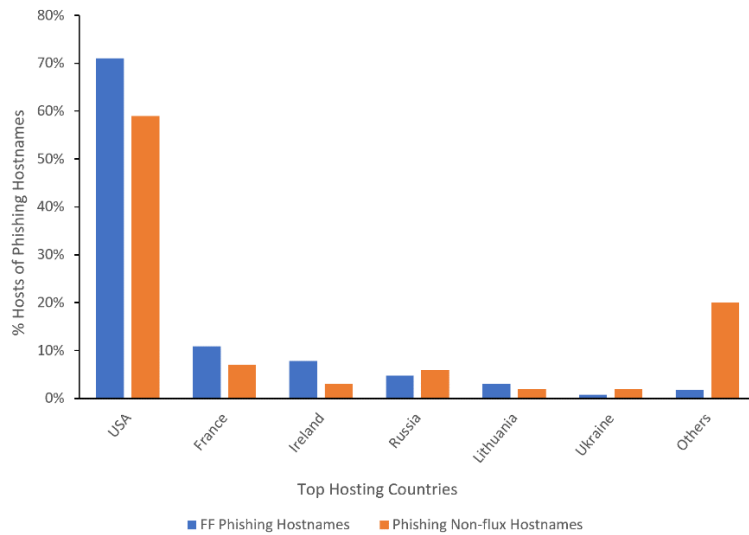


Figure 5.14. Top countries hosting servers of the phishing hostnames.

Network Graph Analysis of FFSNs

We also performed network analysis using graph analysis software (Gephi) to understand the underlying component connectivity of phishing FFSNs. To perform the analysis, the cumulative IP addresses of each FF phishing hostname during the period of the study were imported into Gephi to obtain directed graphs in which the hostnames and IP addresses¹⁸ are

¹⁸ The source of an edge is thus always a hostname and the destination is always an IP address.

nodes. An edge connecting a hostname to an IP address indicates that at least one DNS query for that hostname returned that IP address within the response¹⁹.

The overall graph of phishing FFSNs consists of 9,295 nodes, of which 6,664 (72%) are IP addresses and 2,631 (28%) are hostnames, and 16,114 edges. Thus, on average, each hostname is linked to around 6 IP addresses and each IP address is used by around 1.4 hostnames. The dataset contains 903 disjoint networks, a few of which are very large, but most of which are very small (see Figure 5.15). The largest network has 2276 nodes (24.5% of the total) and 43% of the nodes belong to networks of size less than 50. Figure 5.16 is similar but shows the absolute sizes of the 25 largest networks, with the columns split to indicate the numbers of hostname and IP address nodes. It may be seen that the majority of networks are dominated numerically by IP nodes but a significant minority (e.g. networks 6, 166, 388 and 149) are dominated by hostname nodes. About 43% of the nodes belong to the two largest networks and about the same number belongs to networks of size less than 60. A third of nodes reside in networks of size less than 8.

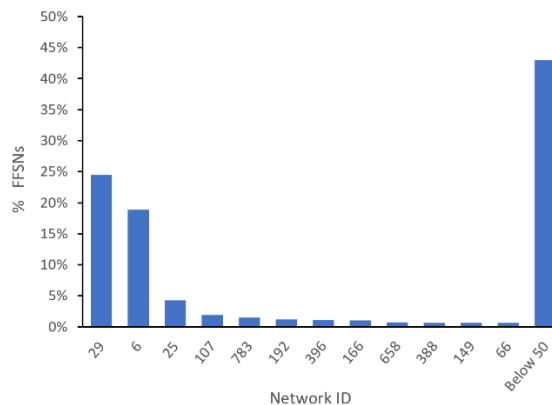


Figure 5.15. Sizes of 12 largest phishing FFSNs as a percentage of the total. Network IDs were allocated by Gephi.

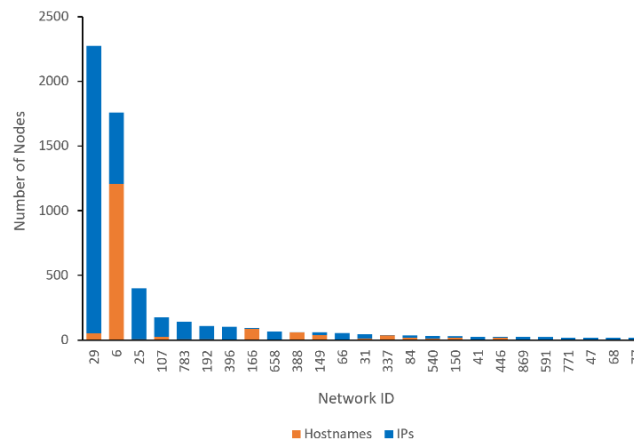


Figure 5.16. Absolute sizes of 25 largest networks. Columns are split based on node type.

Figure 5.17 visualises structure of the largest FFSN (with network ID 29) with 2,275 nodes of which 51 are hostnames and 2,224 are IP addresses. The hostname nodes are shown in green and the IP address nodes are in pink, and the node size indicates its degree (i.e. the number of connections to other nodes). The figure gives the impression of clouds of IP addresses clustered

¹⁹ DNS queries can, and frequently do, return more than one IP address.

around hostnames, with many IP addresses being connected to only a single hostname. Individual clusters are connected by through the IP addresses they have in common. The distribution of the degrees of hostnames is shown in Figure 5.18. This is relatively flat if one ignores the most highly-connected nodes, but with two or three plateaus. In contrast, the vast majority of IP nodes have degree 1, and the numbers having higher degrees drops off rapidly (see Figure 5.19). The most highly connected IP address is of degree 12. This high ratio of IP addresses to hostnames is what one would expect from an FFSN. It is not clear at this stage whether network 29 is a single ultra-large FFSN or a collection of smaller (but still large) FFSNs run by different criminal groups that have similar strategies for recruiting bots, so that some IP address have been infected multiple times. Analysis of the domains to which the hostnames belong and of the distribution of IP addresses in each hostname-centred cluster may give an indication of which is more likely.

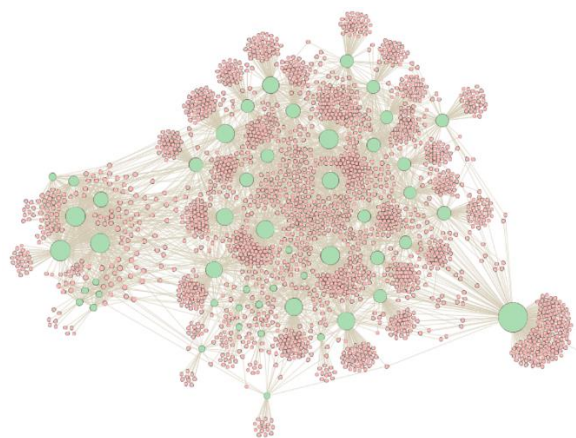


Figure 5.17. Visual graph of the largest FFSN with 2275 nodes, of which 51 are hostnames and 2224 are IP addresses.

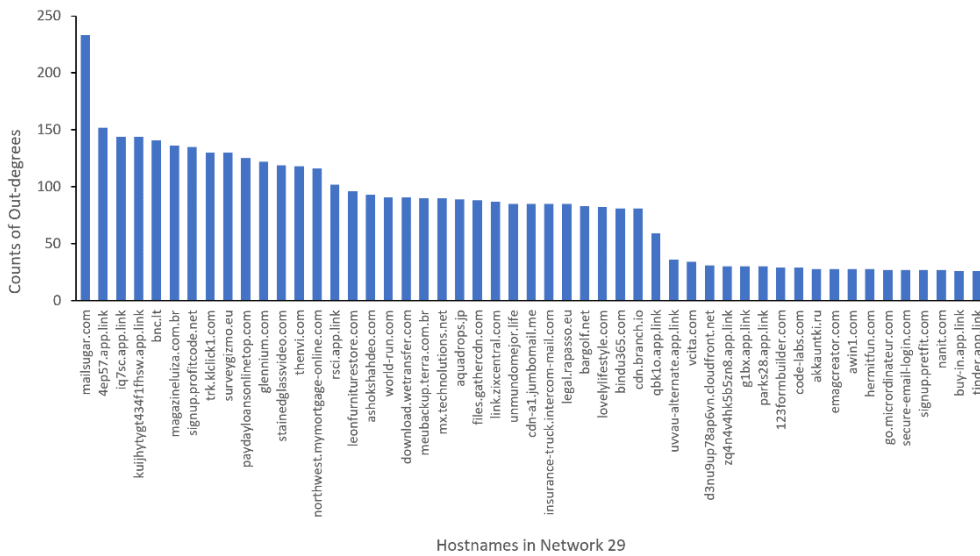


Figure 5.18. Hostnames in network 29 ordered by out-degree.

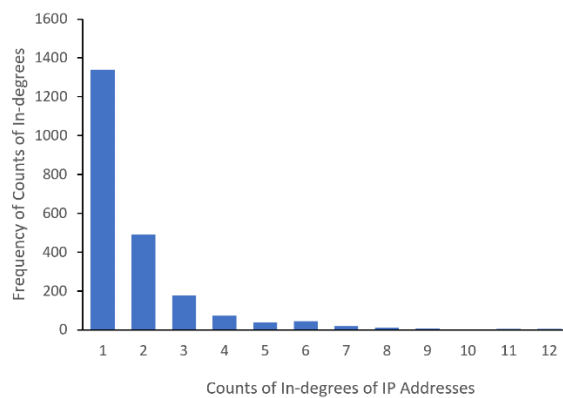


Figure 5.19. Frequencies of in-degree counts of IP addresses in network 29.

In summary, this section has highlighted some of the structural characteristics of the networks hosting the hostnames whose A records were monitored. Most characteristics described here are in line with those observed by other studies as discussed in sections 2.2.3 and 2.3.3. The only exception is the lifespan of FF phishing hostnames, which in this case is longer for most hostnames than the one that was observed by other studies. This shows that phishing networks especially FFSNs have not changed much over the years. The analysis has also indicated the following key differences between the hostnames:

- The lifespan of phishing hostnames is shorter than those of legitimate hostnames.
- The hosts of some of the FF phishing hostnames are widely distributed in terms of subnets, networks and ASs than the other hostname types.

- The hosts of some of the FF phishing hostnames are hosted in a larger number of unique countries compared to other hostnames.

This suggest that these characteristics can be useful in distinguishing FF phishing hostnames from the rest. In section 5.2.6.6, we further provide an analysis of some of the important distinctive characteristics of the four types of hostnames using the feature data extracted in section 5.2.6.3.

5.2.4. Features for Prediction of FF Phishing Hostnames

Based on the distinctive characteristics of FFSNs, CDNs, phishing non-flux networks and legitimate non-flux networks described in sections 2.3.3 and 5.2.3.2, we derived various novel features for distinguishing FF phishing hostnames from CDN, phishing non-flux and legitimate non-flux hostnames. We also studied features used by previous works addressing the same problem and identified those which can be extended or improved to derive other new features. Also, we adopted features that were used and defined as strong predictors in several works in our proposed set of features. In total, we identified 83 potential features grouped in five categories; temporal, spatial, DNS, network, host and hostname reputation. Of the 83 features, 62 are newly introduced by this study and 21 features are adopted from previous works. In this section, we describe some of the features which were observed to be among the best features for this problem. A complete list of the features is given in Table 5.1. Table 5.2 provides a list of sources of information we used to generate the features.

Feature Category	Feature #	Features	Novel or Adopted Features	
Temporal	1 - 5	Round trip time: average, standard deviation, entropy, minimum and maximum	1 - 2	Adopted
			3 - 5	Novel
	6	DNS response time	6	Novel
	7	TTL for A records	7	Novel
	8 - 10	Uptimes of hosts: average, standard deviation and entropy	8	Novel
			9 - 10	Novel
	11 - 12	Domain age, domain validity	11	Adopted
			12	Novel

	13 - 15	Domain ages of co-hosted websites: average, standard deviation and entropy	13 - 15	Novel
Spatial	16 - 18	# hops on route to host: average, standard deviation and entropy	16 - 18	Novel
	19 - 21	# unique hop countries on route: average, standard deviation and entropy	19 - 21	Novel
	22	Ratio of hosts in the same country with a user	22	Novel
	23 - 24	# unique hosts' countries, # unique of hosts' continents	23	Adopted
			24	Novel
	25 - 27	# unique hops' continents: average, standard deviation and entropy	25 - 27	Novel
	28 - 30	Geo-distances between the user and hosts: average, standard deviation and entropy	28 - 30	Novel
	31 - 34	Geo-distance between the hosts: sum, average, standard deviation and entropy	31 - 34	Novel
35 - 38	IP range between hosts: minimum, maximum, average, standard deviation	35 - 38	Novel	
DNS	39	# unique A records	39	Novel
	40 - 43	Ratio of hosts: with PTR records, with their PTRs containing IP addresses, with PTR's hostnames matching with the URL's hostname, with PTR's hostname identity matching with the URL's hostname identity	40 - 43	Novel
	44 - 47	Average: length of hosts' PTRs, # of digits in hosts' PTRs, # of hyphens in hosts' PTRs, # of dots in hosts' PTRs	44 - 47	Novel
	48	# unique TLDs of hosts' PTRs	48	Novel
	49 - 52	Ratio of hosts: with co-hosted websites, with co-hosted websites' hostnames matching with the URL's hostname, using private IP addresses, with dynamic IP addresses	49 - 51	Novel
			52	Adopted
	53 - 54	# co-hosted websites: average, standard deviation	53 - 54	Novel
	55	# unique hostnames of co-hosted websites in the hosts	55	Novel

	56	# unique TLDs of hostnames of co-hosted websites in the hosts	56	Novel
	57 - 58	Difference of average KL divergence between hostnames of non-FF phishing and FF phishing hostnames: KL of unigram characters, KL of bigram characters	57 - 58	Adopted
	59 - 60	Difference of average Jaccard Index between hostnames of non-FF phishing and FF phishing hostnames: JI of unigram characters, JI of bigram characters	59 - 60	Adopted
	61	Difference of average Levenshtein's Edit distance (ED) between hostnames of non-FF phishing and FF phishing hostnames	61	Adopted
Network	62 - 63	Subnets of hosts: unique #, entropy of # of subnets	62 - 63	Adopted
	64 - 65	Networks of hosts: unique #, entropy of # of networks	64 - 65	Adopted
	66 - 67	ASNs of hosts: unique #, entropy of # of ASNs	66	Adopted
			67	Novel
	68 - 69	Organizations managing hosts' ASs: unique #, entropy of # of organizations	68 - 69	Adopted
70	Ratio of hosts' networks with generic gateways	70	Novel	
Host	71	Ratio of available (up) hosts	71	Novel
	72 - 73	OS of hosts: unique #, common OS	72 - 73	Novel
	74 - 75	Webserver software of hosts: unique #, common webserver software	74 - 75	Novel
	76	Ratio of hosts with known proxy IP addresses	76	Novel
Reputation	77 - 79	Hosts' IP addresses in a blacklist of phishing IP addresses: total # of occurrences, average #, ratio of hosts with their IP addresses matched	77 - 79	Novel
	80 - 82	Hosts' IP addresses in a blacklist of IP addresses of name servers of phishing websites: total # of occurrences, average #, ratio of hosts with their IP addresses matched	80 - 82	Novel
	83	Domain registrar	83	Adopted

Table 5.1. A list of the proposed features for predicting phishing FF hostnames.

Feature #	Source/Tool
1 – 5, 8 – 10, 16 – 18, 70 - 73	Network queries with traceroute and Nmap commands
6 – 7, 36 – 39, 40 – 49	Authoritative nameserver
11 – 15, 83	WHOIS database
13 – 15, 50 – 51, 54 - 56	Bing Search engine
19 – 35, 62 – 69, 76	IP geolocation database
51	A list of private IP addresses provided by Internet Assigned Numbers Authority (IANA)
52	Dynamic User List (DUL) block list
57 - 61	Phishing blacklist (PhishTank, OpenPhish) and Tranco’s list of top ranked websites
74 - 75	HTTP response header
77 - 82	Phishing blacklist (PhishTank)

Table 5.2. Sources of data for each feature.

Many of the features measure the distribution of various attributes across a set of IP addresses identifying the hosts associated with a given hostname. Consequently, the first step in extracting the features is to perform a DNS query to obtain this set. The feature value is then obtained for each hostname and statistical measures such as average, standard deviation, entropy, minimum and maximum are calculated. Entropy features are approximated as $-\sum_i p_i \ln(p_i)$ where i runs over the set of unique values of the feature within the sample and p_i is the number of occurrences of i within the sample divided by the number of hosts. Entropy is greatest when all the feature values are different and least when they are all the same.

5.2.4.1. Temporal Features

Round Trip Time (RTT). Using the traceroute command, we measure the time interval between sending a data packet and receiving an acknowledgement from each host corresponding to the hostname in question. Five metrics are computed from these RTT values as feature 1 to 5 (see Table 5.1).

DNS Response Time. This is the time taken to receive an answer to a query for A records from the DNS server.

Authoritative TTL for A records. This is a maximum time set in the authoritative name server for caching A records of each hostname. It is obtained by querying A records of the hostname from its authoritative nameserver.

Uptime of Hosts. This is an estimated time, in hours, the machine has been up and running since the last time it was switched off. The measured time is retrieved from the host by a host scanning tool (Nmap). We extract features 8–10 from the uptime recorded for each IP address associated with the hostname.

Domain Age and Domain Validity. We query a WHOIS database to extract a date of first registration and an expiry date of the hostname’s domain. The former is used to compute its age with respect to the current date while the latter is used to compare with the current date to determine whether the domain has expired or is still valid.

5.2.4.2. Spatial Features

Geographical and Network Distances. We use the traceroute command to obtain the IP addresses of intermediate hops on the route to each host of the given hostname. The hop IP addresses are then used to generate features 16-21 and 25-28. For instance, we identify the country location of each hop and count the number of unique countries on the route to a host. By combining the numbers of all the hosts per each hostname, we derive features 19–21. Using geographical coordinates of IP addresses of a user and the hosts, various geographical distances are computed to obtain features 29–35. We obtain the coordinates from Maxmind’s Geolite2 database²⁰.

5.2.4.3. DNS Features

Characteristics of Hosts’ PTR Records. PTR records hold information allowing DNS to perform an inverse look-up of a hostname given an IP address. A DNS PTR query is performed for each host of the same hostname to obtain its PTR records. Features 41–49 are then extracted from the records.

²⁰ <https://dev.maxmind.com/geoip/geoip2/geolite2/>

Characteristics of Co-hosted Websites. We search for websites that are co-hosted on each IP address of the hostname using a Bing search engine with a search command ‘ip:W.X.Y.Z.’. From the search results, we count the number of co-hosted websites and extract their URLs from which we generate features 50-57.

Similarity of Hostnames. Similar to the approaches used by Yadav, et al. [267] and Fu, et al. [268], we measure similarity of string/character composition of all types of hostname to the FF phishing hostnames. Three similarity measures are used namely Kullback-Leibler distance (KL) [269], Jaccard Index (JI) [270] and Levenshtein’s Edit Distance (ED) [271]. For KL and JI, the similarity is measured for unigrams and bigrams of the hostnames. The symmetric values of KL are computed rather than asymmetric ones. For example, the JI of unigrams between hostnames h_1 and h_2 is defined as

$$JI(h_1, h_2) = \frac{|X \cap Y|}{|X \cup Y|} \quad 22$$

where X and Y are the sets of unigrams of h_1 and h_2 . JI is small if h_1 and h_2 are very different and approaches 1 if they are similar. For classifier B (described in section 5.2.6.5) as an example, we calculate as a feature

$$\Delta JI = \frac{1}{|O|} \sum_{o \in O} JI(h, o) - \frac{1}{|P|} \sum_{p \in P} JI(h, p) \quad 23$$

the difference between the average over the JI scores of a hostname, h , tested against the combined set of non-FF phishing hostnames (O) and the average JI scores tested against the FF phishing hostnames (P). A small difference means that the hostname is more similar to the FF phishing hostnames than to the others. A similar approach is used to compute the differences for KL and ED to obtain features 57-61.

5.2.4.4. Network Features

Network Characteristics. For each IP address associated with the hostname, we extract its network identity information including subnet, network and Autonomous System Number (ASN) from an IP geolocation database (we use IP2location²¹) to compute features 62 to 69. For instance, for feature 62-63, the subnet of each host is identified and then we count the number of unique subnets and also compute the entropy of all subnets per hostname.

²¹ <https://lite.ip2location.com>

5.2.4.5. Host Features

Up State of Hosts. Using a host scanning tool (Nmap), we scan each machine hosting a given hostname to determine its availability state. We then compute a ratio of hosts in the ‘up’ state as feature 71.

Host’s Operating System. Using Nmap, we scan each host of a given hostname and identify its operating system (OS). We count the number of unique OSs and identify the most common OS for the hostname as features 72 and 73 respectively.

Host’s Webserver Software. We extract the name of the webserver software installed on each host of a given website from a response to an HTTP header request. From this we generate features 74 and 75.

Hosts with Proxy IP Addresses. We compute the proportion of the IP addresses of each host of a given hostname that are found in a database of known public proxy IP addresses (we use IP2Proxy²² database) as feature 76.

5.2.4.6. Reputation Features

IP Addresses Shared with Other Malicious Hostnames. We identify the IP addresses associated with a given hostname that appear in a blacklist of phishing URLs collected in the past three months to generate features 77-82. For instance, in feature 77, we count the total number of times the IP addresses of all hosts for each hostname that have matched in the database. Similarly, we query for NS records of each phishing website in the blacklist to generate a list of IP addresses of nameservers and match these against IP addresses of a hostname’s nameservers to generate features 80–82.

Domain Registrar. We identify the registrar of the website’s domain by querying against a WHOIS database to obtain feature 83.

²² <https://lite.ip2location.com/ip2proxy-lite>

5.2.5. System Architecture of the Prediction Model

Our proposed model uses a supervised ML approach to train and develop a classifier that predicts FF phishing hostnames. The process of building the classifier and predicting a new hostname consists of the following tasks (shown as steps 1 to 10 in Figure 5.20);

Step 1 - 2: Monitoring of the A records returned by DNS for sets of known phishing and legitimate websites in order to label their hostname classes according to their fluxing behaviour as described in section 5.2.3.1.

Step 3: Extraction of feature data. For each labelled hostname (from step 2 above), a range of services are queried and the features (described in section 5.2.4) are extracted from the returned information at once to generate the training dataset.

Step 4 - 6: Training an ML algorithm on the training dataset to develop a classifier.

Step 7 - 10: Predicting a new hostname. First, a requested webpage is loaded and checked whether it is a PDC webpage or not (step 8). The same process as in the previous model is performed here (see section 4.4.2). This process will ensure only hostnames of webpages that pose a potential phishing threat are analysed in order to avoid degrading of a user's browsing experience when accessing non-PDC webpages. Next, a hostname of the PDC webpage is retrieved, its features are extracted and then fed into the classifier, which outputs the hostname's predicted class.

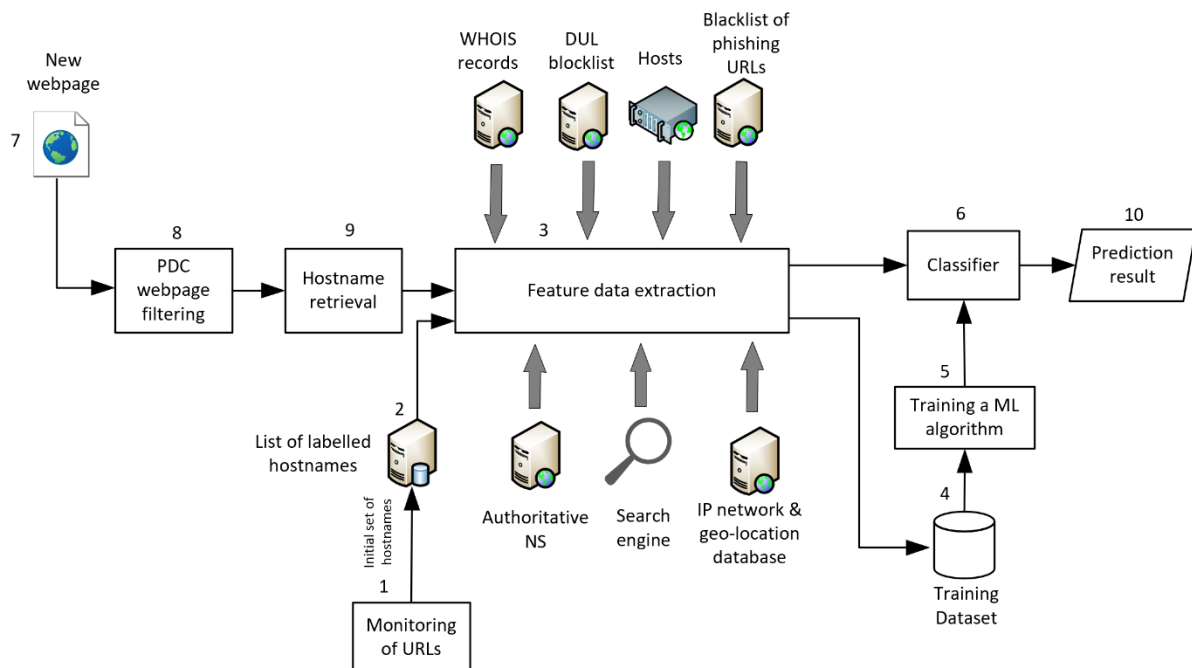


Figure 5.20. A system architecture of our proposed model for the prediction of FF phishing hostnames.

5.2.6. Experiments

We designed experiments to build an ML model for predicting FF phishing hostnames hosted in FFSNs. First, we designed four architectures for the model, based on flat and hierarchical classifications, and identified binary and multi-class classifiers building each architecture using the features described in section 5.2.4. We then ran two sets of experiments to evaluate the prediction performance of each classifier for each architecture. The first and the second experiments used eight traditional ML and three DL algorithms, respectively, the same algorithm sets as in Chapter 4 (see section 4.5). The best set of features for the prediction task was determined for each classifier. The performances of all classifiers in each architecture were determined and then combined to obtain the overall performance of each architecture in predicting FF phishing hostnames. Finally, we compared performances of the architectures to determine the best performing architecture for the model. The same eight standard ML performance metrics used in Chapter 4 were also used here to evaluate the classifiers and the architectures.

5.2.6.1. Experimental Setup

The same experimental setup used for experiments in Chapter 4 were used to run the ML experiments for this study. In this study, however, a different Python application was created and used to extract and pre-process data, create a training dataset, and train and evaluate the ML algorithms.

5.2.6.2. Flat and Hierarchical Classification Architectures

We defined four architectures, two flat and two hierarchical (see Figures 5.21). Table 5.3 summarizes only the classifiers that are useful for the prediction of FF phishing hostnames in the four architectures. Architectures A and B take the flat classification approach (Figures 5.21a and 5.21b respectively). Architecture A performs multi-class classification whereas the binary classification-based architecture B distinguishes FF phishing hostnames from the other three classes combined. Architectures C and D (Figures 5.21c and 5.21d respectively) apply the hierarchical classification approach in which a multi-class classification task is performed through two layers of binary classification tasks. In each hierarchical architecture, the LCPN technique is employed, in which a distinct classifier is used at each node. A selective classifier

approach was applied to identify the best classifier to use at each node. LCPN and the selective classifier approach were explained in detail in section 3.3.2. Feature selection (for traditional ML algorithms) was performed separately at each node to determine the most relevant features for each classifier. In evaluating the hierarchical architectures, we combined performances of the classifiers, from the parent node of the hierarchy to the leaf nodes consisting of the FF phishing hostname class, to obtain the overall performance. We then compare the performance of all architectures to determine the best performing architecture as a recommendation for implementing the model.

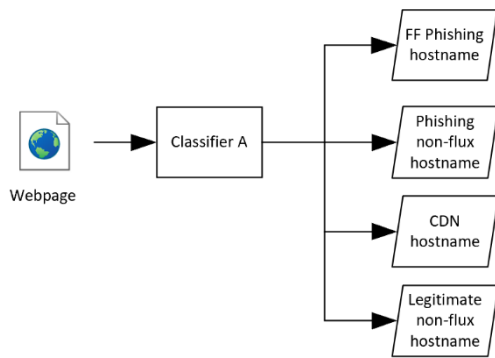


Figure 5.21a. Architecture A - flat multi-class classification.

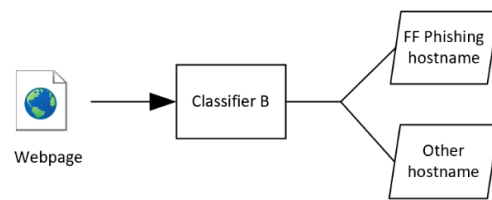


Figure 5.21b. Architecture B - flat binary classification.

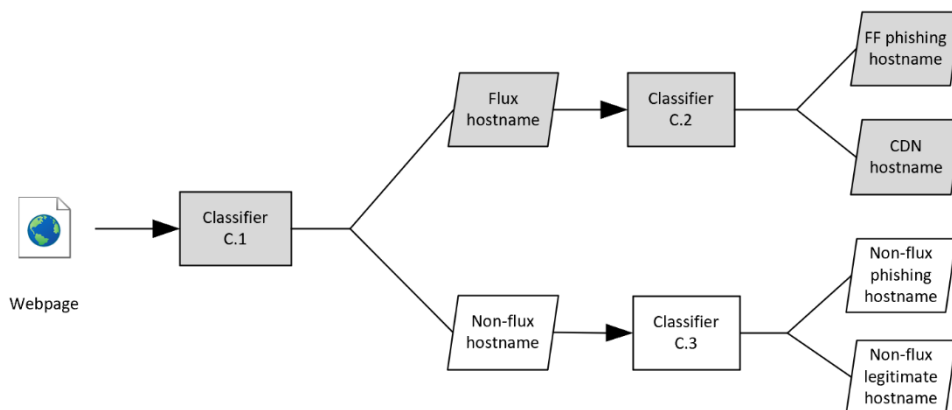


Figure 5.21c. Architecture C – hierarchical classification.

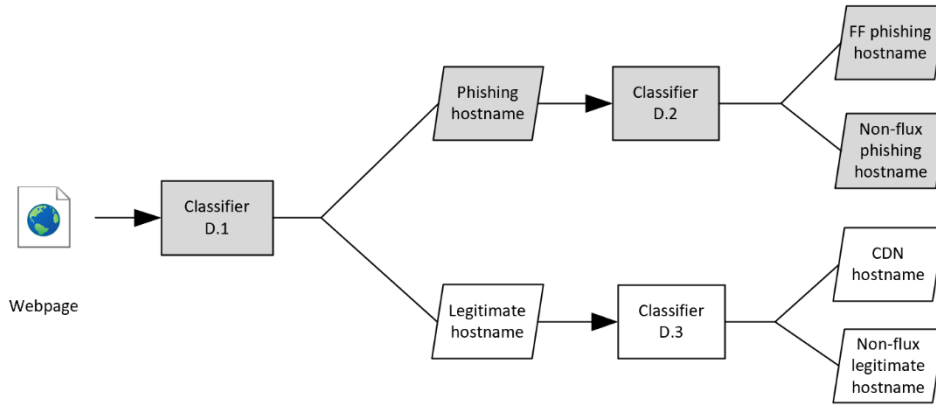


Figure 5.21d. Architecture D - hierarchical classification.

Classifier	Classifier Description	Classification Type
Classifier A	Allocates a hostname to one of four classes: FF phishing hostname, phishing non-flux hostname, CDN hostname and legitimate non-flux hostname	Multi-class classification
Classifier B	Classifies a hostname as FF phishing or other (phishing non-flux, CDN and legitimate non-flux combined)	Binary classification
Classifier C.1	Classifies a hostname as flux or non-flux	Multi-class classification
Classifier C.2	Classifies a hostname as FF phishing or CDN	
Classifier D.1	Classifies a hostname as phishing or legitimate	Multi-class classification
Classifier D.2	Classifies a hostname as FF phishing or phishing non-flux	

Table 5.3. Classifiers that are useful for the prediction of FF phishing hostnames in each architecture.

5.2.6.3. Training Dataset

The features described in section 5.2.4 were extracted from all the monitored and labelled URLs (described in section 5.2.3.1) to create an overall training dataset from which specific training datasets for each classifier were derived. As indicated in Table 5.4, the training datasets of classifiers A, B, C.1 and D.1 were formed by labelling the entire dataset with their respective classes whereas in classifier C.2, all hostnames with no changes in IP addresses were removed from the dataset. In classifier D.2, we removed all legitimate hostnames from the dataset.

Classifier	Hostname Class Labels	Class Size	Dataset Size
Classifier A	FF Phishing hostname	1257	11801
	Phishing non-flux hostname	3014	
	CDN hostname	3867	
	Legitimate non-flux hostname	3663	
Classifier B	FF Phishing hostname	1257	11801
	Other hostnames	10544	
Classifier C.1	Flux hostname	5124	11801
	Non-flux hostname	6677	
Classifier C.2	FF Phishing hostname	1257	5124
	CDN hostname	3867	
Classifier D.1	Phishing webpage	4271	11801
	Legitimate webpage	7530	
Classifier D.2	FF Phishing hostname	1257	4271
	Phishing non-flux hostname	3014	

Table 5.4. Classes and dataset sizes of training datasets used for each classifier.

5.2.6.4. Data Pre-processing

Several standard ML pre-processing tasks were performed to transform each dataset into a data structure that can be used by the ML algorithms to produce their optimal prediction performance. First, features with missing values were identified. Of the 83 features, 53 have missing values with a percentage ranging from 0.3 to 56.4. These percentages of missing values are acceptable by many ML practitioners. Correlations between features were analysed using Pearson's correlation matrix in order to determine redundant features. Two features (feature 26 and 27) were found to have a high correlation value of 0.99, thus one of them (feature 27) was dropped. Categorical features were then encoded as unique numeric values, with missing values given their own unique labels.

We compared four common imputation methods: mean, median, most frequent and k-NN (k=4) to determine the most efficient approach to handle missing values. We found that the mean method produced the best results when we ran one of the algorithms (RF) on the dataset and this was subsequently used for all features. The datasets have imbalanced class sizes as

indicated in Table 5.4. We oversampled small size classes in order to balance all the classes by applying a SMOTE method (described in section 3.4.3.5). Finally, a data scaling technique was applied to standardize data ranges of all features by transforming the data such that its distribution has a mean value 0 and standard deviation of 1.

5.2.6.5. Performance Results

In this section, we present and compare results of the classifiers (for both traditional and DL based experiments) and the architectures.

A. Performance of Individual Classifiers Using Traditional Machine Learning Algorithms

For each of the six classifiers (A, B, C.1, C.2, D.1, D.2), the performance of the eight traditional ML algorithms (LR, k-NN, DT, NB, SVM, ANN, RF and GB) were compared to determine the best performing algorithm for each classifier. Firstly, feature selection using backward feature elimination method (described in section 3.4.3.3) was performed to find the best set of features for each classifier. Then the ML algorithms were applied on the best set of features to determine the best performing algorithm and its best performance results. For each classifier, stratified cross validation technique (k -fold where k is 10) (described in 3.4.5) was applied to train and test the algorithms in order to obtain their average prediction scores. Each algorithm was tuned using Random Search method to obtain its optimal performance. Figures 5.22a-f show the performances of ML algorithms in each classifier across all threshold values in the ROC curves. Tables 5.5a-b summarize results of the four best performing algorithms for each classifier. The results indicate that RF yields the best performance across most metrics in each classifier. Table 5.6 indicates the tuned hyperparameters of RF for classifier B (the best overall classifier – see section 5.2.6.5-part C) and their values which yielded the optimal performance.

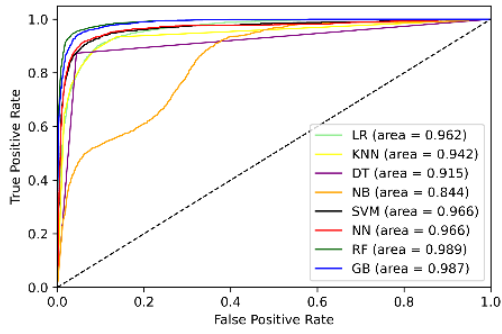


Figure 5.22a. ROC curves of the traditional ML algorithms for classifier A.

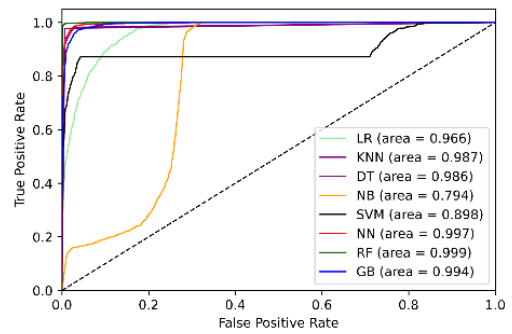


Figure 5.22b. ROC curves of the traditional ML algorithms for classifier B.

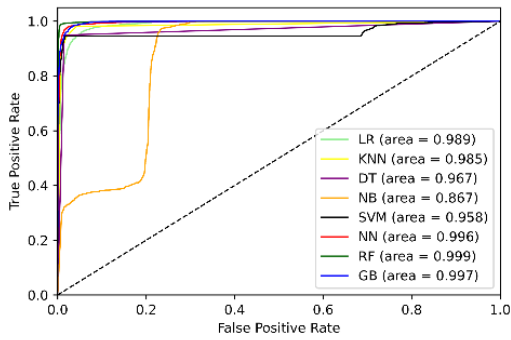


Figure 5.22c. ROC curves of the traditional ML algorithms for classifier C.1.

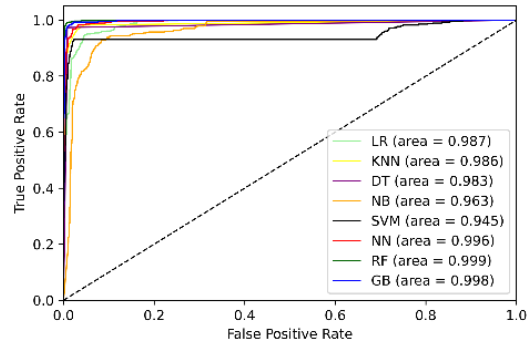


Figure 5.22d. ROC curves of the traditional ML algorithms for classifier C.2.

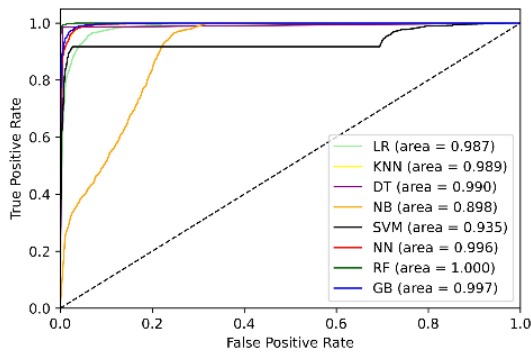


Figure 5.22e. ROC curves of the traditional ML algorithms for classifier D.1.

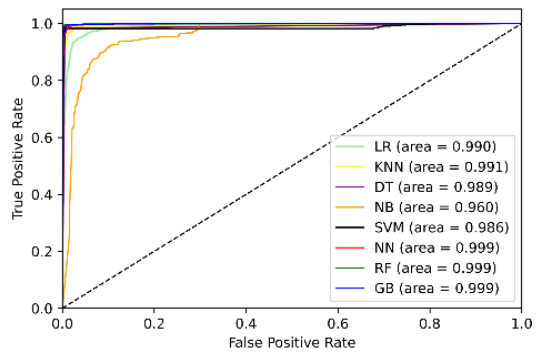


Figure 5.22f. ROC curves of the traditional ML algorithms for classifier D.2.

Algorithm	Classifier A				Classifier B				Classifier C.1			
	Acc. (%)	FPR (%)	FNR (%)	F1 Score	Acc. (%)	FPR (%)	FNR (%)	F1 Score	Acc. (%)	FPR (%)	FNR (%)	F1 Score
DT	88.54	5.20	12.27	0.89	97.23	1.37	13.88	0.98	95.44	4.62	6.10	0.95
ANN	89.13	4.23	11.15	0.89	98.07	1.25	14.02	0.98	97.60	1.67	4.97	0.97
RF	93.58	1.82	8.62	0.94	98.42	0.57	5.88	0.99	98.36	0.69	5.05	0.99
GB	91.94	3.73	7.64	0.92	95.05	4.26	7.79	0.95	96.57	2.04	10.33	0.96

Table 5.5a. Performance results of top four best performing traditional ML algorithms for classifiers A, B and C.1.

Algorithm	Classifier C.2				Classifier D.1				Classifier D.2			
	Acc. (%)	FPR (%)	FNR (%)	F1 Score	Acc. (%)	FPR (%)	FNR (%)	F1 Score	Acc. (%)	FPR (%)	FNR (%)	F1 Score
DT	97.14	1.65	4.68	0.98	97.92	1.53	3.08	0.98	98.14	1.63	9.83	0.98
ANN	97.44	1.64	5.72	0.98	97.19	2.48	3.35	0.97	98.70	0.75	8.36	0.99
RF	98.49	0.56	3.82	0.99	98.64	0.62	2.15	0.99	99.16	0.19	7.07	0.99
GB	97.63	1.28	5.77	0.98	95.67	3.71	4.88	0.96	98.48	0.91	9.62	0.98

Table 5.5b. Performance results of top four best performing traditional ML algorithms for classifiers C.2, D.1 and D.2.

Hyperparameter	Description	Value
n_estimators	Number of trees	1000
max_features	Max number of features considered for splitting a node	log2
max_depth	Max number of levels in each decision tree	38
min_samples_split	Min number of data points placed in a node before the node is split	2
min_samples_leaf	Min number of data points allowed in a leaf node	3
bootstrap	Method for sampling data points	false

Table 5.6. The optimal values of the tuned RF hyperparameters for classifier B.

B. Performance of Individual Classifiers Using Deep Learning Algorithms

The three DL algorithms were run and their results were compared to identify the best performing algorithm for each classifier. First, the datasets for LSTM and 1D CNN were converted into time series (with time step = 1), a standard input data format for the algorithms. The DL algorithms were tuned using a similar approach, and the same hyperparameters and ranges of their values described in section 4.5.4.2. The optimal values of the hyperparameters identified by the method were used to build the classifiers. Here, we also found that only one layer was sufficient to produce optimal performance in all three algorithms. The final performance rating of each classifier was obtained by taking the average over five runs. Tables 5.7a-b summarize results of the algorithms for each classifier. The results show that FC-DNN has performed best across most of the metrics in most classifiers (A, C.2 and D.1) followed by 1D CNN (in C.1 and D.2). LSTM has outperformed the others algorithms in classifier B only. Figures 5.23a-c show the three network architectures of the best classifier (classifier B), as an example, along with the tuned hyperparameters and their optimal values.

Algorithm	Classifier A				Classifier B				Classifier C.1			
	Acc. (%)	FPR (%)	FNR (%)	F1 Score	Acc. (%)	FPR (%)	FNR (%)	F1 Score	Acc. (%)	FPR (%)	FNR (%)	F1 Score
FC-DNN	83.56	6.96	20.12	0.72	94.86	0.79	36.52	0.94	92.60	4.68	19.24	0.92
LSTM	82.60	7.91	28.23	0.73	96.29	0.63	35.07	0.95	93.39	1.28	24.44	0.93
1D CNN	81.89	7.45	31.05	0.71	94.50	1.18	32.46	0.93	93.78	1.10	23.22	0.94

Table 5.7a. Performance results of the evaluated DL algorithms for classifiers A, B and C.1.

Algorithm	Classifier C.2				Classifier D.1				Classifier D.2			
	Acc. (%)	FPR (%)	FNR (%)	F1 Score	Acc. (%)	FPR (%)	FNR (%)	F1 Score	Acc. (%)	FPR (%)	FNR (%)	F1 Score
FC-DNN	93.18	3.66	16.63	0.93	94.55	6.60	3.13	0.95	93.13	3.15	34.27	0.94
LSTM	91.54	5.44	21.35	0.92	90.24	10.14	10.26	0.90	94.69	1.47	41.03	0.94
1D CNN	90.58	2.50	37.67	0.90	91.89	6.37	12.04	0.90	94.62	1.35	42.68	0.94

Table 5.7b. Performance results of the evaluated DL algorithms for classifiers C.2, D.1 and D.2.

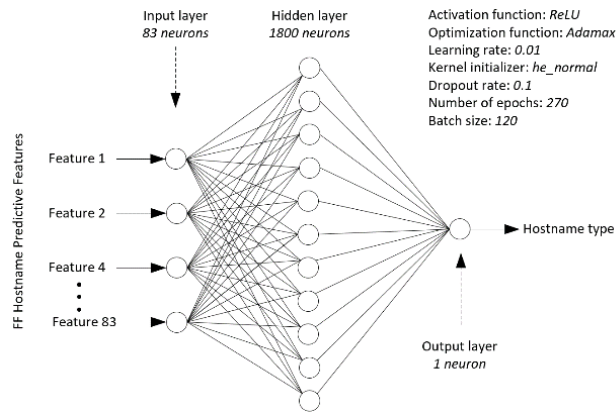


Figure 5.23a. FC-DNN architecture of classifier B.

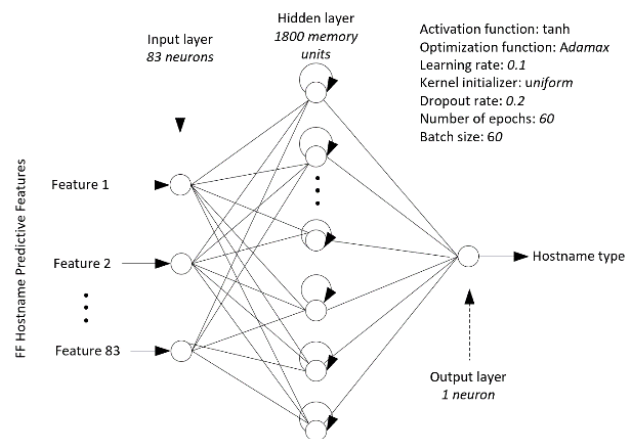


Figure 5.23b. LSTM architecture of classifier B.

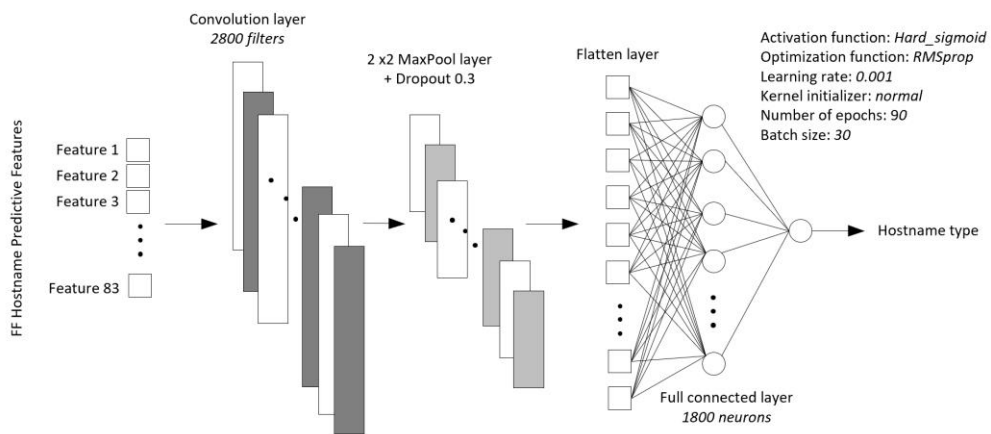


Figure 5.23c. 1D CNN architecture of classifier B.

C. Overall Performance of the Proposed Architectures

We now consider which of the architectures proposed in section 5.2.6.2 offers the best FF phishing hostname prediction performance. We first determined the overall performance of the architectures. For the hierarchical architectures, we applied the approach used by Kowsari, et al. [168] in which the overall accuracy at the leaf class (in this case the FF phishing hostname) is obtained by taking the accuracy of the child classifier as a fraction of the accuracy of its parent classifier. Since the LCPN hierarchical approach propagates misclassification errors from top to bottom [165], we compute the errors at the leaf class by summing the errors of the parent and child classifiers.

$$\text{Overall (leaf) accuracy} = \text{Accuracy of a parent classifier} * \text{Accuracy of a child classifier} \quad 24$$

$$\text{Overall (leaf) error rate} = \text{Error rate of a parent classifier} + \text{Error rate of a child classifier} \quad 25$$

Using the formulars above, the performance of architecture C was obtained by combining performances of classifiers C.1 and C.2 while for architecture D, the performances of classifiers D.1 and D.2 were combined. Table 5.8 below summarizes the best performance of each architecture. Of the multi-class architectures, Architecture D produced the overall best performance combining the highest accuracy with a relatively low FPR. However, the binary classification architecture B produced the best overall performance in all three metrics. We therefore conclude that the architecture B is the most suitable one for implementing the model.

Architecture	Acc. (%)	FPR (%)	FNR (%)	Classification Type
A	93.58	1.82	8.62	Multi-class
B	98.42	0.57	5.88	Binary
C	96.87	1.25	8.87	Multi-class
D	97.81	0.81	9.22	Multi-class

Table 5.8. Prediction performances of the model architectures.

5.2.6.6. Feature Performance Analysis

This section studies the importance of feature categories and individual features in the performance of the best performing classifier (classifier B). The classifier achieved the best performance with 56 features, of which 41 are novel and 15 were adopted from other works. 51 of these features were derived from third party services while only 5 of them were derived from the hostname string (local features). The importance weights of the best features of the classifier (shown in Figure 5.24) were computed using the tuned RF algorithm (described in section 5.2.6.5). To help explain this ranking, we will examine the data distributions of some of the best features with respect to the four hostname classes.

The number of times the hosts of a hostname matches with IP addresses of blacklisted phishing websites (features 77 in Table 5.1) was found to be the strongest predictor. Figure 5.25a visualises the distribution of values of this feature within each of the four classes by means of boxen plots. It may be seen that the hosts of FF phishing hostnames have the largest median count while a large number of them have counts of nearly 250. Those hosted in phishing non-flux networks have the second largest median value and a significant number of them have their hosts matched in the blacklist. A small number of CDN hostnames were found to have their hosts matched with those in the blacklist, suggesting that the hosts are the compromised machines used by phishers to host their phishing websites. Legitimate non-flux hostnames have the lowest count by far. A similar pattern is observed in the binary classification task in Figure 5.25b. This suggests that hosts of both phishing FFSNs and phishing non-flux networks are often used to host different phishing websites across different periods. For FFSNs, this could be through an FFSN hosting multiple phishing websites with the same pool of flux agents or multiple FFSNs hosting their unique websites using the shared pool of flux agents. This is consistent with the results of the network graph analysis of the monitored data reported in section 5.2.3.2.

The hostname similarity features (KL, JI and ED) are ranked at positions 3, 6, 7, 13 and 16. Although the features were originally used by Yadav, et al. [267] and Fu, et al. [268] to detect algorithmically generated domains (AGDs) hosted in domain flux botnets, their high ranking demonstrates that they are also effective in detecting hostnames hosted in the phishing FFSNs. Figure 5.26a shows the distribution of differences in average edit distances (feature 61) between the average ED of each hostname type and sum of averages of ED of the other three

types of hostnames. The FF phishing hostnames have the largest median difference followed closely by phishing non-flux hostnames while the two types of legitimate hostnames have the smallest median. The pattern is also illustrated in Figure 5.26b in which the differences were obtained by subtracting the average ED of FF phishing hostnames from the average ED of the phishing non-FF hostnames, shows that FF phishing hostnames require the fewest number of operations to transform one hostname string to another, thus they have the closest resemblance to each in terms of character composition compared to the rest. Similar observations were made in the other four hostname similarity features (features 57 - 60). The close similarity in the median values and the patterns between the two categories of phishing hostnames when compared to those of legitimate hostnames suggests that FFSN and phishing non-flux network owners tend to use similar strategies in composing their phishing hostnames, which are different from the strategies that are used to generate legitimate hostnames.

TTL, domain age and average domain age of co-hosted hostnames lead the ranking of temporal based features at positions 2, 8 and 10 respectively. As expected, Figure 5.27a shows that CDN and FF phishing hostnames tend to have lower TTLs than the non-flux hostnames. The most popular TTL values for all classes are 60s and 300s, with 60s being more popular for FF phishing and CDN hostnames and 300s for non-flux hostnames. CDN hostnames are more likely than FF phishing hostnames to have TTLs below 60s, and less likely to have TTLs above 300s. Phishing non-flux hostnames are the most likely to have TTLs above 3600s while a significant number of them have TTLs as large as 144000s. When CDN hostnames and the two types of non-flux hostnames are combined as a single category (see Figure 5.35b), a hostname with a TTL of 300s is much less likely to be an FF phishing hostname than a non-FF hostname. Note that in both figures, the total percentage of hostnames for each hostnames type are less than 100% because of the missing values.

The prediction significance of domain age is explained by the data analysis in Figure 5.28a-b. Although the median domain age of phishing non-flux hostnames is the lowest (Figure 5.28a), the value is increased considerably to 3571 days when they are combined with both types of legitimate hostnames (Figure 5.28b). The median age of FF phishing hostnames is 1219 days. A similar pattern is observed in the other domain age related features (features 13-15). The fact that the two phishing classes have the lowest domain age medians, this shows that phishers often use newly registered domains to operate their phishing websites.

The average number of hops between user and hosts of a hostname (feature 16) is the highest ranked spatial features at position 11. The data in Figure 5.29a shows the distribution of the four hostnames. Unexpectedly, CDN hostnames have the largest median while phishing non-flux hostnames have the smallest median. We anticipated that FF phishing hostnames would have the largest median count of hops as flux agents are expected to be hosted in more widely distributed networks. On the other hand, CDNs serve contents to users from local hosts, therefore they would be expected to exhibit a low median. However, it is important to note that the data distribution of any feature relating to user and the hosts depends on the current geographical location of the user relative to the hosts. Different distributions of such features are expected to be generated when users are at different locations. Figure 5.29b shows that the median of FF phishing hostnames is larger compared to that of the other three hostname types combined, indicating that the feature is relevant for the binary classification task.

The average number of co-hosted websites in the hosts of a hostname (feature 53) is ranked at the 14th position. The distribution of the feature in Figure 5.30a shows that hosts of the majority of phishing non-flux hostnames co-host a large number of other websites, followed by the hosts of FF phishing hostnames. Hosts of most CDN hostnames co-host the smallest number of other websites compared to the rest of the hostname types. We expected phishing FFSNs and CDNs would co-host a large number of malicious and legitimate websites respectively compared to non-flux networks. A possible explanation for the observed data distribution is that the non-flux phishing networks require the fewest resources and therefore they are cheap to build and maintain compared to the other networks. Since phishers are driven by profits, the networks become an ideal platform to host multiple phishing websites in order to maximize their returns of investment. While hosts in CDNs are typically high-performance machines, the services they support often have a high demand for resources, so the number of services running on a given host is likely to be modest. Furthermore, very large internet companies such as Google and Microsoft are likely to own CDNs dedicated to their own services. Figure 5.30b shows a comparison of the feature between the FF phishing hostnames and the other three types of hostnames combined as a single type, in which the median of the former is slightly large the latter thus the feature is useful for the binary prediction task.

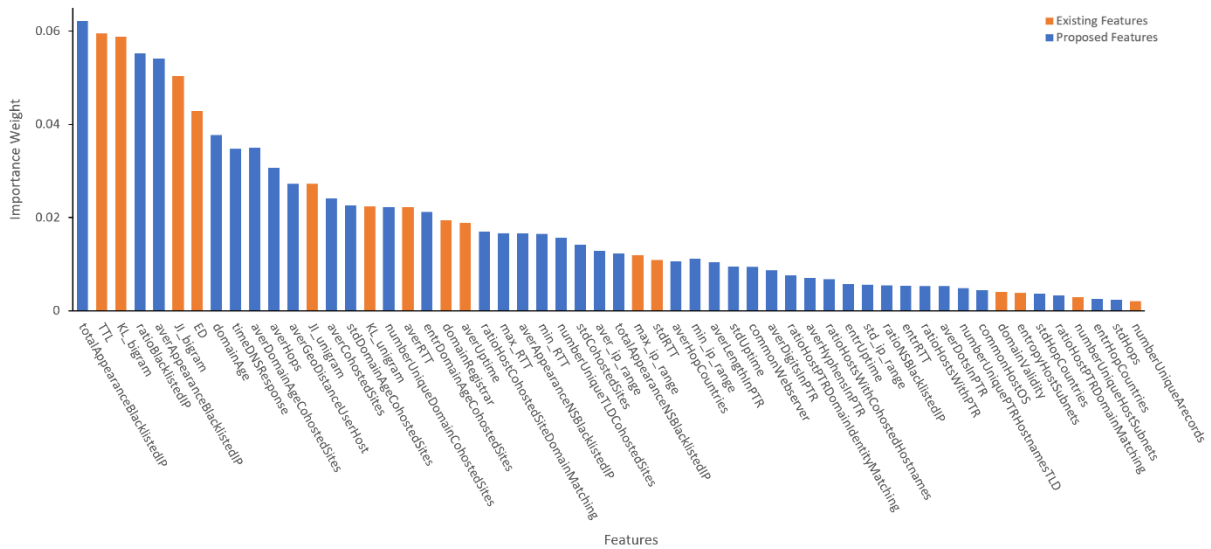


Figure 5.24. Importance weights of the best features of classifier B.

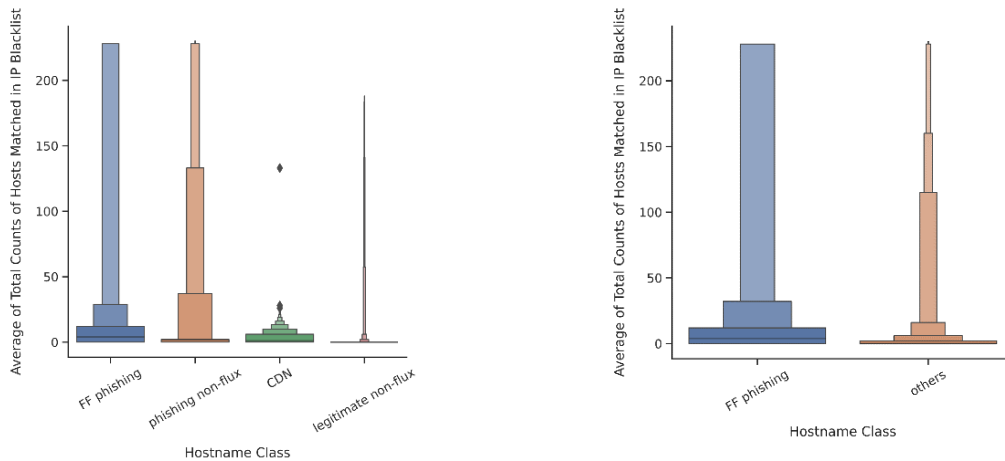


Figure 5.25a. Distribution of total number of occurrences of hosts' IP addresses of the four types of hostnames in a list of IP addresses of blacklisted phishing websites.

Figure 5.25b. Distribution of total number of occurrences of hosts' IP addresses of two types of hostnames in classifiers B in a list of IP addresses of blacklisted phishing websites.

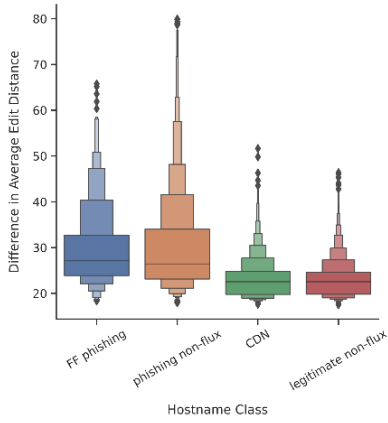


Figure 5.26a. Distribution of differences in average edit distances between FF phishing hostnames and each of the four types of the hostnames.

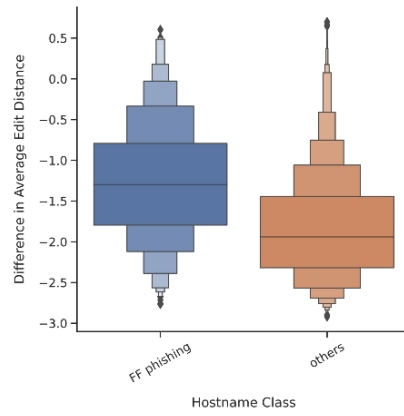


Figure 5.26b. Distribution of differences in average edit distances between FF phishing hostnames and each of the two types of the hostnames in classifier B.

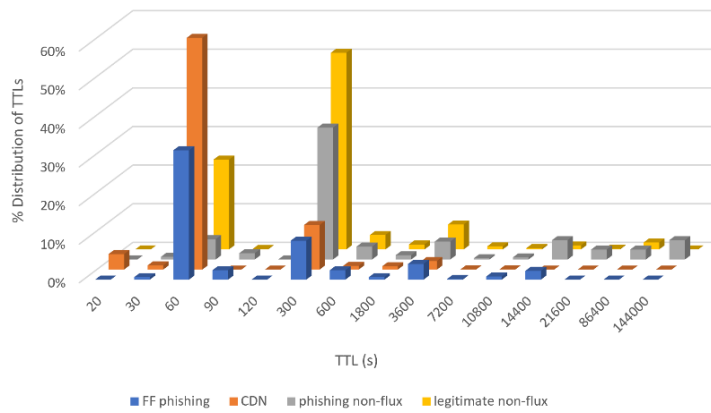


Figure 5.27a. Distribution of TTLs of the four types of hostnames.

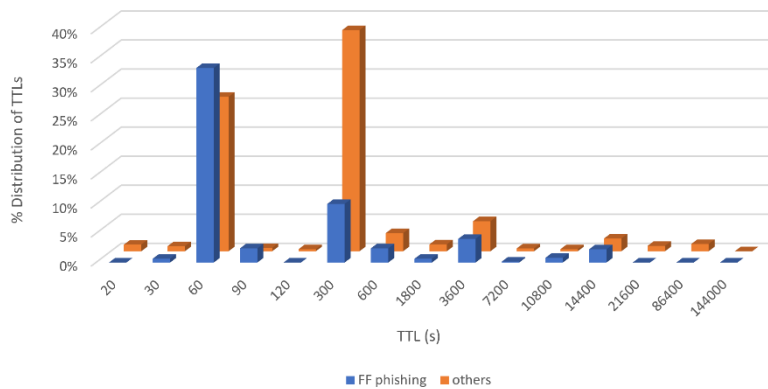


Figure 5.27b. Distribution of TTLs of the two types of hostnames in classifier B.

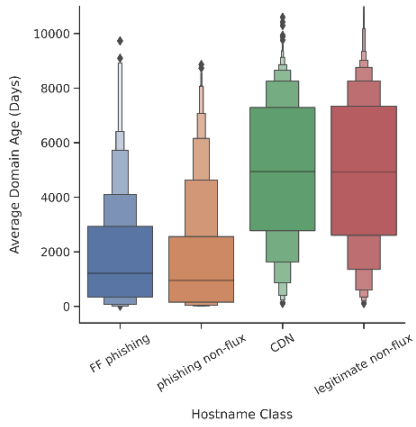


Figure 5.28a. Distribution of domain age of the four types of hostnames.

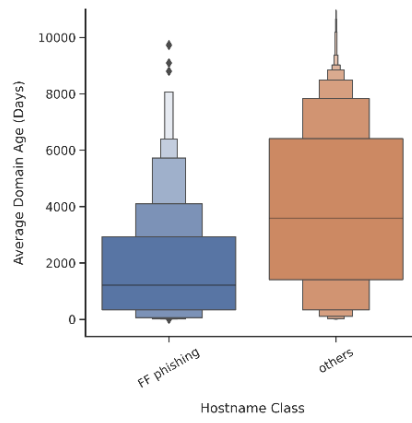


Figure 5.28b. Distribution of domain age of hostnames of the two types of hostnames in classifier B.

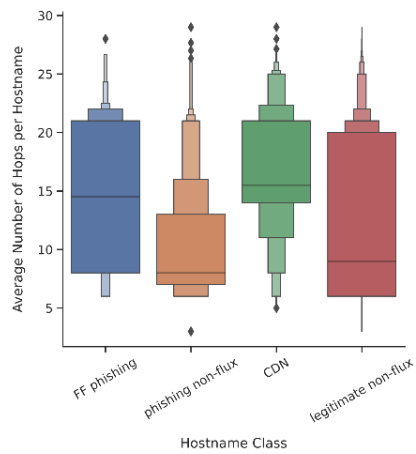


Figure 5.29a. Distribution of average number of hops between user and hosts of a hostname in each of the four types of hostnames.

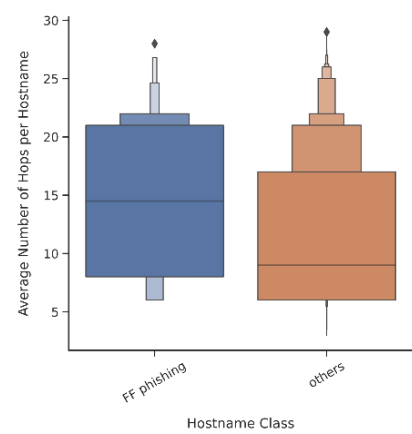


Figure 5.29b. Distribution of average number of hops between user and hosts of a hostname in each of the two types of hostnames.

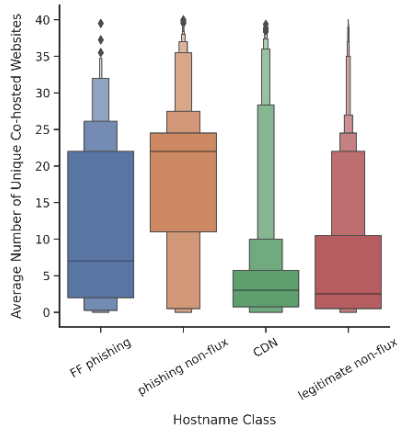


Figure 5.30a. Distribution of average number of unique co-hosted websites in the hostname’s hosts for all four types of hostnames.

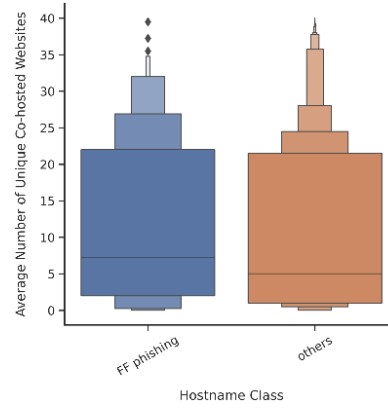


Figure 5.30b. Distribution of average number of unique co-hosted websites in the hostname’s hosts for the two types of hostnames in classifier B.

We also analysed the composition and the performance contribution of each feature category in the best feature subset of classifier B. Table 5.9 shows that all categories are represented in this subset. As shown in Figures 5.31a-b, temporal and DNS categories produced the highest accuracy rates and lowest error rates indicating they are strong predictors. The network and the host features, on the other hand, are the weakest predictors, having the lowest accuracy rates and the highest error rates. This result is consistent with Table 5.9 which shows that all or most temporal and DNS features are in the best feature subset and Figure 5.24 which indicates that the majority of them are highly ranked. The performance of the full best feature set is better than that of each individual category in terms of accuracy and FNR. The overall FPR, however, is slightly higher than those of temporal and DNS. Accuracy and FNR are the most important metrics for this problem as they measure percentages of correct predictions and of misclassified FF phishing hostnames as non-FF phishing hostnames respectively. Their improvements after combination of the feature categories suggests that the combination is beneficial. The increased diversity of features due to the combination should also improve the resistance to detection evasion techniques.

Similarly, we compared performance contributions of the novel and the adopted features to the overall performance of the classifier (results illustrated in Figures 5.31c-d). While our novel features achieved better results than the adopted ones, a combination of the two has improved

the overall accuracy, and more significantly the FPR and FNR, suggesting that their combination is important to achieve the best possible prediction performance.

Feature Category	Tally of Full Features	Tally of Best Features	Best Features # (# from Table 5.1)
Temporal	15	15	1-15
Spatial	23	10	16 - 20, 28, 35 - 38
DNS	23	20	39, 40, 42 - 50, 53 - 61
Network	9	2	62, 63
Host	6	2	73, 75
Reputation	7	7	77 - 83

Table 5.9. Composition of categories in the best feature set of classifier B.

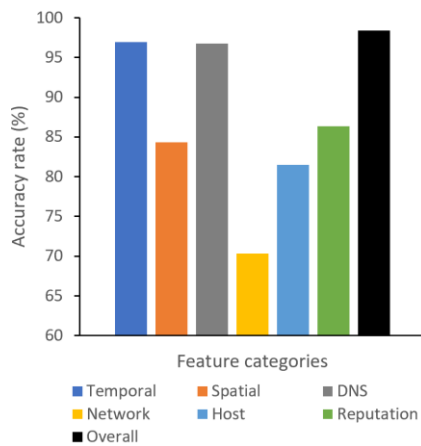


Figure 5.31a. Comparison of accuracy rates of feature categories of classifier B.

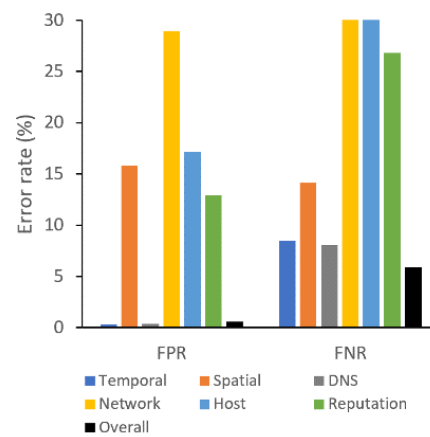


Figure 5.31b. Comparison of error rates of feature categories of classifier B.

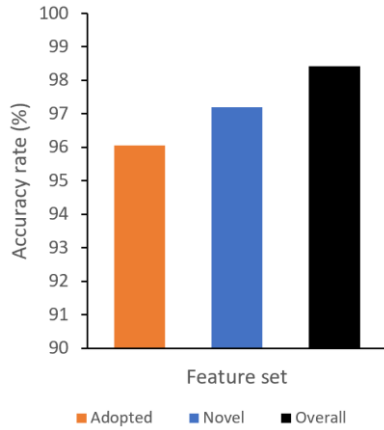


Figure 5.31c. Comparison of accuracy rates between novel, adopted and overall features of classifier B.

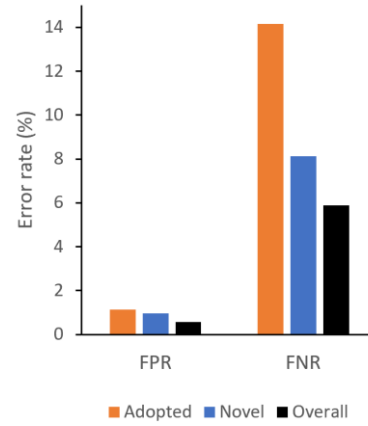


Figure 5.31d. Comparison of error rates between novel, adopted and overall features of classifier B.

5.2.6.7. Detection Time Analysis

Table 5.10 lists the runtimes of the classifier B's main steps (i.e webpage downloading from its server, PDC webpage filtering, hostname retrieval, feature extraction, dataset training and prediction) as illustrated in Figure 5.20. To determine the detection time per a webpage, we summed the average time to load a webpage, check for PDC webpage, retrieve a hostname, extract the 56 best features for a hostname and the average time to predict a new hostname, giving a total of 163.6 seconds. Almost all the prediction time is due to feature extraction. Note that many features are derived from the results of queries to third-party services, which entail data retrieval and network overheads. All the features were extracted sequentially. Figure 5.32 breaks down the extraction time by activity. There is a considerable variation of times with host scanning (using Nmap) taking the longest by far, over 20s more than the second most expensive, traceroute. It is important to note that the runtime of each activity was affected by the Python libraries we selected to use as well as the function(s) and the coding style we used to implement the activity. We expect a considerable speed-up to be achievable through more efficient coding and choice of libraries, and by querying services concurrently where possible. Nevertheless, it is likely that the latency will not be acceptable for a real-time use.

Phase	Time (s)
Webpage loading	0.8430
PDC webpage filtering	0.0976

Hostname retrieval	0.0001
Feature extraction per webpage	162.6400
Prediction per webpage	0.0003
Detection Time	163.5837
Training the dataset	8.5000

Table 5.10. Runtimes of the classifier B’s three stages for predicting FF phishing hostnames.

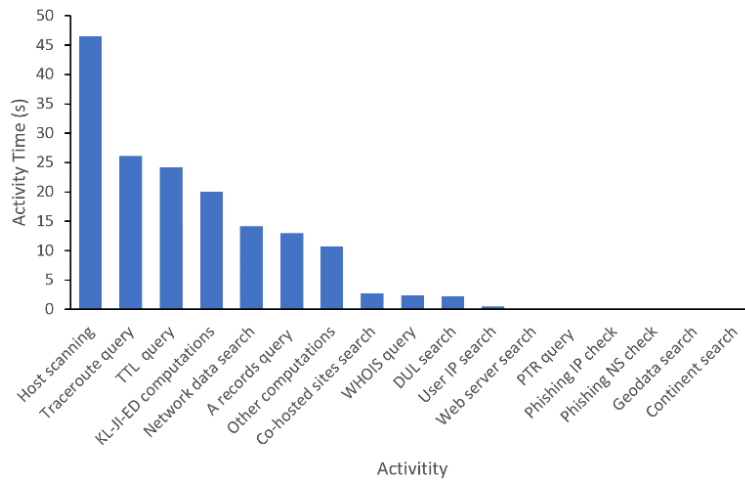


Figure 5.32. Distribution of feature extraction times by activities.

5.2.6.8. Model Validation Using New Data

To confirm that the excellent performance figures reported above were not due to characteristics of the particular dataset used, a new dataset was collected over a later period. 1,742 legitimate and 1,717 phishing new websites were monitored for eight weeks between 16th of November 2020 and 3rd of February 2021 and labelled to form the new testing dataset. Similar data pre-processing to that described in section 5.2.6.4 was performed. The best classifier (classifier B), trained using the dataset described in section 5.2.6.3, was tested against the new dataset. The classifier achieved an **accuracy of 96.07%**, **FPR of 2.55%** and **FNR of 7.09%**. Comparing with the evaluation performance obtained in section 5.2.6.5, the classifier performed less well by margins of 2.35% for accuracy, 0.98% for FPR and 1.21% for FNR. Though the performance is lower, it is still better than those reported by some of the previous works (see appendix VI). Nevertheless, the consistency of performance and the reason for the variations should be investigated using further new datasets collected in other separate periods.

One contributing factor to the reduction in performance may be that phishers and/or owners of legitimate services are altering their practices over time, meaning that regular retraining of the classifier using new datasets needs to take place to maintain the performance.

5.2.7. Discussions

5.2.7.1. Comparison with Existing Works

This section compares previously proposed FF hostname detection approaches with our work. To ensure a fair comparison, we focus on methods that utilize supervised ML techniques and that were trained and evaluated using datasets of a similar size to ours. Table 5.11 summarizes the comparison.

Work	Features	Data Size (URLs)	Classification Type	Evaluation Algor.	Perform.	Detection Time (s)
Hsu, et al. [137]	2 temporal features, 4 host features	17,214	Binary classification (FF hostnames versus benign hostnames)	SVM	Acc. = 95% AUC = 0.993	Unknown
Lin, et al. [128]	3 DNS features, 1 temporal feature	12,952	Binary classification (FF hostnames versus benign hostnames)	Genetic algorithm	Acc. = 98.2% FPR = 1.78%	18.54
Jiang and Li [258]	2 DNS features, 1 network feature, 1 temporal feature, 1 spatial feature	26,873	Binary classification (FF hostnames versus benign hostnames)	SVM, Naïve Bayes, K-NN	Acc. = 96.7% Prec. = 0.965 Recall = 0.98 F1 = 0.973 AUC = 0.99	400
Almomeni [130]	11 DNS features, 3	7,615	Binary classification	Adaptive evolving	Acc. = 98% Error rate =	Unknown

	temporal features		(FF hostnames versus benign hostnames)	fuzzy neural network, SVM, Naïve Bayes	1 – 16 (%)	
Yadav, et al. [267]	4 features DNS features	79,930	Binary classification (AGDs versus benign hostnames)	L1-Regularized Linear Regression	Acc. = 100% FPR = 2.0%	Unknown
Our work	15 temporal, 23 spatial, 20 DNS, 4 network, 3 host, 7 reputation	11,801	Binary classification (FF phishing hostnames versus other hostnames)	Linear Regression, K-NN, Decision Tree, Naive Bayes, SVM, Neural Network,	Acc = 98.42% FPR = 0.57% FNR = 5.88% Prec. = 0.99 Recall = 0.99 F1 = 0.99 AUC = 0.99	163.6
			Multi-class classification (FF phishing hostnames versus phishing non-flux, CDN hostnames and legitimate non-flux hostnames)	Random Forest, Gradient Boosting, Fully-connected Deep Neural Network, LSTM, CNN	Acc = 97.81% FPR = 0.81% FNR = 9.22% Prec. = 0.98 Recall = 0.98 F1 = 0.98 AUC = 0.98	

Table 5.11. A summarized comparison chart between our work and the related works.

I. Prediction Performance

Our proposed classifier has generally performed well across all metrics compared to the other classifiers in the chart. With the exception of Yadav, et al. [267]’s classifier, our binary classifier has outperformed the rest in terms of accuracy. The FPR achieved by our classifier is the lowest compared to the other classifiers. Our multi-class classifier has achieved a comparable accuracy with those of the other classifiers and has attained the lowest FPR.

Notably, Lin, et al. [128] reported a performance almost as good as ours with a smaller number of features and lower detection time. Their classifier, however, was trained and evaluated with a dataset which excluded malicious non-flux hostnames. We evaluated three of their four features, which we adopted in building classifier B, using the dataset we used for classifier B. We evaluated them first using a three-class dataset, from which we had eliminated phishing non-flux hostnames, and then a four-class dataset. In the former, we obtained an accuracy rate, FPR and FNR of 73.96%, 23.49% and 72.36% respectively whereas in the latter, an accuracy of 60.74%, FPR of 56.42% and FNR of 37.86% were obtained. Note that the accuracy and FPR decreased as a result of adding the fourth hostname class. We expect a similar reduction in the performance of their classifier if their dataset was to be augmented with malicious non-flux hostnames. Also, their features performed less well against our phishing-based dataset, suggesting that their performance is sensitive to the type of FFSNs in the dataset. Another main difference with our work is that we have used six different types of feature categories compared to two used in their work. This has a significant implication on the ability of the solution to resist detection evasion as further explained in section IV. In addition, our work has evaluated the features with a larger number of metrics (8) compared to theirs (2), thus we have been able to reliably measure the all-round effectiveness of our model.

We also compared our classifier against Yadav, et al. [267]’s classifier from which we adopted most features (features 57-58 and 60-61). Their classifier used the hostname similarity-based features to distinguish algorithmically generated domains (AGDs) from the legitimate hostnames. AGDs are domains of a malicious website that are frequently changed by an algorithm of a flux botnet. We evaluated the four adopted features using our classifier D.1 (with the rest features being dropped) which distinguishes phishing (both AGDs and non-AGDs) from legitimate hostnames. We obtained an accuracy of 92.37% and FPR of 4.82% which is

not as good as the performance reported by Yadav, et al. [267] (see Table 5.11). This is not surprising, as human-generated (non-AGD) phishing hostnames are likely to resemble legitimate hostnames (often by design) rather than AGDs. Nevertheless, the features do increase significantly the performance of our classifiers. With the features excluded, classifier D.1 produced an accuracy of 96.37% and FPR of 3.85% but the performance improved to 98.64% and 0.62% respectively with the features restored. The improvements were also observed in the other classifiers.

II. Performance Measures Reported

As shown in Table 5.11, the existing works have reported only a small number of performance metrics, omitting other significant metrics such as FNR, which we consider to be one of the most important measures for this problem (based on the reason we cited in section 4.6.1.4). This limits our understanding of the all-round effectiveness of these solutions. Conversely, our work has been evaluated using a larger number of metrics, which proves its reliability and all-round robustness.

III. Specificity

While all other works proposed classifiers to detect FF hostnames operating general malicious services including spam, malware and phishing, our work has a more refined scope, focusing on those hosting phishing websites only. As mentioned in section 5.2.2, based on Caglayan, et al. [272]’s study, our most effective set of features are likely to be more effective in recognising the phishing specific FF hostnames than those proposed by the other works.

IV. Real-world Relevance and Applicability

Our classifier has included all four types of hostnames, thus it is more realistic and relevant to the real-world application. The other studies excluded malicious non-flux hostnames, which host the majority of malicious websites, when training their models. To be useful in practice, their classifiers should include these hostnames in their training datasets, which will increase complexity of the problem and is likely to reduce their performance. We demonstrate this by evaluating our classifier B using a three-class dataset (after removing phishing non-flux

hostnames from the same original dataset we used in evaluating the classifier - see section 5.2.6.3) and compare against the performance we obtained with the four-class dataset (in section 5.2.6.5). With the three-class dataset, the classifier obtained an accuracy of 98.97%, FPR of 0.41% and FNR of 3.80%, which is higher than the one obtained with the four-class dataset.

V. Feature Diversity and Evasion Resistance

We have used features belonging to six different feature categories, the most compared to the other studies. Our widely diverse set of features is likely to increase resilience of the solution to detection evasion techniques. In addition, most of our best features were extracted from third-party services. The third-party features are difficult to emulate or forge for the same reasons mentioned in our previous model (see section 4.3).

5.2.7.2. Limitations of the Proposed Model

I. Detection Time

Our detection time is higher than the typical delay a user can expect when accessing a webpage, measured to be in the region between 1 and 3 seconds [250]. This is not suitable for providing direct real-time protection to end users. Though the subset of 56 best features produced the best performance, smaller subsets have also produced good accuracy and moderate error rates. For instance, the top 7 best features produced an accuracy, FPR and FNR of 96.24%, 15.87% and 1.56% respectively. A trade-off can be made between performance and the number of features with the aim of reducing computing overheads, thus detection time, while maintaining acceptable performance. However, we argue that the use of a large number of features, not only has improved the performance across all metrics, but also is expected to enhance a resilience to detection evasion and therefore the long-term reliability of the solution. It is preferable, therefore, to reduce detection time as far as possible by other means before reducing the number of features.

In the experiments described above, features were extracted sequentially. The detection time can be reduced by extracting them in parallel as far as possible. Querying of A records needs to be performed first to retrieve the hosts' IP addresses as many features depend on this

information. Extraction of the individual features can then proceed in parallel. Based on the timings shown in Figure 5.40, this would reduce detection time to around one minute. In addition, features obtained from some of the time-consuming feature extraction activities (as described in section 5.2.6.7) can be dropped to further reduce the time at the expense of detection performance. For instance, by dropping all features obtained through host scanning, the detection time obtained was reduced to 45.5 seconds while the accuracy attained was 96.85%.

5.2.7.3. Application of the Proposed Model

The impact of longer detection time due to the use of large number of features can be countered by using the model to build and maintain a blacklist of FF phishing hostnames, which in turn can be used to provide a real time detection to end users. A blacklist approach has proved to be efficient for real-time applications in various cybersecurity problems [273, 274]. When applied in this way, our model achieves three important goals; high detection performance, real time detection and enhanced resilience to detection evasion. To build the blacklist, the model would be fed with a stream of hostnames obtained from various sources such as user emails, network traffic, and databases of legitimate and phishing websites, as inputs. Those classified as FF phishing hostnames would be added to the blacklist. The blacklist can then be used in various ways to provide real time protection to end users, e.g. via a web browser plug-in or a secure web gateway. Such a blacklist would be more up to date than those relying on manual reporting and verification of malicious hostnames because of the fast automated detection. The blacklist can also be a useful resource for security researchers, vendors and authorities for further investigation of the structures and operations of phishing FFSNs, and the development of solutions to take them down. Currently there are no active blacklists consisting of known FF phishing hostnames only that we are aware of.

Also, our classifier D.2, which distinguishes FF phishing and phishing non-flux hostnames with a high accuracy (indicated in Table 5.5b), can be integrated (as a second layer in a sequential implementation architecture) with a real time solution which distinguishes phishing and legitimate hostnames, such as the one we have proposed in Chapter 4. In this approach, the former detects FF phishing hostnames within the general phishing hostnames detected by the latter.

Although the multi-class model using architecture D performed less well than the binary model using architecture B, it has still produced good results. The advantage of this model is that it predicts each hostname type as an independent outcome. This can be useful in cases where security experts want to distinguish FF phishing hostnames from phishing non-flux hostnames. By doing so, specific measures to address the attacks at the network level can be applied. For instance, for the former, this would require an approach described in section 5.1. For the latter, hosts of the hostnames can be directly identified by querying A records of the hostnames and then the returned IP addresses blacklisted.

5.3. Prediction of Phishing Hostnames Hosted in NS IP Flux Networks

As mentioned earlier, any PDC webpage that is keeping its DNS records in an NSIFN is potentially a phishing webpage. In this section, therefore, we describe the development of an ML approach to predict phishing hostnames hosted in NSIFNs (here we refer to them as *phishing NS flux hostnames*) based on the DNS, host and networks characteristics of NSIFNs. Due to the similarity in the network characteristics of FFSNs and NSIFNs, we adopt the same supervised ML techniques and some of the predictive features from the previously proposed prediction model to build a prediction model for this task. Through flat and hierarchical classification approaches, several binary and multi-class classification-based architectures are proposed and evaluated to determine the best performing one for implementation of the model. We begin the section by reviewing the related studies and highlight their limitations.

5.3.1. Related Works

Studies in this domain can be categorized into two areas; (1) studies which investigate the existence and behaviours of NS flux hostnames and their networks and (2) those which propose techniques to detect NS flux hostnames. The two types will now be discussed in that order. Salusky and Danford [136] was the first study to define the concepts of NS IP flux and double flux networks and to investigate their operations and behaviours. They monitored A records of known malicious hostnames, NS records of authoritative NSs of the hostnames and A records of these NSs after every 2 minutes for a period of at least one month. As a result, they identified unique characteristics of both FFSNs and NSIFNs hosting the hostnames. These include high IP fluxing rates, IP addresses drawn from larger numbers of unique Autonomous Systems (ASs) and specific types of malicious activities running on the hosted websites. By infecting a

honeypot with the malware of the flux networks, they were able to study how infected machines are recruited and then commanded by their mothership to perform the malicious activities. The study also suggested six different strategies for mitigating flux networks.

Konte, et al. [143] investigated the fluxing behaviours of A records of malicious websites, and the names and A records of the authoritative NSs for the zones they belong to. They extracted 3,360 hostnames from spam emails and monitored their DNS records after every 5 minutes for a period of one month. To compare with legitimate hostnames, they extracted 500 top websites from the Alexa ranking and monitored their similar DNS records. The study observed that all the three types of DNS records of the malicious websites were changing over the time while only A records of the legitimate websites were changing. By comparing similarity of contents of the malicious websites, they generated 21 clusters of malicious campaigns. The study then compared various dynamic aspects of the infrastructures hosting the individual malicious and legitimate hostnames as well as the campaigns. The aspects compared were the rate of change of DNS records, the rate of growth of the networks, location of change in the DNS hierarchy, and topological and geographical distribution of the hosts. Metcalf and Spring [141] monitored NS records of the hostnames from a zone file of common gTLDs and those from the Security Information Exchange's passive data to illustrate the NS IP flux behaviour. The NS records were queried once in a day and then their 28-day records were collected for the analysis. The study statistically analysed the number of changes of A records of the NSs, the number of changes of Asynchronous System Number (ASN) of NSs and the distribution patterns of TTLs of NS records.

In the second category of the studies, Kadir, et al. [142] proposed a k-NN based classifier to detect single flux and double flux malicious hostnames using seven features. The features were based on the numbers of A records of the hostnames and of the NSs for the zones they belong to as well as the rate of change of the two records. They collected their dataset by monitoring the records of 500 malicious FF and legitimate hostnames (250 each) in every 12 hours for a period of 3 months. The results showed that the classifier did not yield any FPR or FNR. 182 of the 250 FF hostnames were classified as single flux while 68 were classified as double flux. The study also identified various types of IP address to NS mappings. Pa, et al. [140] proposed an IP address and hostname mapping technique to detect double flux hostnames by monitoring A and NS records of 50,030 malicious and legitimate hostnames for a period of 6 months. The

records collected were mapped using three criteria as detection features; single NS to many IP addresses, single IP address to many NSs and single IP address to both hostname and NS. Using a threshold of 3 for each of the features to classify the NSs, above 3 being the malicious one and below 3 is a legitimate one, the technique was able to achieve an FPR of 0.8% in detecting NS flux hostnames. The table in appendix VII summarizes the above reviewed works.

5.3.2. Limitations of Related Works

The techniques described above have two main limitations as described below:

1. The techniques proposed by Kadir, et al. [142] and Pa, et al. [140] depend on detection features whose data need to be monitored for an extended period of time (i.e 3 and 6 months respectively). The time allows the malicious NSs to continue providing DNS services to many malicious websites, causing more damages before being detected.
2. While Pa, et al. [140] did not mention the time interval between repeated DNS queries during the monitoring phase, Kadir, et al. [142] specified that they queried after every 12 hours. Salusky and Danford [136]’s work, however, showed that changes in A records of NSs can occur as frequent as every 30 minutes. Querying after long time intervals is likely to miss many NS changes in between, therefore losing valuable information which could affect the observed characteristics of the NS flux networks thus the detection performance.

Based on these limitations, the proposed techniques are insufficient for instant and accurate detection of zero-day phishing NS flux hostnames. In contrast, we aim to detect the hostnames at the instant their NSs are queried for NS records when users are attempting to access the websites. We also intend to monitor A records of the NSs at a shorter time interval in order to collect detailed data to capture more realistic characteristics of the networks. Our review in Chapter 2 (sections 2.3.3.3 and 2.3.3.4) suggested that FFSNs have a number of similar DNS, network and host characteristics to those of NSIFNs. Given the similarities and the high prediction performance of the previous model using features derived from the characteristics of phishing FFSNs, we are convinced that the features whose characteristics are common between the two types of flux networks can also be useful in predicting phishing NS flux hostnames. In this study we therefore identify these features from the previous model and evaluate them using a similar supervised ML approach to determine their effectiveness in

addressing this problem. This is the first study that has adopted features used to predict FF hostnames for the purpose of prediction of NS flux hostnames.

5.3.3. Monitoring and Network Analysis of Name Servers and Their Networks

5.3.1.1. Monitoring of Name Servers of Phishing and Legitimate Hostnames

We monitored changes in the IP addresses of authoritative NSs of 6,638 legitimate and 6,630 phishing websites for the purpose of labelling NS hostname classes for training the model for predicting phishing NS flux hostnames. The websites collected are different from those used in the previous study and were also collected at a different time. First, the legitimate hostnames and phishing URLs were collected from Tranco's list of 1 million most visited websites and PhishTank's blacklist respectively. Authoritative NSs (both primary and secondary) for each hostname and URL were obtained by querying the public NS (we used Google's DNS server) for NS records. For a period of 3 months, from 23rd of July to 27th of October 2020, A records for each NS were queried and collected after every 2 hours from the public NS. We chose a 2-hour interval based on our trial data which indicated that most NSs were changing their A records between 2- and 4-hour intervals. The IP addresses returned in the consecutive queries of the same NS were compared, and the number of times a change was observed throughout the period was recorded. Figure 5.33 illustrates the monitoring of A records of NSs for class labelling.

In order to label the hostnames as classes for the classification tasks, we first had to determine a threshold number of IP address changes to distinguish between flux and non-flux NS hostnames. According to Metcalf and Spring [141], legitimate NSs are not designed to exhibit IP fluxing behaviour, implying that changes in their IP addresses are due to other reasons such as routine maintenance of the servers and scaling of the network. For this reason, we picked 5 as the threshold, the maximum number of changes in legitimate NSs we observed in the data obtained from the NS monitoring task. Therefore, any phishing hostname with a number of IP changes in NS above 5 was labelled as phishing NS flux hostname otherwise it is phishing NS non-flux hostname. All legitimate hostnames were labelled as legitimate NS non-flux hostnames.

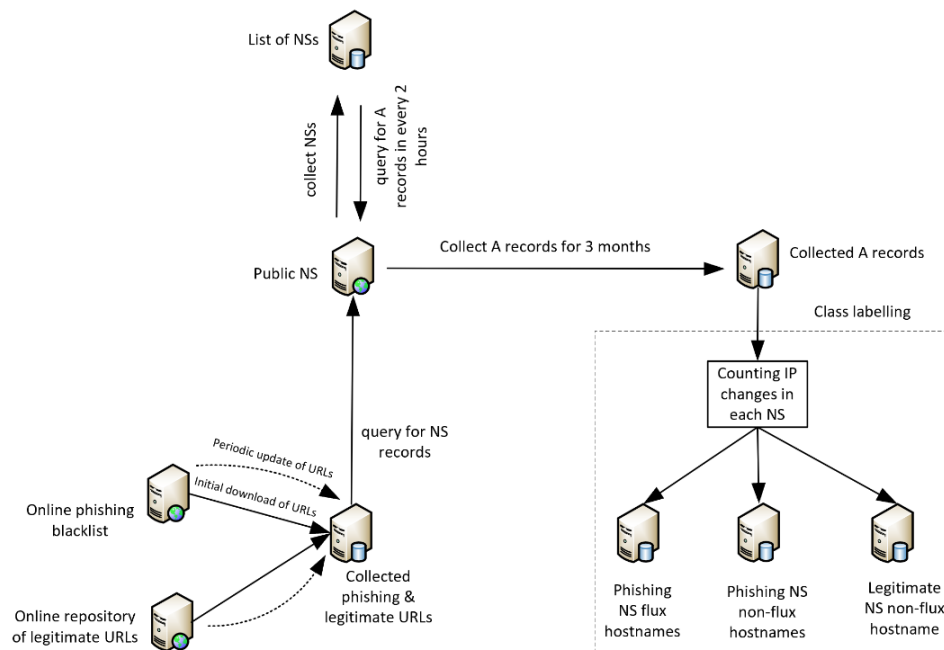


Figure 5.33. Monitoring of A records of NSs for labelling classes of the hostnames.

5.3.1.2. Network Analysis of NS Flux and Non-Flux Networks

Similar to section 5.2.3.2, we describe the network analysis of NS flux and non-flux networks to observe their distinctive structures and behaviours based on the collected data in section 5.3.1.1.

Number of Changes of IP Addresses

Figure 5.34 shows the distribution of the number of changes of IP addresses observed per NS type during the monitoring period. 82.3% of the legitimate NSs did not change their IP addresses, 17.7% changed between 1 to 5 times only, and none changed IP addresses more than 5 times. In phishing NSs, however, 31.6% of them were observed to change their IP addresses, with 5.4% changing more than 5 times. Unlike the legitimate NSs, some phishing exhibited many changes, with the largest number observed being 826. This data shows that a significant number of phishing websites are hosted in networks deploying NS IP fluxing.

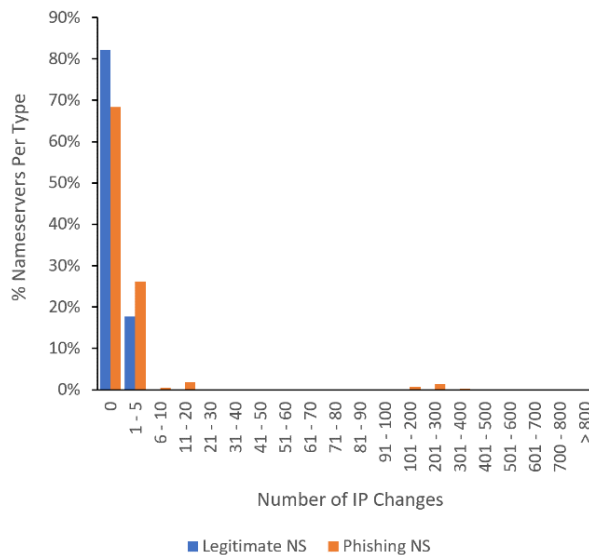


Figure 5.34. Distribution of number of changes of IP addresses of authoritative NSs observed per NS type.

Lifespan of Phishing and Legitimate NSs

During the monitoring process, we observed that 98% of the phishing NSs were active throughout the 3 months. This is different from the phishing hostnames, as observed in the previous study, in which only 37% were active throughout the 5-week monitoring period (see section 5.2.3.2). This suggests that the existing efforts towards taking down the NSs keeping DNS records of malicious web services are still insufficient. This is also reflected in the reviewed research works whereby we observed a smaller number of studies (2) proposing the detection of NS flux hostnames compared to a larger number of studies (12) detecting FF hostnames (see appendixes VI and VII). Also, we found a number of studies addressing the detection of bots, and botnet command and control servers operating malicious web services but none for those addressing the same machines operating the malicious NSs. As expected, all legitimate NSs were active throughout the duration.

Number of Cumulative Unique IP Addresses

Figure 5.35 shows the change in the number of cumulative unique IP addresses of fluxing NSs of phishing hostnames against the query number after every 6 hours. It can be observed that there is a fast rate of accumulation of new IP addresses between 1 and 61 query number followed with a slow rate between 62 and 286 query numbers. The time with the slow rate

indicates that some of the NSs were less or not active in recruiting new IP addresses. After the 286th query number, a sharp increase in the rate was observed. The pattern suggests that some of the phishing NSIFNs expand their networks at particular times and with varying rates. We think this could be because of the level of demands of the services to support the phishing campaigns at a particular season or event and/or the accessibility and availability of vulnerable machines for compromise and recruit.

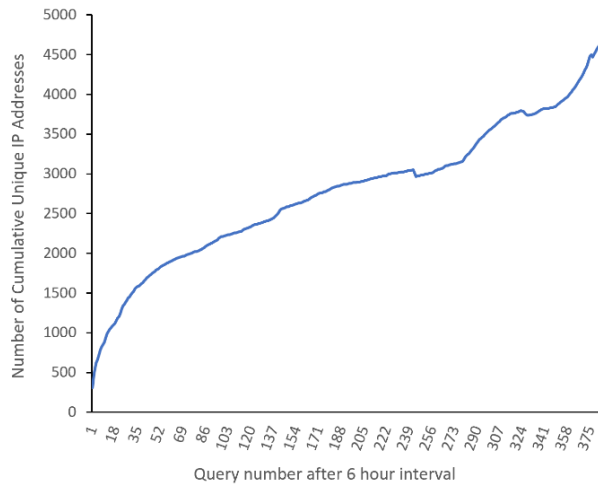


Figure 5.35. The change in the number of cumulative unique IP addresses of fluxing NSs of phishing hostnames.

Network Distribution of NSs

Figure 5.36 shows the distribution of the number of unique networks of NSs of the monitored hostnames. The largest number of unique networks per hostname for NSs of phishing NS flux hostnames observed was 18. The largest percentage of hostnames were hosted in 2 and 3 different networks while the least percentages were hosted between 10 and 15 networks. NSs of legitimate non-flux hostnames were the second most widely distributed hostnames with the largest number being 8. Similarly, NSs of majority of these hostnames were hosted in 2 and 3 unique networks. In the case of phishing NS non-flux hostnames, most of their NSs were hosted in 1 network with a small number hosted in 2 and 3 networks. The number of unique ASs of NSs per hostname (shown in Figure 5.37) shows a similar pattern. The observed wider network distribution of NSs of phishing NS flux hostnames compared to the others in this study is similar to the observations made by others as discussed in section 2.3.3.

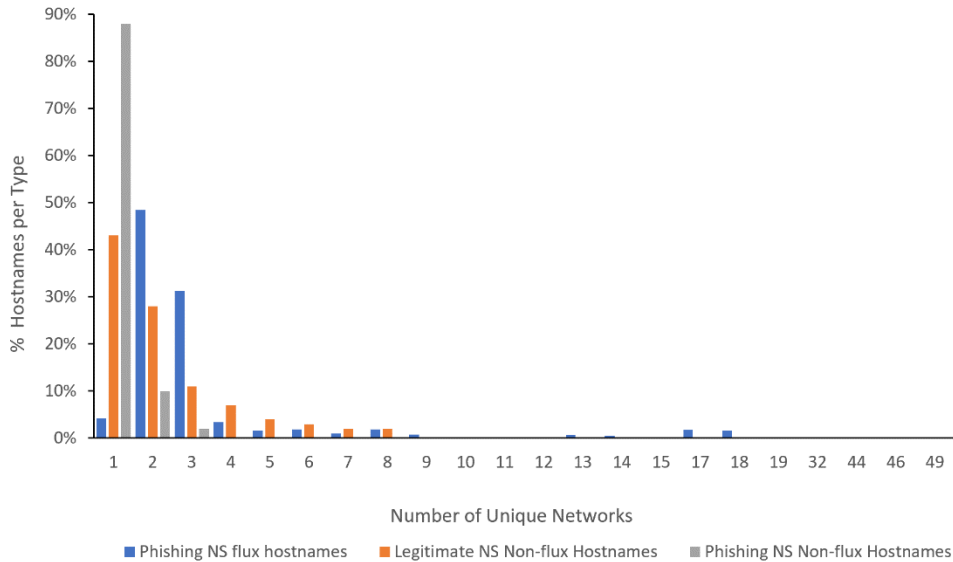


Figure 5.36. Distribution of number of unique networks of NSs of phishing and legitimate hostnames.

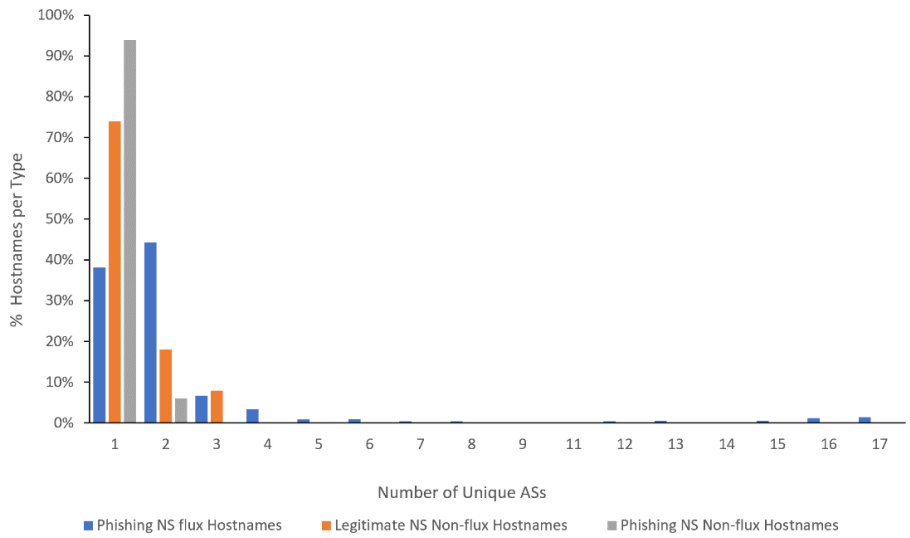


Figure 5.37. Distribution of number of unique ASs networks of NSs of phishing and legitimate hostnames.

The same organizations managing ASs were found to be the top hosts of NSs of both types of phishing hostnames. As in the previous study, Cloudflare was found to be the largest culprit followed by Amazon (see Figure 5.38). A total of 423 different organizations were observed to host NSs of the phishing hostnames in their ASs. With the exception of Cloudflare, the top organizations for hosting servers of phishing hostnames (see section 5.2.3.2) are different from those hosting NSs of the same type of hostnames. Also, the unique number of AS organizations for the latter (423) is much higher than the former (34). This is an indication that networks of

many different organizations, some of which belong to large internet service providers, are vulnerable to this exploitation. This is likely due to lack of or existence of weak security policies for monitoring potential exploitation of their resources. Figure 5.39 illustrates top organizations managing ASs containing NSs of legitimate NS non-flux hostnames. Here, we observed a total of 747 different organizations. It can be seen that the list of organizations here is different from the previous graph, with an exception of Cloudflare.

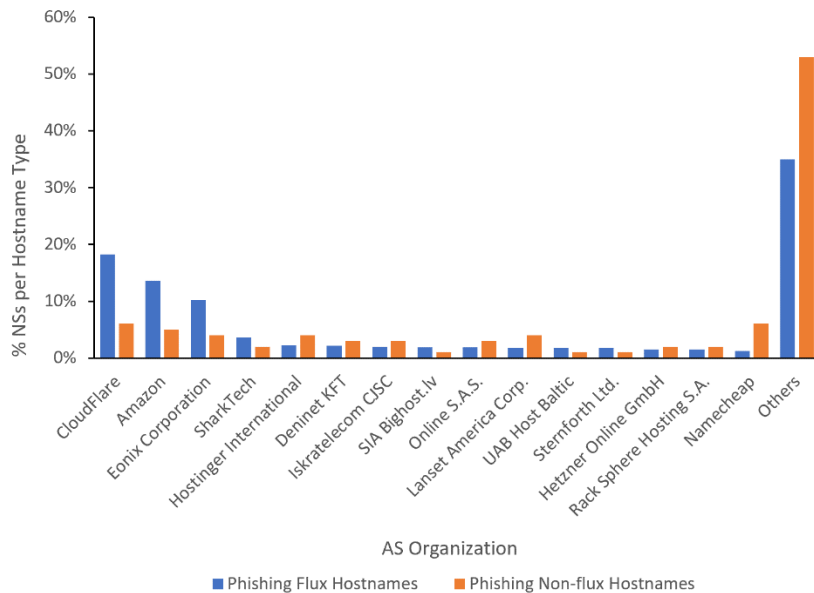


Figure 5.38. Top AS organizations hosting NSs of phishing hostnames.

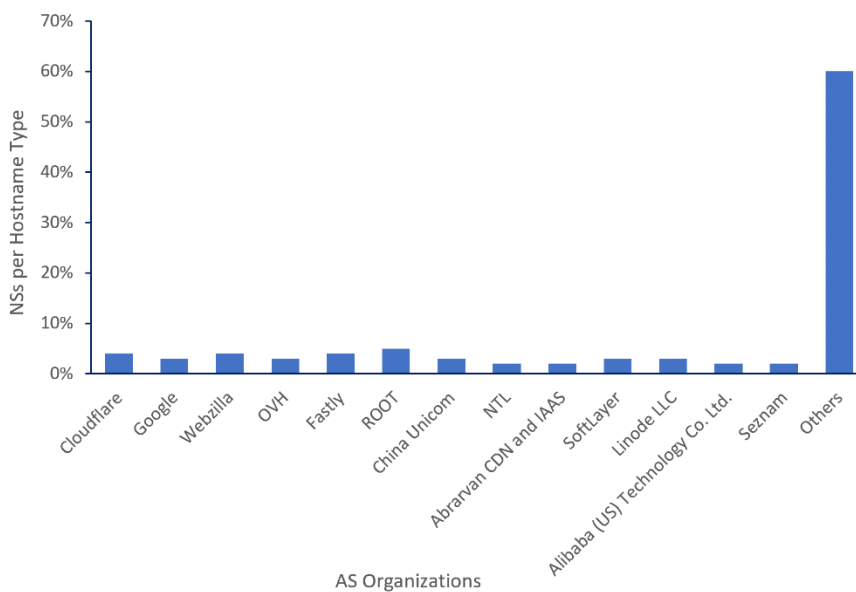


Figure 5.39. Top AS organizations hosting NSs of legitimate non-flux hostnames.

Hosting Countries of NSs

Figure 5.40 shows the distribution of unique countries hosting NSs of each hostname. NSs of phishing NS flux hostname were the most distributed with up to 13 unique countries whereas those of phishing non-flux hostnames were the least distributed, only between 1 and 2 countries. As shown in Figures 5.41 and 5.42, USA is by far the most popular country in hosting NSs of all types of hostnames. Compared to the equivalent distribution of hosts of phishing hostnames (see section 5.2.3.2), the NS flux agents are spread across more countries. For instance, in the former distribution, only five countries were hosting at least 2% of the flux agents with the top country hosting 70% while in the latter distribution, there were 14 countries with the same least percentage while the top country hosts only 39% of the agents. In Figure 5.42, it can be seen that there are countries in the previous list, including Russia, Poland, Lithuania, Hungary, Latvia and Panama, that are not in this list. Some of these countries were also observed to be the top hosts of servers of phishing hostnames as described in section 5.2.3.2. This suggest that there are large communities of phishers in these countries.

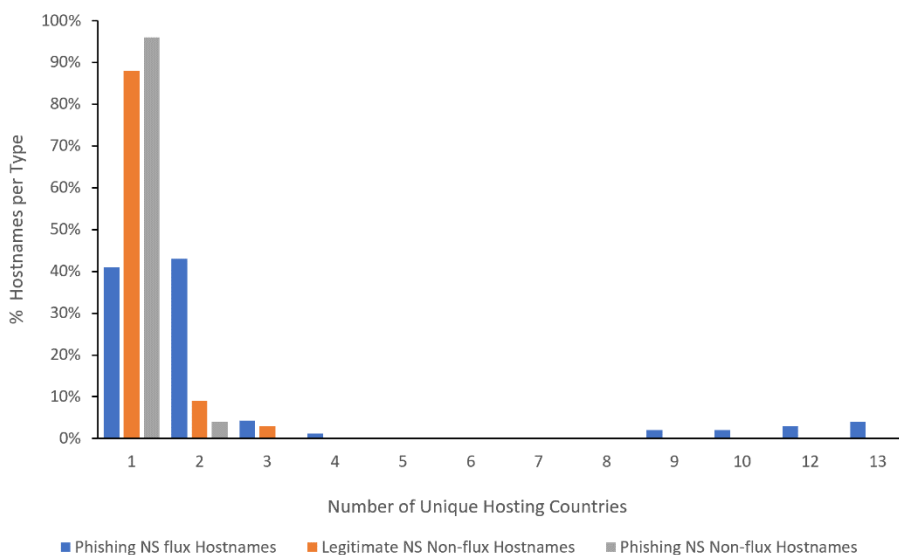


Figure 5.40. Distribution of number of unique hosting countries of NSs of the hostnames.

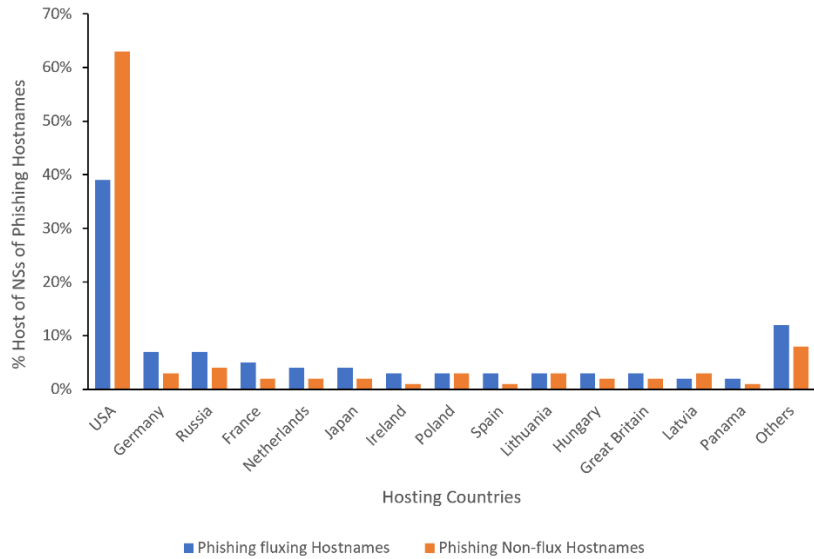


Figure 5.41. Top countries hosting NSs of phishing hostnames.

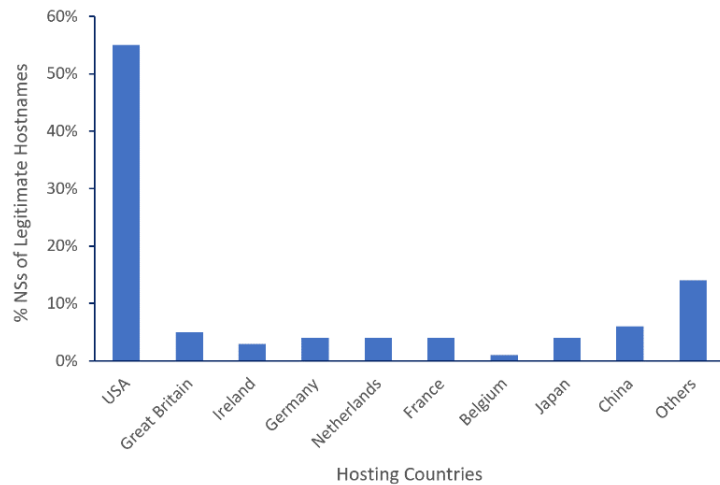


Figure 5.42. Top countries hosting NSs of legitimate hostnames.

Double Fluxing Behaviour of Phishing Hostnames

As pointed out in section 2.3.3.4, some phishing websites are hosted in networks which exhibit both fast flux and NS flux behaviours, a phenomenon known as double fluxing. To investigate whether the URLs observed with NS fluxing also exhibit fast-fluxing, we monitored changes in the IP addresses of hosts of the same phishing hostnames (described in section 5.3.1.1) at the same time as monitoring changes in the IP addresses of their NSs. The A records of the hostnames were queried at the same interval (2 hours) and duration (3 months) as those of NS records. We observed that 29.8% of the hostnames which were found to exhibit NS fluxing

were also undergoing fast fluxing (we refer to them as double fluxing phishing hostnames). The same thresholds of changes of IP addresses used in labelling the hostnames of the monitored hosts and the hostnames of the monitored NSs (described in sections 5.2.3.1 and 5.3.1.1 respectively) were also applied in this case (i.e 1 and 5).

Figure 5.43 shows the observed total number of changes of IP addresses of the hosts and the NSs of the double fluxing phishing hostnames. The figure indicates a high correlation of number of changes between the two fluxing behaviours. It is also interesting that the numbers of changes cluster around three distinct values. Figure 5.44 shows the cumulative number of unique IP addresses mapped against the double fluxing phishing hostnames at the end of the monitoring period. Again, the distribution indicates a high correlation between the two behaviours. It might be expected that a hostname with a high number of changes would also have a high number of unique IP addresses, but this correlation is less strong. We further investigated their cumulative number of IP addresses at various phases during the monitoring period. Figure 5.45 illustrates the results of this investigation by comparing the combined cumulative number of IP addresses of all FF phishing hostnames against those of all phishing NS flux hostnames. It can be seen that the two curves are very similar with both showing the periods of rapid and slow changes nearly at the same time. This suggests that attackers expand both flux networks at varying paces to support the varying demands of the current phishing campaigns or due to the availability of vulnerable machines.

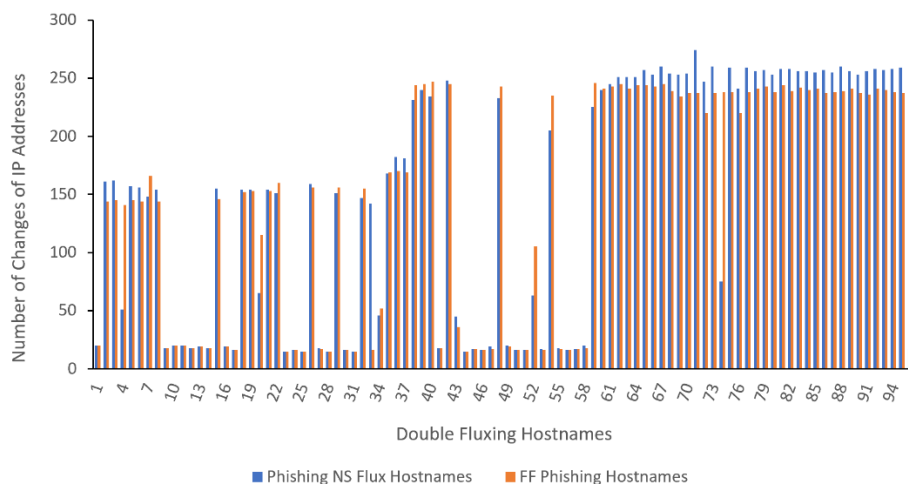


Figure 5.43. Number of changes of IP addresses observed in double fluxing phishing hostnames.

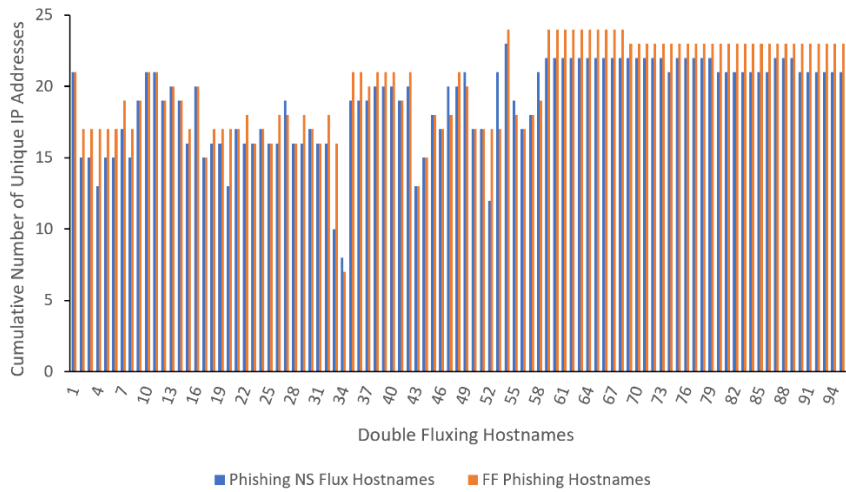


Figure 5.44. Cumulative number of unique IP addresses mapped against the double fluxing phishing hostnames.

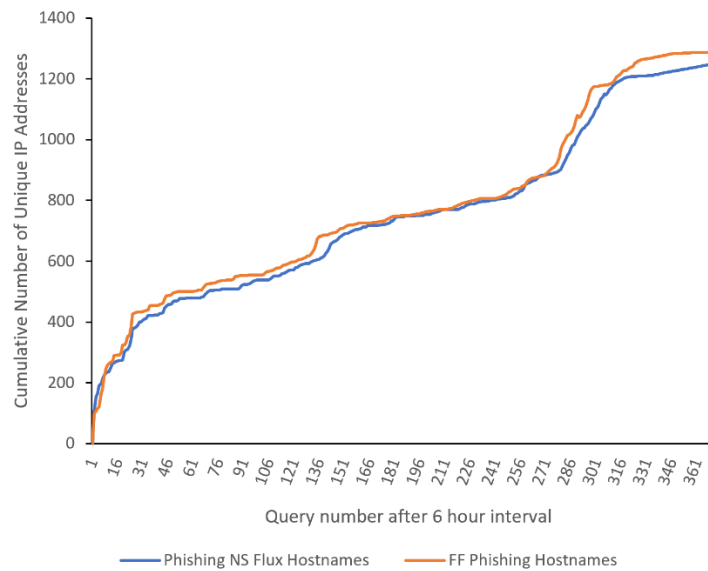


Figure 5.45. Cumulative number of unique IP addresses of double fluxing phishing hostnames at various stages of the monitoring period.

Due to high correlations in the three cases, we suspected that double fluxing phishing hostnames often use the same flux agents for both FFSN and NSIFN behaviours. To further investigate this, we compared the cumulative number of unique IP addresses of each pair of the same hostname to identify any shared addresses between its host web servers and NS servers. Unsurprisingly, we found that 65 of the 97 double fluxing hostnames (67%) were

sharing some of the addresses. Figure 5.46 shows the number of shared addresses between the two networks related to the same hostname. We observed that the number of the shared addresses was between 65% and 92% of the addresses associated with each pair. These observations suggest that FFSNs and NSIFNs hosting the majority of double fluxing phishing hostnames share a large number of flux agents. It is likely that the two such networks are often controlled by the same attacker or by different attackers exploiting most of the same vulnerable machines.

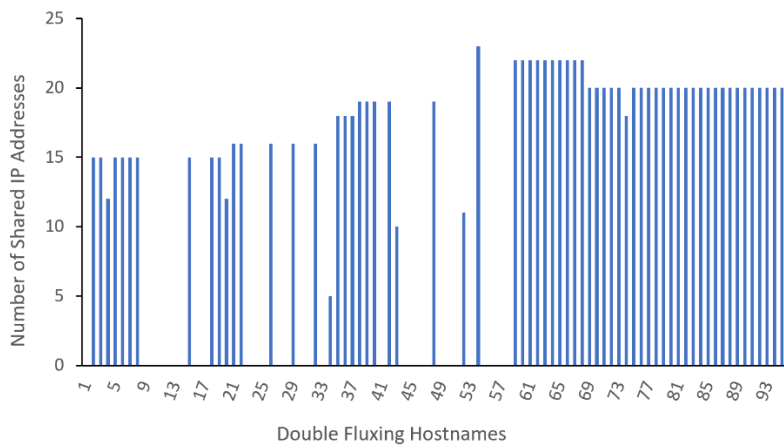


Figure 5.46. Number of the shared IP addresses between FFSNs and NSIFNs hosting the double fluxing phishing hostnames.

In summary, the section has provided an insight into some of the characteristics of networks of NSs of the monitored hostnames. Most characteristics described here are in line with the expected ones described in section 2.3.3. Unexpectedly, we found that most NSs keeping DNS records of the phishing hostnames were active throughout the 3-month period. The analysis has indicated that the NSs of some of the phishing NS flux hostnames are widely distributed in terms of subnets, networks and ASs compared to the other hostname types, and are hosted in a larger number of unique countries compared than the other hostname types. These characteristics can be useful in distinguishing phishing NS flux hostnames from the rest. In section 5.3.5.6, we provide an analysis of some of the important distinctive characteristics of the three types of hostnames using the feature data extracted in section 5.3.5.3.

5.3.4. Features for Prediction of Phishing NS IP Flux Hostnames

The review of FFSNs and NSIFNs described in section 2.3.3 and the analyses of the networks based on the collected data in section 5.3.1.2 suggested several similarities between the two in terms of their structures and operations. Some of the similarities including the use of proxies (flux agents) to host multiple malicious websites, network and geographical dispersion of flux agents, low TTL of the hostnames and host characteristics of the agents. Based on these common characteristics, we adopt some of the features we used in the previous model to predict phishing NS IP flux hostnames. Table 5.12 lists the adopted features. A total of 26 features were selected from six feature categories based on their relevance to NSIFNs. With the exception of feature 11, which was also used by Kadir, et al. [142], the rest of the features are used to address this problem for the first time.

Feature #	Category	Feature Name
1	Temporal	Average Round Trip Time (RTT)
2		DNS response time for NS records
3		Average uptime of NSs
4		TTL of NS records
5	Spatial	Average number of hops between user and NSs
6		Average number of unique hops' countries between user and NSs
7		Average number of unique hops' continents between user and NSs
8		Number of unique countries hosting the NSs
9		Number of unique continents of NSs
10		Average geo-distance between user and NSs
11	DNS	Average number of unique A records per NS per single lookup
12		Average number of co-hosted websites per NS
13		At least one NS with a dynamic IP address
14	Network	Number of unique subnets of NSs
15		Number of unique networks of NSs
16		Number of unique ASNs of NSs
17		Number of unique AS organizations of NSs
18	Host	Ratio of available (up) NSs
19		Number of unique OSs of NSs
20		The most common OS

21		Installed with a webserver
22		At least one NS uses a proxy IP address
23	Reputation	Total number of occurrences of IP addresses of all NSs of a hostname in a list of IP addresses of blacklisted phishing websites
24		Average number of occurrences of IP addresses of all NSs of a hostname in a list of IP addresses of blacklisted phishing websites
25		Ratio of NSs of a hostname with their IP addresses matching with IP addresses of blacklisted phishing websites
26		Registrar of NS records

Table 5.12. A list of proposed features for predicting phishing NS IP flux hostnames.

5.3.5. System Architecture of the Prediction Model

The prediction model uses a supervised ML approach to train and develop a classifier that predicts phishing NS flux hostnames. A similar system architecture of the model to that of the previous model (see section 5.2.5) is used, consisting of the same steps for building the model and predicting a new hostname user is attempting to access. The main difference here is that we monitor the A records of NSs of the collected phishing and legitimate hostnames to label the hostnames (as described in section 5.3.1.1)

5.3.6. Experiments

A number of experiments were performed to build a prediction model that evaluates the features for predicting phishing NS flux hostnames. First, we designed three architectures for the model, based on flat and hierarchical classifications, and identified binary and multi-class classifiers building each architecture. We then ran two sets of experiments to evaluate the prediction performance of each classifier for each architecture. The first and the second experiments used eight traditional ML and three DL algorithms, respectively, the same algorithm sets used in the two previous models. The best set of features for the prediction task was determined for each classifier. The performances of all classifiers in each architecture were determined and then combined to obtain the overall performance of each architecture in predicting the flux hostnames. Finally, performances of the architectures were compared to determine the best performing architecture for the model. The same eight standard ML

performance metrics used in the previous two models were also used here to evaluate the classifiers and the architectures.

5.3.5.1. Experimental Setup

The same machine specifications and software tools used to develop the previous prediction model (see section 5.2.6.1) were also used here.

5.3.5.2. Flat and Hierarchical Classification Architectures of the Framework

Three approaches to distinguishing phishing NS flux hostnames from phishing and legitimate NS non-flux hostnames were considered (see Figures 5.47a-c and Table 5.13). The first (Architecture X) performs a single-step three class classification, the second (Architecture Y) performs a binary classification distinguishing phishing NS flux hostnames from the rest combined as one class, and the third (Architecture Z) takes a hierarchical approach in which phishing hostnames are first distinguished from legitimate hostnames and then the phishing NS flux hostnames are distinguished from phishing NS non-flux hostnames. As in the previous model, LCPN and the selective classifier approach were used.

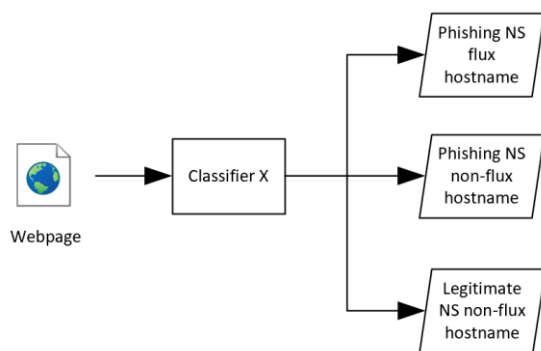


Figure 5.47a. Architecture X - flat classification-based architecture.

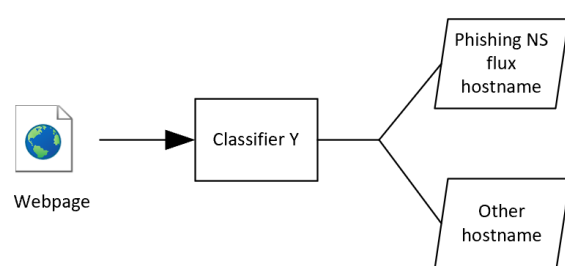


Figure 5.47b. Architecture Y - flat classification-based architecture.

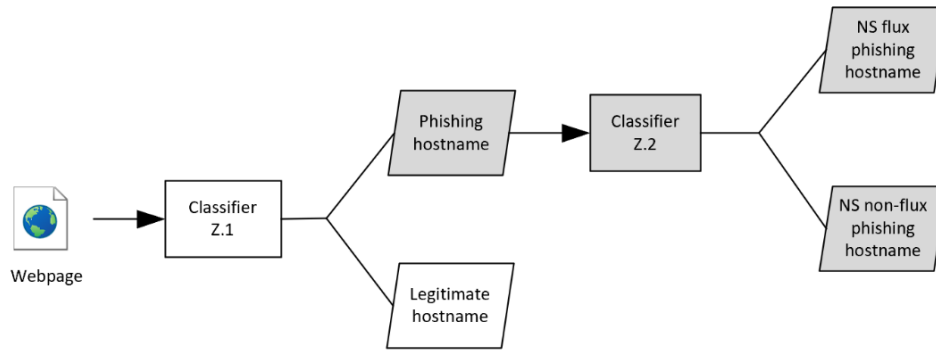


Figure 5.47c. Architecture Z – hierarchical classification-based architecture.

Classifier	Classifier Description	Classification Type
Classifier X	Classifies a hostname into three classes of hostnames; phishing NS flux hostname, phishing NS non-flux hostname and legitimate NS non-flux hostname	Multi-class classification
Classifier Y	Classifies a hostname as phishing NS flux hostname or other hostname (phishing NS non-flux hostname and legitimate NS non-flux hostname combined)	Binary classification
Classifier Z.1	Classifies a hostname as phishing or legitimate webpage	Binary classification
Classifier Z.2	Classifies a hostname as phishing NS flux or phishing NS non-flux hostname	

Table 5.13. Classifiers that are useful for the prediction of phishing NS flux hostnames in each architecture.

5.3.5.3. Training Dataset

The features described in section 5.3.4 were extracted from the labelled hostnames (defined in section 5.3.1.1) to create a training dataset. As indicated in Table 5.14, classifiers X, Y, Z.1 and Z.1 were trained with the full dataset whereas for classifier Z.2, all legitimate hostnames were removed from the dataset.

Classifier	Hostname Class Labels	Class Size	Dataset Size
Classifier X	Phishing NS flux hostname	326	13,268
	Phishing NS non-flux hostname	6,304	
	Legitimate NS non-flux hostname	6,638	
Classifier Y	Phishing NS flux hostname	326	13,268
	Other hostname	12,942	
Classifier Z.1	Phishing hostname	6,630	13,268
	Legitimate hostnames	6,638	
Classifier Z.2	Phishing NS flux hostname	326	6,630
	Phishing NS non-flux hostname	6,304	

Table 5.14. Classes and dataset sizes of training datasets used for the classifiers.

5.3.5.4. Data Pre-processing

The original dataset was pre-processed as follows. First, features with missing values were identified. Of the 26 features, 13 had missing values with percentages ranging from 1.6 to 38.8, which is within the acceptable range. Pearson’s correlation matrix was applied in order to determine redundant features. As indicated in the matrix chart in Appendix VIII, 6 features (feature 14 and 15, 16 and 17, 23 and 24) were found to have a high correlation value of at least 0.98 and therefore one feature from each pair (feature 15, 17 and 24) was dropped. Categorical features were then encoded as unique numeric values, with missing values given their own unique labels.

Four common imputation methods; mean, median, most frequent and k-NN (k=4) were compared using the RF algorithm. We found that the mean method produced the best results and therefore it was used to replace the missing values for all features. The SMOTE method was then applied to balance the datasets. Finally, the data was normalised such that all features have a mean value 0 and standard deviation of 1.

5.3.5.5. Performance Results

In this section, we present and compare results of the classifiers and the architectures.

A. Performance Results of Individual Classifiers Using Traditional Machine Learning Algorithms

The same approach used in the previous model (section 5.2.6.5-part A) was also used here to determine the best set of features, evaluate the features for each classifier using the eight traditional ML algorithms and tune hyperparameters of the best classifier. Figures 5.48a-d show the performances of ML algorithms in each classifier across all threshold values in the ROC curves. Tables 5.15a-b summarize results of the four best performing algorithms for each classifier. Again, the results indicate that RF yields the best performance across most metrics in each classifier. Overall, classifier Y's RF produced the best performance, with the tuned hyperparameter values as listed in Table 5.16.

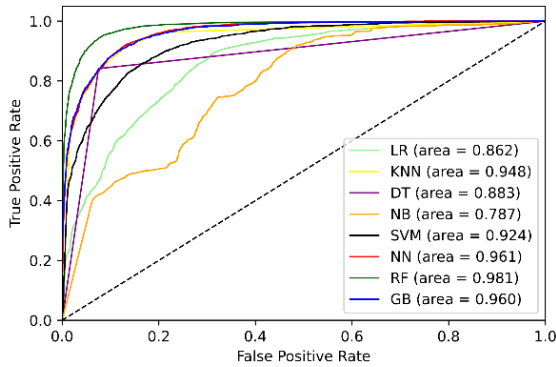


Figure 5.48a. ROC curves of the traditional ML algorithms for classifier X.

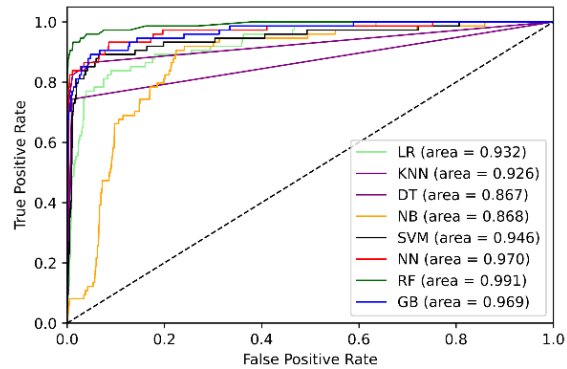


Figure 5.48b. ROC curves of the traditional ML algorithms for classifier Y.

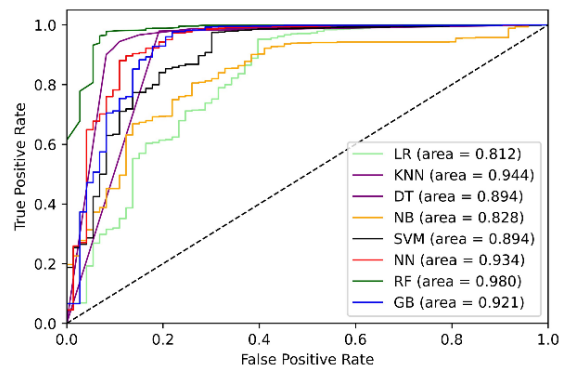
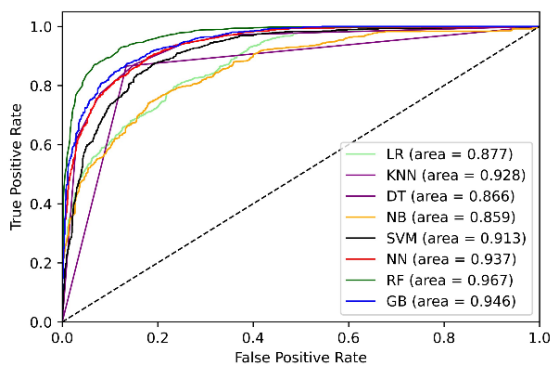


Figure 5.48c. ROC curves of the traditional ML algorithms for classifier Z.1.

Figure 5.48d. ROC curves of the traditional ML algorithms for classifier Z.2.

Algorithm	Classifier X				Classifier Y			
	Acc. (%)	FPR (%)	FNR (%)	F1 Score	Acc. (%)	FPR (%)	FNR (%)	F1 Score
KNN	84.29	9.34	16.31	0.85	97.21	2.56	9.50	0.98
DT	89.67	6.34	13.21	0.90	97.53	1.04	7.17	0.99
RF	90.41	5.92	12.41	0.90	98.59	0.76	5.29	0.99
GB	82.74	10.24	18.33	0.83	97.97	2.64	7.01	0.97

Table 5.15a. Performance results of top four best performing ML algorithms for classifiers X and Y.

Algorithm	Classifier Z.1				Classifier Z.2			
	Acc. (%)	FPR (%)	FNR (%)	F1 Score	Acc. (%)	FPR (%)	FNR (%)	F1 Score
KNN	85.43	15.29	13.85	0.85	94.95	14.72	4.55	0.96
DT	86.52	12.93	14.03	0.87	96.52	23.31	2.46	0.97
RF	90.19	10.29	9.34	0.90	98.42	11.17	0.57	0.98
GB	86.67	13.77	12.88	0.87	96.89	19.94	2.24	0.97

Table 5.15b. Performance results of top four best performing ML algorithms for classifiers Z.1 and Z.2.

Parameter	Description	Value
n_estimators	Number of trees	800
max_features	Max number of features considered for splitting a node	auto
max_depth	Max number of levels in each decision tree	32
min_samples_split	Min number of data points placed in a node before the node is split	5
min_samples_leaf	Min number of data points allowed in a leaf node	2
bootstrap	Method for sampling data points	false

Table 5.16. Values of the tuned RF hyperparameters which yielded the optimal performance.

B. Performance Results of Individual Classifiers Using Deep Learning Algorithms

We used the same approach as in the first model (section 4.5.4.2) to train each classifier using the three DL algorithms and tune their hyperparameters. The results in Tables 5.17a-b show that FC-DNN has the best performances in classifiers X and Z.1 whereas LSTM has outperformed the other algorithms in classifiers Y and Z.2. In overall, the LSTM in classifier Y achieved the optimal performance compared to the other algorithms in all four classifiers. Figures 5.49a-c show the three network architectures of best performing classifier (classifier Y) and its tuned hyperparameters as an example.

Algorithm	Classifier X				Classifier Y			
	Acc. (%)	FPR (%)	FNR (%)	F1 Score	Acc. (%)	FPR (%)	FNR (%)	F1 Score
FC-DNN	85.59	9.01	17.13	0.86	97.84	0.44	21.89	0.98
LSTM	81.94	11.59	26.09	0.82	98.51	0.25	16.60	0.98
CNN	74.39	16.74	28.23	0.74	98.40	0.42	21.49	0.98

Table 5.17a. Performance results of the evaluated DL algorithms for classifiers X and Y.

Algorithm	Classifier Z.1				Classifier Z.2			
	Acc. (%)	FPR (%)	FNR (%)	F1 Score	Acc. (%)	FPR (%)	FNR (%)	F1 Score
FC-DNN	86.43	12.20	15.55	0.86	95.77	0.89	23.01	0.96
LSTM	85.25	14.39	24.03	0.86	97.54	0.65	18.63	0.98
CNN	78.91	14.92	19.43	0.79	97.40	0.71	21.05	0.98

Table 5.17b. Performance results of the evaluated DL algorithms for classifiers Z.1 and Z.2.

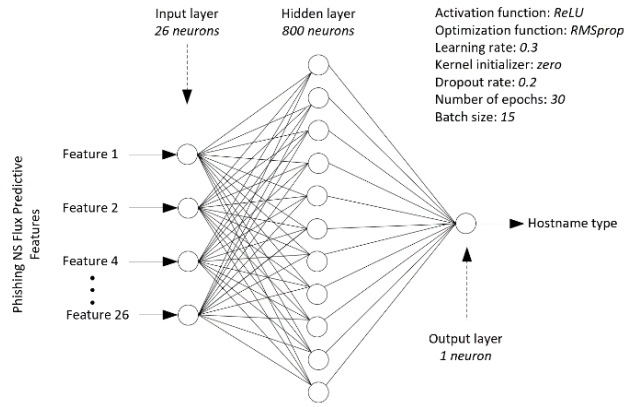


Figure 5.49a. FC-DNN architecture of classifier Y.

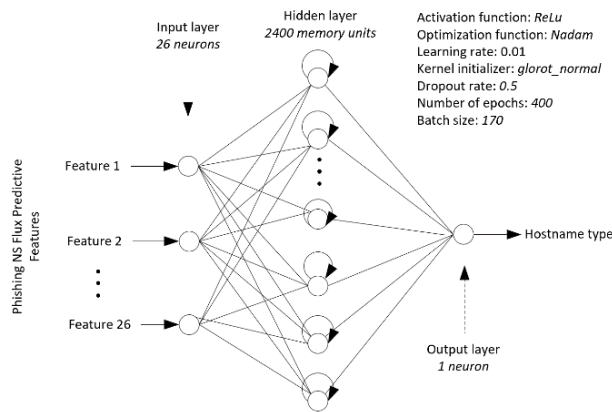


Figure 5.49b. LSTM architecture of classifier Y.

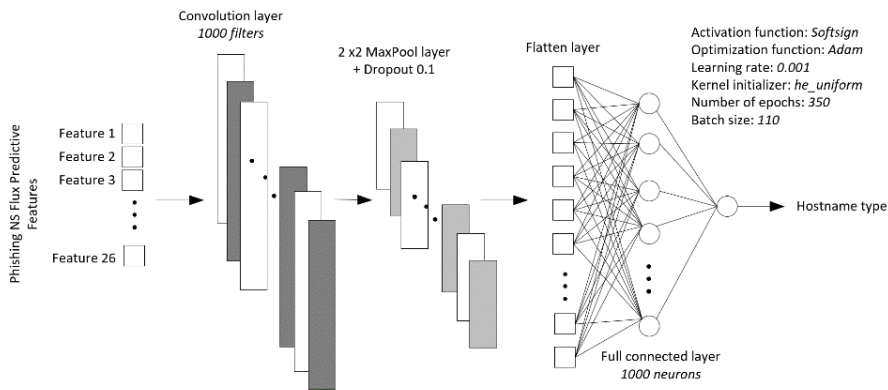


Figure 5.49c. 1D CNN architecture of classifier Y.

C. Overall Performance Results of the Proposed Architectures

Table 5.18 below summarizes the best performance of each architecture. To obtain the performance of architecture Z, RF performances of classifiers Z.1 and Z.2 were combined as

was done previously for Architectures C and D (see section 5.2.6.5 – part C). Architecture Y achieved the best performance across all metrics. We therefore conclude that the architecture Y is the most effective one for implementing the model.

Architecture	Acc. (%)	FPR (%)	FNR (%)	Classification Type
X	90.41	5.92	12.41	Multi-class
Y	98.59	0.76	5.29	Binary
Z	88.76	21.46	9.91	Multi-class

Table 5.18. Prediction performances of the model architectures.

5.3.5.6. Feature Performance Analysis

This section studies the importance of feature categories and individual features to the performance of the best performing classifier (classifier Y). The feature selection performed using RF algorithm resulted in 11 best features for the classifier (shown in Figure 5.51), all of which are proposed for the first time for this problem and are third party based features.

The feature which checks whether the NS hosts have webserver software installed (feature 21 in Table 5.12) is the most influential. Figure 5.51a shows that that nearly 70% of NS hosts of the phishing NS flux hostnames have webserver installed while phishing NS non-flux hostnames have almost 40% and legitimate NS non-flux hostnames have less than 5%. Even after combining the two NS non-flux hostnames for classifier Y, the percentage in the two is still almost a half of the percentage in the NS flux hostnames (see Figure 5.52b). Assuming equal numbers of each hostname type in the samples, then if only this feature is known, then a hostname with a ‘Yes’ value is about 1.6 times more likely than not to be a phishing NS flux hostname in the three-class case and about 3.3 times more likely to be a phishing NS flux hostname. It is plausible that most of the NSs of phishing hostnames are co-hosted with malicious websites, whereas legitimate NSs mostly run on dedicated servers. The small percentage of legitimate NSs with webserver may have been compromised by attackers to host their malicious web services.

The second ranked feature is the average number of websites co-hosted in the NS servers of a hostname (feature 12). Its data distributions in Figures 5.52a-b correlate with those in the previous feature and thus confirms our suspicion. We think attackers use their servers to host both NS software and multiple malicious websites because; (1) their websites expect a low volume of queries from users compared to the established legitimate websites, thus computing resources required are low, and (2) attackers prefer to use minimum number of resources to launch many attacks so as to maximize their profits.

Average uptime of NS hosts of a hostname (feature 3) is the third in the ranking and the highest ranked temporal feature. Looking at the data distribution in Figure 5.53a, the median uptime of the hosts of NSs for phishing NS flux hostnames is higher than that of the other classes, but the distribution is much more concentrated at low values. Very few phishing NS flux hostnames have average uptimes above 5 million seconds whereas a significant minority of other hostnames have uptimes above this value and some exceed 30 million seconds. Based on the median values, it appears that a subject with a very low average uptime is likely to be a legitimate NS non-flux hostname, the one with a medium average uptime is likely to be a phishing NS non-flux hostname and the one with a high average uptime is likely to be a phishing NS flux hostname. A similar pattern can be observed in the binary classification case in Figure 5. 53b.

Average number of network hops between user and NS hosts of the same hostname (feature 5) ranks at the 5th position. Figures 5.54a-b show its data distribution in three and two class classifications respectively. Based on the median values and the density distributions, NS hosts of phishing NS flux hostnames tend to be located at a larger number of hops from the user than the other two classes of hostnames, with legitimate NS non-flux hostnames have the smallest median. Similar patterns occur in other spatial features ranked at 4th, 6th and 11th positions. Our data, therefore, agrees with other studies that NS flux agents are geographically dispersed compared to NSs of other hostnames. This is consistent with the expectation that the malware controlling the flux networks recruits vulnerable computers from as many networks as it can and in a random manner.

Registrar of NS records (feature 26) is the only reputation-based feature in the ranking and is at the 8th position. As shown in Figures 5.55a-b, some of the registrars including Namecheap

Inc., R01, PDR Ltd, eName Technology Co. Ltd, Dynadot LLC and Internet Domain Service BS Corp register a large percentage of NS records of both phishing NS flux and non-flux hostnames compared to other registrars. This suggests that some of the registrars have less strict measures in validating owners of the records at the time of registration and in monitoring uses of the records, allowing attackers to take advantage of their platforms.

TTLs of NS records (feature 4) is at the 10th position in the ranking. Figures 5.56a-b show the distributions of its data. Most of the phishing NS flux hostnames have a TTL of 600 seconds followed by 86400 and 7200 seconds whereas most of both types of NS non-flux hostnames have TTLs of 86400 and 3600 seconds. It is to be expected that NS flux hostnames use short TTLs to ensure that frequently changed NS records are returned to users, when queried, from their authoritative NSs instead of the cached records, thus maintaining the fluxing behaviour of the NS networks.

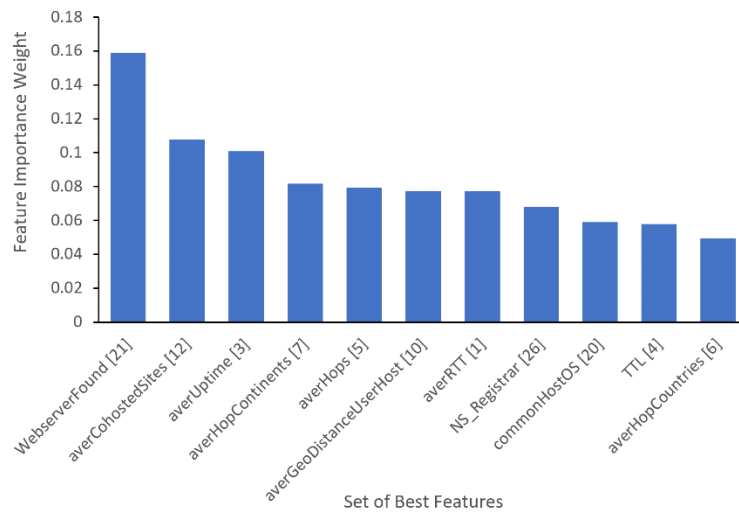


Figure 5.50. Ranking by importance weights of the best features of classifier Y. Numbers in the brackets represent numbers of the features as indicated in Table 5.12.

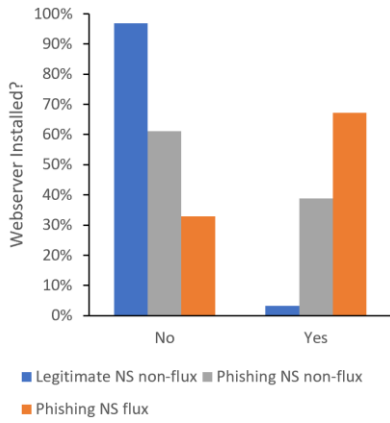


Figure 5.51a. Percentage of hostnames per each class with their NSs installed with webservers.

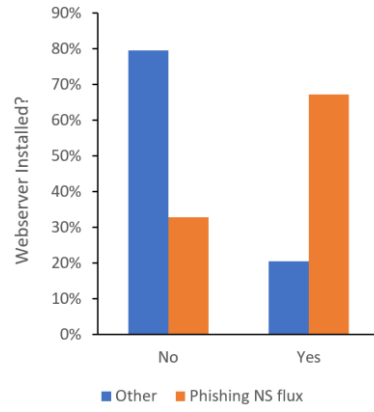


Figure 5.51b. Percentage of hostnames of classifier Y with their NSs installed with webservers.

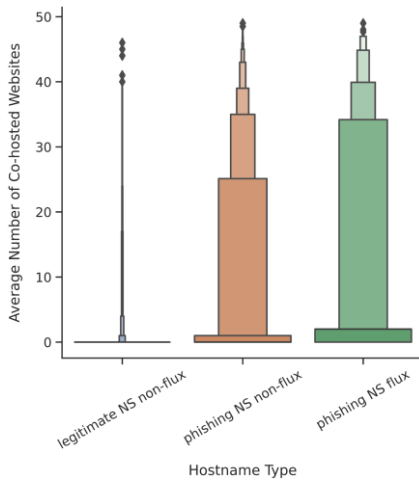


Figure 5.52a. Distribution of average number of unique co-hosted websites in the hostname's NSs for each hostname class.

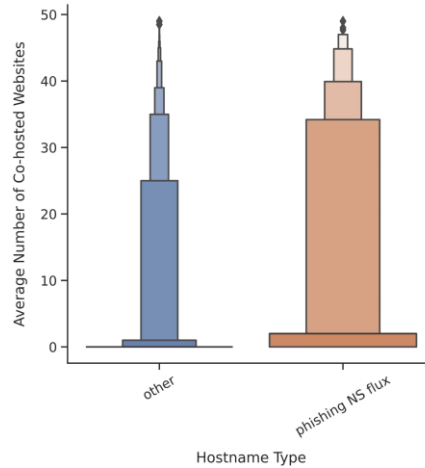


Figure 5.52b. Distribution of average number of unique co-hosted websites in the hostname's NSs for the two classes of hostnames in classifier Y.

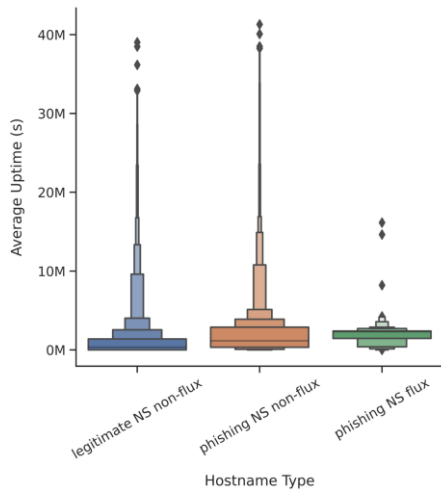


Figure 5.53a. Distribution of average uptime of NSs of hostnames per each class.

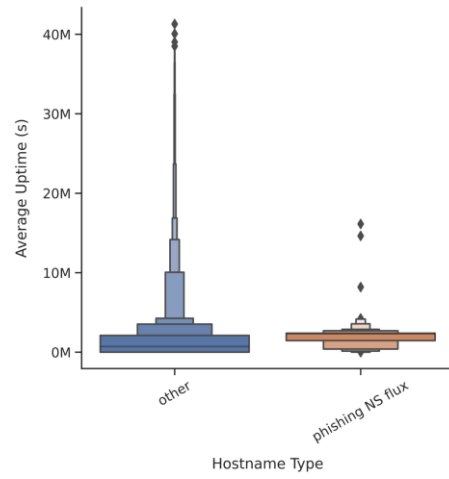


Figure 5.53b. Distribution of average uptime of NSs of hostnames in classifier Y.

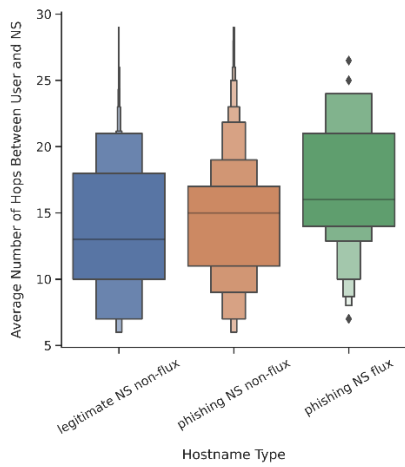


Figure 5.54a. Distribution of average number of network hops between user and NS hosts of the three hostname classes.

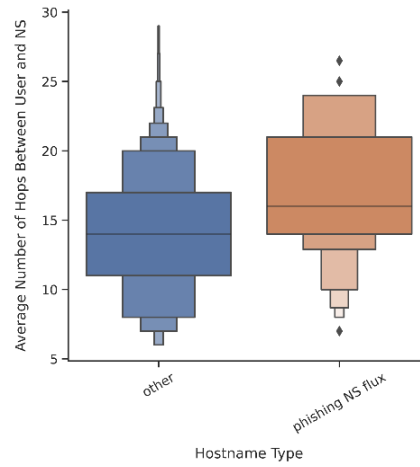


Figure 5.54b. Distribution of average number of network hops between user and NS hosts of the two classes of hostnames in classifier Y.

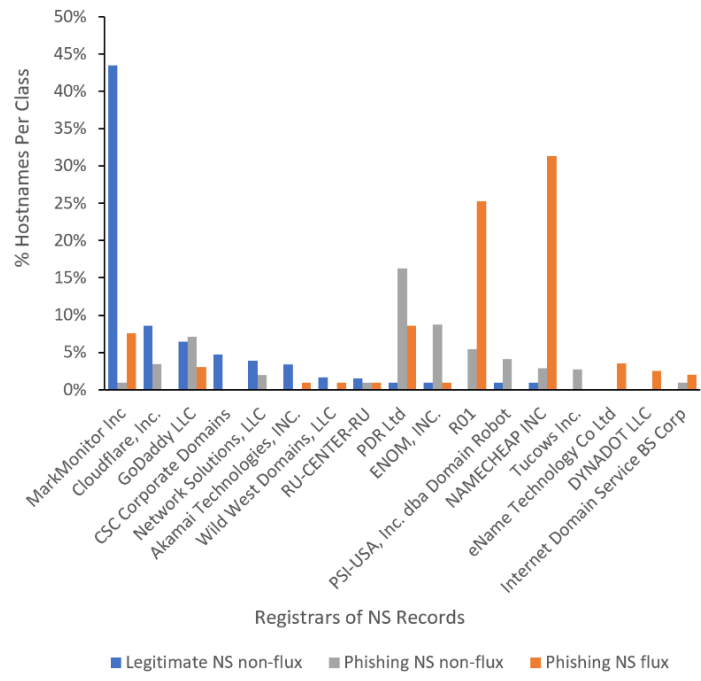


Figure 5.55a. Distribution of registrars of NS records of the three classes of hostnames.

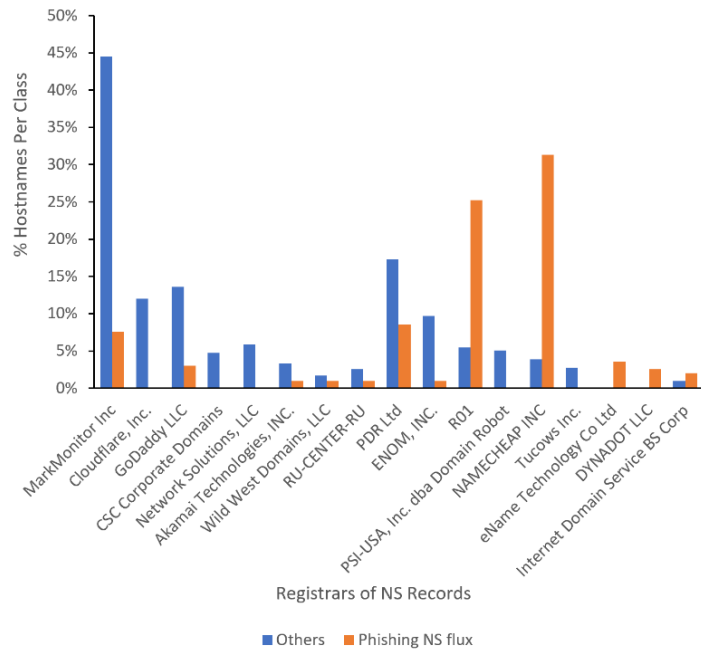


Figure 5.55b. Distribution of registrars of NS records of the two classes of hostnames in classifier

Y.

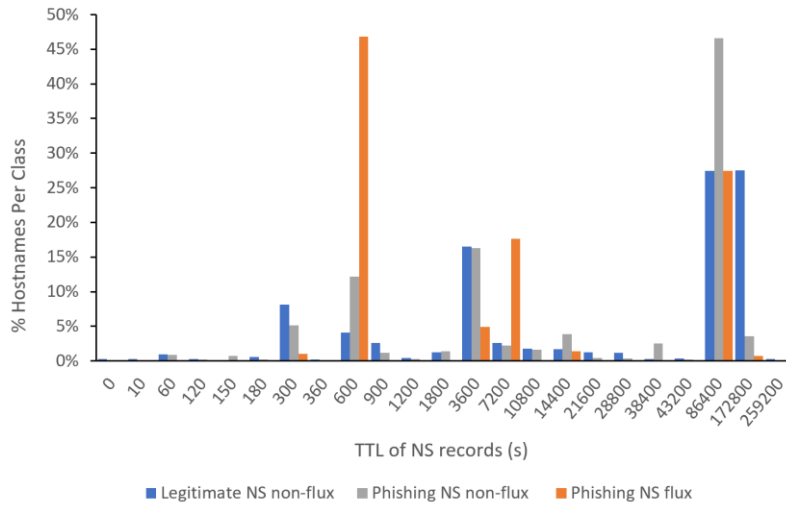


Figure 5.56a. Percentage distribution of TTLs of NS records of hostnames for the three hostname classes.

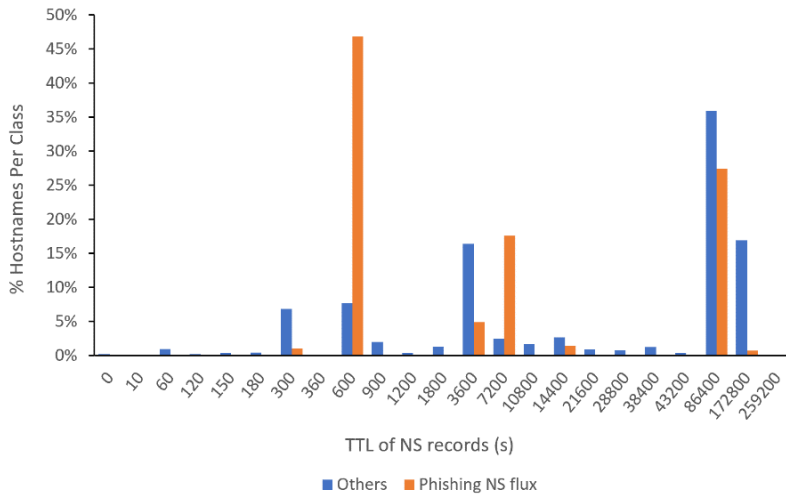


Figure 5.56b. Percentage distribution of TTLs of NS records of hostnames in classifier Y.

The composition and the performance contribution of each feature category in the best feature subset of classifier Y was also analysed. Table 5.19 provides a breakdown by category. Temporal and spatial categories have the largest number of features in the best feature set. DNS, host and reputation-based features contributed slightly less than a half of their proposed features while none of the network features made into the set of best features. This is consistent with the performance of the individual categories when run on the tuned classifier Y (shown in Figures 5.57). Spatial features yielded the highest accuracy, the lowest FPR and the second lowest FNR. Temporal and DNS features produced accuracies and FPRs slightly lower than those of spatial but generated the highest FNRs. Reputation, on the other hand, produced the

lowest accuracy and FNR but the highest FPR. None of the categories produced the best performance across all three metrics.

Feature Category	Tally of Full Features	Tally of Best Features	Best Features # (# from Table 5.12)
Temporal	4	3	1, 3, 4
Spatial	6	4	5, 6, 7, 10
DNS	3	1	12
Network	4	0	-
Host	5	2	20, 21
Reputation	4	1	26

Table 5.19. Composition of categories in the best feature set of classifier Y.

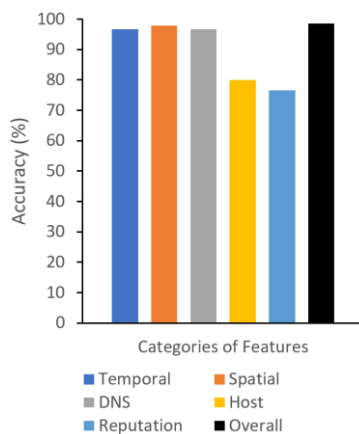


Figure 5.57a. Comparison of accuracy rates of feature categories of classifier Y.

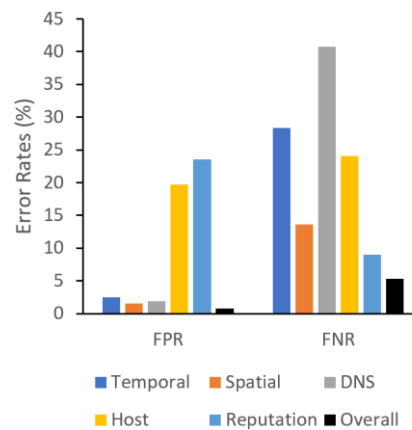


Figure 5.57b. Comparison of error rates of feature categories of classifier Y.

5.3.5.7. Detection Time Analysis

The runtimes of the Classifier Y's main phases (i.e webpage downloading from its server, PDC webpage filtering, hostname retrieval, feature extraction, dataset training and prediction) were measured to understand its efficiency in detecting phishing NS flux hostnames (see Table 5.20). The detection time (sum of the first five phases in Table 5.20) of 6 seconds is almost entirely due to the retrieval of feature data from their sources. Figure 5.58 shows the breakdown of times of activities related to extraction of the best features. HTTP header request for checking

the presence of a webserver, which is the strongest predictor, takes the longest time. Host scanning using Nmap for extracting uptime and common OS features (at the 3rd and 9th positions in Figure 5.48) took the second longest time while NS records queries for extracting TTL (10th position) took the shortest time. As the strongest predictors took the longest times to extract, there is little scope for improving the detection time by removing the least important features from the set.

Phase	Time (s)
PDC webpage loading	0.8430
PDC webpage filtering	0.0976
Hostname retrieval	0.0001
Feature extraction per webpage	5.0571
Prediction per webpage	0.0001
Detection Time	5.9977
Training the dataset	11.010

Table 5.20. Runtimes of the classifier Y’s three stages for predicting phishing NS flux hostnames.

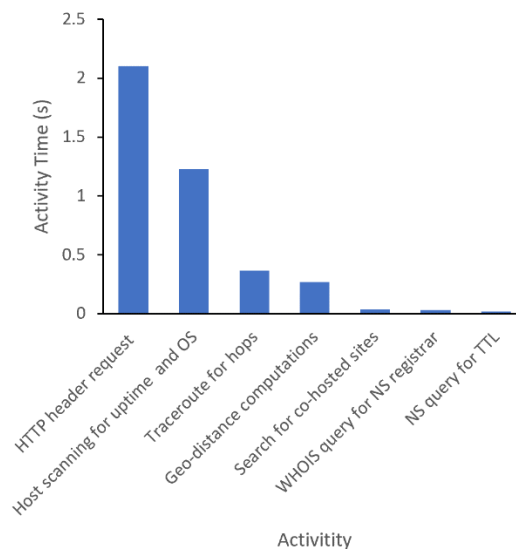


Figure 5.58. Distribution of feature extraction times by activities.

5.3.5.8. Model Validation Using New Data

As in the two previous studies (see sections 4.5.6 and 5.2.6.8), performance consistency of the best classifier was checked using a new dataset collected over a different period. The new testing dataset of 1,413 legitimate and 1,398 phishing websites was collected between 10th of January and 15th of March 2021. After performing similar data pre-processing as described in section 5.3.5.4, classifier Y was tested against the new dataset. It achieved an **accuracy of 97.18%**, **FPR of 1.85%** and **FNR of 6.09%**, which is slightly less good than those obtained with the original dataset on which the classifier was trained. The differences of 1.41% in accuracy, 1.09% in FPR and 0.8% for FNR were noticed. We plan to conduct further validation tests using other new datasets collected in other separate periods in the future to learn more about the performance consistency of the classifier, including whether it is due to a change in the tactics of phishers over time. Despite the differences, the testing performance is still acceptable and within the range reported by most works in the domain of detection of IP flux networks (see appendices VI and VII).

5.3.7. Discussions

5.3.7.1. Comparison with Existing Works

In this section we compare our work with the existing works proposing approaches for detecting malicious NS flux hostnames (described in section 5.3.1 and appendix VII). Only two such works, Kadir, et al. [142] and Pa, et al. [140], were found. Table 5.21 below summarizes key aspects of the comparison. The major difference between our work and these ones is the detection time. While our work uses feature data collected at a single point in time for prediction, which takes only 6 seconds, the other two works collected feature data for 3 and 6 months. The proposed approaches, not only are unsuited for instant detection, but also allow sufficient time for fluxing NSs under investigation to continue operating and serving malicious web services.

The other difference is the diversity of the features used. The two works have used only DNS based features for the prediction while our work has used 5 different feature categories. As we have pointed out in the two previous studies, the use of only a small number of feature categories increases the risk of detection evasion compared to the large number of categories.

For instance, Kadir, et al. [142] used the total number of unique IP addresses of NSs and their fluxing rates during the observation period as the features. As observed in our data (see section 5.3.1.2), some of the legitimate NSs also change their IP addresses but at a lower rate compared to those of phishing NS flux networks. However, the attacker may opt to lower the fluxing rates and/or the number of unique IP addresses of NSs to a range similar to those of legitimate NSs in order to increase the error rates, thus reducing the detection effectiveness.

The time interval between consecutive DNS queries for A records of NSs during the NS monitoring phase is another significant difference. While Pa, et al. [140] did not mention their time interval, Kadir, et al. [142] queried the A records once after every 12 hours. For the reason mentioned in section 5.3.1.1, the records in our study were queried every 2 hours. Given our smaller time interval compared to the Kadir, et al. [142]’s work, our work is likely to have captured more precise information about the NS fluxing behaviours and therefore should be able to make better predictions.

Work	Feature # and Categories	Data Size (URLs)	Classification Type	Evaluation Algorithms	Performance	Detection Time
Kadir, et al. [142]	7 DNS features	500	Binary classification (NS IP flux hostnames versus benign hostnames)	k-NN	FPR = 0% FNR = 0%	3 months
Pa, et al. [140]	3 DNS features	50,030	Binary classification (NS IP flux hostnames versus non-NS IP flux hostnames)	Mappings of IP and hostnames of NSs	FPR = 0.8%	6 months
Our Work	3 temporal, 4 spatial, 1 DNS, 2	13, 268	Binary classification (phishing NS IP	LR, k-NN, DT, NB, SVM, ANN,	Acc = 98.59% FPR = 0.76% FNR = 5.29%	6 seconds

	host and 1 reputation features		flux hostnames versus phishing non-NS IP flux hostnames)	RF, GB, FC-DNN, LSTM, 1D CNN	Prec. = 0.99 Recall = 0.98 F1 = 0.99 AUC = 0.99	
			Multi-class classification (phishing NS IP flux hostnames versus phishing NS IP non-flux and legitimate NS IP non-flux hostnames)		Acc = 90.41% FPR = 5.92% FNR = 12.41%	

Table 5.21. Performance comparison of some of the related works with our work.

5.3.7.2. Application of the Proposed Model

The proposed model can be of use in several ways. One of the applications is to protect users from visiting PDC webpages whose DNS records are hosted in the phishing flux NSs. This can be achieved by, for instance, incorporating the model in a web browser or network gateway application to filter out any PDC webpage that a user is attempting to access. The current detection time can be reduced by extracting features in parallel rather than in sequence (the current implementation). With this approach, the detection time will drop to 3 seconds, a sum of processes prior to feature extraction and the longest time to extract a feature (2.1 seconds) as indicated in Figure 5.58. As described in 4.6.2, this time is feasible for real time protection of users.

Note that the model proposed by Kadir, et al. [142] produced no prediction errors, albeit at the cost of a long detection time. To take advantage of their accurate detection with the fast detection capability of our model, the two models can be combined in parallel in order to improve the overall prediction task. In this approach, our model will provide instant detection of most phishing NS flux hostnames with a few errors while their model will provide a more accurate detection in long run. This will reduce the number of phishing NS flux hostnames which will remain operating before being detected by the second model.

Similar to the two previous models, our model can also be used to build a blacklist of phishing NS flux hostnames. Currently we are not aware of any existing blacklist of NS flux hostnames. The blacklist can be a useful resource to security researchers, vendors and authorities for further investigation of phishing NSIFNs and the development of various tools to address the networks and their hosting web services. Lastly, the multi-class model using architecture X can be used to distinguish between phishing NS flux hostnames from phishing NS non-flux hostnames as suggested in the previous model (see section 5.2.7.3).

5.4. Summary

In this chapter, we have proposed two novel sets of predictive features for building two models, based on the supervised ML approach, that predict phishing hostnames hosted in IP flux networks. The first model predicts hostnames hosted in FFSNs and the second one predicts those hosted in NSIFNs. The first model was built using 41 novel features and 15 features that were adopted from previous studies, all grouped into six categories. Four model implementation architectures based on binary and multi-class classification approaches were proposed and evaluated using eight traditional ML and three DL algorithms. The binary classification-based architecture distinguishing FF phishing hostnames from phishing non-flux, CDN and legitimate non-flux hostnames, combined as a single class, was found to be the most accurate architecture for implementing the model. The four-class multi-class classification-based architecture was observed to be a slightly less accurate than the former. However, this architecture is useful in detecting the exact type of a hostname, allowing a more informed decision to be taken to address the attacks. We also investigated the prediction importance of the proposed features in the context of the best performing model architecture and found that temporal and DNS related features are the strongest predictors while network and host related features are the weakest. The proposed model has delivered a higher detection performance than most of the related works. This suggest that our novel feature set, which is also highly diversified, is as effective as others but is also expected to increase the resistance of the solution to detection evasion. We have addressed the problem in a four-class classification context, thus our work is more realistic in addressing the real-world problem than other works that addressed the problem in a three-class classification context. We have also reported our results using a wider range of performance metrics, informing us on the all-round effectiveness of the model.

The second model was built using 11 features, grouped in five categories, all of which are proposed for the first time in this problem. Three model implementation architectures based on binary and multi-class classification approaches were proposed and evaluated using eight traditional ML and three DL algorithms. The results showed that the binary classification-based architecture distinguishing phishing NS flux hostnames from phishing NS non-flux and legitimate NS non-flux hostnames, combined as a single class, was the most accurate architecture for the model. The three-class multi-class classification-based architecture, which is useful in identifying a specific hostname type of the three hostnames, was found to be less accurate than the former. The prediction importance of the proposed features in the best performing architecture was investigated and showed that features in spatial and temporal categories are the strongest predictors while network related features do not have any impact in the prediction. The proposed model has delivered a high detection performance comparable to other similar works in the literature, suggesting that our novel features deliver more or less the same effectiveness as the existing ones. However, unlike the existing approaches, our approach has used more diversified features for improving resistance to detection evasions, has achieved fast detection for real time applications and has addressed the problem as a three-class classification problem, thus a more pragmatic solutions to the actual problem. Also, a wider range of performance metrics were used to report the results, thus affirming the reliability of the solution.

It is important to note that the performances obtained in this work depend on the nature of datasets used for training the models. Attackers are likely to vary configurations of their flux networks over the time which may result in variations in the effectiveness of some of the detection features, and indeed the models. We believe that continuous observations of behaviours of the flux networks and re-evaluation of the models using new periodically collected datasets are important to ensure that detection performance remains at a high level.

These two models can be implemented to complement the model we have proposed in Chapter 4, that is, the three can be run concurrently as independent applications. As suggested previously, for developing applications of these models for real world uses, we recommend that sizes of their training datasets should be increased to a much larger sizes in an attempt to improve further their performances.

Chapter 6

Conclusions and Future Work

6.1. Summary of a Research Background

Today, phishing is widely regarded as the biggest security threat to the internet community. Phishing websites, delivered mainly through phishing spams, have been the major channel used by attackers to acquire personal data from users for performing malicious activities. In an attempt to address phishing website attacks, various solutions have been applied to prevent the websites from being delivered to their targets. These include anti-spam email filters, email authentication standards and phishing awareness training programs. Though these solutions have played a significant role in reducing the number of attacks reaching to users, they have not been able to entirely prevent the distribution of the websites, allowing a significant number of phishing websites to be accessed by users. This is due to the attackers' adoption of various techniques to elude detection and human fallibility in identifying the phishing spams.

This raised the need to have highly effective solutions to prevent users from accessing the websites once the spams are successfully delivered to the users. Anti-phishing filters built into web browsers are the most commonly used solutions in this category. Research works have also proposed a number of phishing webpage detection solutions, the most popular being the ones using ML. Most of the existing solutions being applied in this area depend on blacklists of known phishing websites, a technique which performs well in detecting known phishing websites but has a poor instant detection rate against the zero-day websites. In spite of the solutions, there has been a sharp increase in the number zero-day phishing websites with detection evasion capabilities. This is due to the growing use of sophisticated phishing toolkits and flux networks for creating and hosting the websites respectively. This has led to the rise in the number of successful phishing website attacks, making the websites the most significant phishing threat today.

Since the blacklist approach is not effective in addressing the zero-day websites, a different approach that can accurately and instantly detect zero-day phishing websites is required. In order to achieve the objective, the approach must not rely on a blacklist, achieves a high detection rate, performs the detection in real time, focuses on phishing-specific websites to

optimize detection, and uses novel and highly diversified features to make the solution ahead of the attackers and more resistant to detection evasion. ML has shown the potential to meet these design goals, given appropriate data and prediction features. Aiming to meet the goals, this research has developed ML-based approaches for fast and highly accurate prediction of zero-day phishing webpages using novel sets of highly diversified features. The following section summarizes contributions made from this research.

6.2. Research Contributions

First, a method for identifying PDC webpages as a webpage pre-prediction filtering process is proposed in *Chapter 4*. This method identifies the webpages by detecting an HTML form or script-based dialogue window and at least one of the 43 common phrases, contained in the webpage structure or contents, that label the type of personal data being collected. The method is useful in filtering out non-PDC webpages from the prediction process, thus avoiding web browsing overheads due to the prediction analysis when accessing these webpages and the possibility of producing false positives on the webpages.

Second, in *Chapter 4*, we have identified 26 features that can distinguish zero-day phishing PDC webpages from legitimate ones with high prediction performance using ML. The features, of which 20 are introduced by this study and the rest were adopted from previous works as their best features, are derived from five different feature categories namely URL structure, webpage structure and contents, webpage's digital certificate, domain registrar's records, and webpage reputation in the search engines and blacklist of phishing websites. The features performed well against data collected at over a different time period without retraining the data, indicating their long-term ability in identifying the webpages created by attackers at different times. The measured prediction time was found to be sufficiently low for potential use for real-time protection of users. Our feature analysis has indicated that the novel and third party-based features are stronger predictors than the adopted and local (extracted from the URL and webpage) based features. Also, most of the features based on the webpage reputation against blacklisted phishing IP addresses and search engines are the most influential ones for the task while those based on URL structure are among the least influential ones.

Third, in *Chapter 5*, we have identified a set of 56 features that are capable of distinguishing zero-day phishing hostnames hosted in FFSNs from those not hosted in FFSNs, and from legitimate hostnames hosted and not hosted in CDNs with high prediction performance and low prediction time using ML. The features, of which 41 are novel and 15 were adopted from previous works, are categorized in six feature categories, namely temporal, spatial, DNS, network, host and reputation-based features. The features were found to have long-term effectiveness when the evaluation was repeated with respect to an independent data set collected at a later time, without retraining. It was shown by our feature analysis that novel and third-party features are more influential in the prediction compared to adopted and local features. The analysis also indicated that temporal and DNS feature categories are strong predictors while network and host categories are the weakest ones. In terms of the individual features, the majority of features based on reputation are the most influential in the prediction whereas the majority of network features are the least influential.

Fourth, in *Chapter 5*, we have identified a set of 11 features that can distinguish zero-day phishing hostnames hosted in NSIFNs from those hosted in non-NSIFNs and legitimate hostnames with a high prediction performance using ML. The features, which are all based on third party information, are classified in five different feature categories namely temporal, spatial, DNS, reputation and host. The features produced good prediction performance without retraining when evaluated against phishing webpages collected at a later time. This suggests that the features have a long-term ability to identify the hostnames created by attackers. The low prediction time measured suggests that the approach can potentially be used to protect users in real-time. Weighing the significance of the features for the task, we observed that spatial based features collectively are the most influential predictors whereas host and reputation are the least influential. The presence of webserver software in name servers was found to be the strongest individual feature while the average number of unique countries of network hops between user and name servers is the weakest.

Fifth, we have taken an approach of evaluating the features sets in each prediction task using a large number of different ML algorithms. Given that each algorithm relies on different data distribution assumptions and uses a different statistical approach in creating the prediction rules, comparing the performance results from such a large set of algorithms allows us to draw conclusions on the general effectiveness of a feature set for a specific prediction task. As

observed in our experiments, the feature set in each task produced good prediction performance across all metrics when they were evaluated using most ML algorithms. This affirms the relevance and robustness of the feature sets for their respective tasks. In addition, the DL algorithms CNN and LSTM were also used for evaluation for the first time in this domain. The algorithms have all produced good results especially in the first and second prediction tasks, outperforming a number of traditional ML algorithms which produced the best performances in the related works. This suggest that future research into this problem needs also to explore the potential of new algorithms to improve detection performance as the ML domain evolves and new algorithms are developed.

Sixth, a methodological contribution is presented in *Chapter 5*. Besides evaluating the features in both prediction tasks using a binary classification approach, we also evaluated them using a multi-class classification approach for the first time in this context. In the binary classification approach, phishing hostnames hosted in FFSNs or NSIFNs were distinguished from the rest of the hostnames combined as single hostname class. The multi-class approach is a more difficult task from the ML perspective than the former since it predicts each hostname type as an independent outcome class. The relative performance of the two in both tasks illustrated this by showing that the multi-class classification approach performed a slightly less well than the binary one. However, the multi-class approach has still produced a good prediction performance comparable to related works, which are all based on the binary classification approach. This suggests that the approach can potentially be used for real world applications. One of the possible applications is to identify the exact types of hostnames being evaluated. For instance, security experts may use the approach to differentiate between phishing hostnames hosted in flux networks from those hosted in non-flux networks in order to take appropriate measures to take down the websites through their hosting infrastructures.

Our last contribution, described in *Chapter 5*, is the proposed approach to label a training dataset based on the changes in the IP addresses returned by DNS queries about a particular name server monitored over time. The instantaneous features are then used to classify the hostnames. This approach has resulted in shorter classification time than would be the case if features derived directly from monitoring behaviour over time were used for classification.

In conclusion, this study has achieved its research aim by identifying relevant sets of features and illustrating their ability of accurate and fast prediction of phishing webpages and hostnames hosted in flux networks using the ML approach. The proposed prediction models have also met the design goals to a large extent as described below;

- The models have used the ML technique thus eliminating the dependency on blacklists of phishing webpages.
- Though the final performance in each prediction task is slightly lower than the target mentioned in the design goals (in Chapter 1), our proposed features have still yielded better results than those proposed by most related works. However, there is still a room for improvement on our performances through, for instance, increasing the training dataset size and exploring other thresholds or techniques for labelling the hostnames.
- The prediction times vary significantly in the three prediction tasks where the times for the first and third tasks are significantly lower than the second one. This is because the number of third-party features, which are more time consuming to extract, is greater in the second task. By improving the way the features are extracted, that is, extracting them in parallel rather than sequentially, the times for the first and third prediction tasks will be reduced to real time in the context of speed of webpage access. This implies that the prediction models for the two tasks can be used to protect users in real time. This is not the case, however, for the second model for which the feature extraction time is above the real time range.
- Because the features are extracted from phishing websites only, and not from other malicious web services, the prediction models are more accurate in identifying phishing specific websites.
- Most features in the first and second prediction tasks and all the features in the third prediction task are novel ones. Also in each task, a larger number of different feature categories was used than by any related work, thus our work has the most diversified sets of features.

6.3. Limitations of Our Work

Almost a half of our proposed best features in the first prediction model, most features in the second model and all the features in the third model were derived from third party services. Data retrieved from these services may be missing from time to time, for reasons including poor network connection, temporary unavailability of their servers or absent records in the

databases. A high percentage of missing data is likely to reduce a detection performance. However, our experience during data collection suggests that scenarios which could give rise to missing data in third-party services are relatively rare in normal circumstances. In some other features, for instance, those related to SSL certificates and domain age in the first model, and PTR and uptimes in the second model, missing data is quite common. SSL certificate and uptime features, for example, had 45.4% and 44.6% of missing data respectively. Some features with significant percentages of missing values were actually among the best features for their respective models. In addition, we obtained better results in each model when we included all features with missing values under 50% compared to removing them completely. This suggests that our models, based on the training datasets used, can cope well with up to 50% of missing data in several features. This, however, might not be the case if training datasets with different data distributions are used.

Another challenging issue with the use of features derived from third party services is the substantial increase in prediction time due to the overheads in the retrieval of data. This is a more challenging issue for the second model in which the features contributed most of the total prediction time of 163.6 seconds, making the solution less than ideal for real-time detection.

Another limitation for the first model is that it predicts webpages which are in English language only. We used a list of PDC phrases (listed in Table 4.1) in English as texts to identify PDC webpages. The model, however, can be extended to PDC webpages written in other languages by, for instance, deploying a real time language translation from a specific webpage language to English before checking if the webpage is an PDC one or not. Examples of such translation services that can be added to our model are Google translation API²³ and iTranslate²⁴. Addition of these, however, will increase the prediction time on such webpages.

6.4. Future Work

While completing this thesis, we have identified a number of interesting areas which can be explored further to improve or extend the contributions made in this study. One approach to improving the models, is to increase the sizes of the training datasets. It has already been

²³ <https://cloud.google.com/translate/docs/reference/libraries/v2/python>

²⁴ https://www.itranslate.com/api?gclid=Cj0KCQjwi7yCBhDJARIsAMWFScM0oM-2xNjEDuCvWbTTUP8WKKXH_SIQN2FcA5TCEocb6n4Jb1HLhmEaAuCwEALw_wcB

established that ML algorithms tend to generate better performance with large datasets up to a limit. The performance of DL algorithms, however, increases continuously with the increase in the dataset size. Given the moderate dataset sizes we have used to build our models compared to those used in related studies (shown in Tables 4.12, 5.11 and 5.21), our models are likely to achieve better performance if we increase the data sizes to match or better the largest ones used in other studies.

Although the approaches we have used to determine the thresholds for labelling classes of hostnames in the second and third prediction models have resulted in good prediction performance, we think other approaches to class labelling also have potential and may produce equally good or better prediction performance. The approaches we aim to investigate, evaluate and compare against the one used here include ML clustering and the use of more than one attribute to determine the class labelling threshold. In the former, based on a few relevant features, related hostnames can be grouped by an ML algorithm into clusters in which the majority hostname type in each cluster can be used as a class label for all hostnames belonging to that cluster. In the latter, a combination of the attribute used in this study (the number of changes of IP addresses during the monitoring period) and at least one other attribute such as the average number of IP addresses returned per single lookup or the average number of unique networks of hosts per hostname are likely to be useful.

As pointed out in Chapter 5, FFSNs hosting different types of malicious web service vary in terms of their DNS, host and network characteristics. This is likely to affect the effectiveness of the solutions detecting hostnames hosted in generic FFSNs when they are applied to detect hostnames hosted in specific types of FFSNs. We are aiming to investigate this thoroughly by developing a prediction model based on a generic FFSN and then evaluate and compare its performance against the specific types of FFSNs.

As observed by Salusky and Danford [136], Kadir, et al. [142] and Yadav, et al. [267], it is likely that attackers can combine more than one type of fluxing behaviours including IP flux, domain flux, NS IP flux and NS name flux in the same network. We aim to extend our work by investigating the extent to which attackers use some or all of these approaches concurrently. We also intend to explore potential solutions for detecting hostnames hosted in domain flux and NS name flux.

In recent years, it has been demonstrated that a number of ML models are prone to adversarial ML attacks. These are techniques that attackers can use to inject poisonous data samples in the training data or slightly corrupt some of the benign samples of the training data in order to lower the classification rates of the models [275, 276]. To determine a confidence level of a model in resisting these attacks, it is advised that the developer should assess the model against possible attacks through various experiments. We aim to undertake this evaluation for our proposed models.

References

- [1] APWG. (2021, October, 2021). *Phishing Activity Trends Reports - 2nd Quarter 2021*. Available: https://docs.apwg.org/reports/apwg_trends_report_q1_2021.pdf
- [2] IBM Security. (2019, April, 2020). *Cost of a Data Breach Report 2019*. Available: <https://www.ibm.com/downloads/cas/ZBZLY7KL>
- [3] A. Gendre. (2015, April, 2020). *How Much Does A Spear Phishing Attack Cost?* Available: <https://www.vadecure.com/en/spear-phishing-cost/>
- [4] Retruster. (n.d, April, 2020). *The True Cost of a Phishing Attack*. Available: <https://retruster.com/blog/phishing-attack-true-cost.html>
- [5] Ponemon Institute. (2015). *The Cost of Phishing and Value of Employee Training*. Available: https://info.wombatsecurity.com/hubfs/Ponemon_Institute_Cost_of_Phishing.pdf
- [6] Internet Society (2016). *Global Internet Report 2016*. Available: https://www.internetsociety.org/globalinternetreport/2016/wp-content/uploads/2016/11/ISOC_GIR_2016-v1.pdf
- [7] T. Koulopoulos. (2017, April, 2020). *60 Percent of Companies Fail in 6 Months Because of This (It's Not What You Think)*. Available: <https://www.inc.com/thomas-koulopoulos/the-biggest-risk-to-your-business-cant-be-eliminated-heres-how-you-can-survive-i.html>
- [8] FBI. (2018, April, 2020). *Business E-mail Compromise The 12 Billion Dollar Scam*. Available: <https://www.ic3.gov/media/2018/180712.aspx>
- [9] SecureWorks. (2019, August 2021). *COBALT DICKENS Goes Back to School...Again*. Available: <https://www.secureworks.com/blog/cobalt-dickens-goes-back-to-school-again>
- [10] Verizon. (2018, May, 2020). *2018 Data Breach Investigations Report*. Available: <https://www.phishingbox.com/assets/files/images/Verizon-Data-Breach-Investigations-Report-2018.pdf>
- [11] W. Lee and B. Rotoloni. (2016). *Emerging cyber threats report 2016*. Available: http://www.iisp.gatech.edu/sites/default/files/documents/2016_georgiatech_cyberthreatsreport_onlinescroll.pdf
- [12] A. Gendre. (2019, March, 2020). *4 Ways Hackers Use Phishing to Launch Ransomware Attacks*. Available: <https://www.vadecure.com/en/3-ways-hackers-use-phishing-to-launch-ransomware-attacks/>
- [13] Sophos. (2019, June, 2020). *Don't Take the Bait*. Available: <https://secure2.sophos.com/en-us/medialibrary/Gated-Assets/white-papers/Dont-Take-The-Bait.pdf>
- [14] Allianz. (n.d, March, 2020). *Cyber attacks on critical infrastructure*. Available: <https://www.agcs.allianz.com/news-and-insights/expert-risk-articles/cyber-attacks-on-critical-infrastructure.html>

- [15] T. Ball. (2017, March, 2020). *Top 5 critical infrastructure cyber attacks*. Available: <https://www.cbronline.com/cybersecurity/top-5-infrastructure-hacks/>
- [16] J. Rodríguez. (2019, March, 2020). *Most common attack vector over Critical Infrastructures*. Available: <https://www.cipsec.eu/content/most-common-attack-vector-over-critical-infrastructures>
- [17] R. Pompon. (2019, April, 2020). *Three Ways to Hack the U.S. Election*. Available: <https://www.f5.com/labs/articles/threat-intelligence/three-ways-to-hack-the-u-s-election>
- [18] A. Greenberg. (2017, April, 2020). *Everything We Know About Russia's Election-Hacking Playbook*. Available: <https://www.wired.com/story/russia-election-hacking-playbook/>
- [19] E. Brattberg, Maurer, T., (2018, April, 2020). *Russian Election Interference: Europe's Counter to Fake News and Cyber Attacks*. Available: <https://carnegieendowment.org/2018/05/23/russian-election-interference-europe-s-counter-to-fake-news-and-cyber-attacks-pub-76435>
- [20] CNN. (2020, April, 2020). *2016 Presidential Campaign Hacking Fast Facts*. Available: <https://edition.cnn.com/2016/12/26/us/2016-presidential-campaign-hacking-fast-facts/index.html>
- [21] Verizon. (2020, August 2021). *2020 Data Breach Investigations Report*. Available: <https://dd80b675424c132b90b3-e48385e382d2e5d17821a5e1d8e4c86b.ssl.cf1.rackcdn.com/external/2020-verizon-data-breach-investigations-report.pdf>
- [22] Verizon. (2021, August, 2021). *2021 Data Breach Investigations Report*. Available: <https://enterprise.verizon.com/resources/reports/2021/2021-data-breach-investigations-report.pdf>
- [23] PhishLabs. (2020, April, 2020). *2019 Phishing Trends and Intelligence Report*. Available: <https://info.phishlabs.com/hubfs/2019%20PTI%20Report/2019%20Phishing%20Trends%20and%20Intelligence%20Report.pdf>
- [24] Valimail. (2019, August 2021). *Email Fraud landscape Spring 2019: Billions of fake emails every day — and most industries still vulnerable*. Available: <https://valimail.docsend.com/view/qndhuhn>
- [25] PhishLabs. (2017, March 2017). *2016 Phishing Trends and Intelligence Report: Hacking the Human*. Available: <https://info.phishlabs.com/pti-report-download>
- [26] Akamai. (2019, May, 2020). *Phishing — Baiting the Hook*. Available: <https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/soti-security-phishing-baiting-the-hook-report-2019.pdf>

- [27] Akamai. (2019, April, 2020). *Phishing Is No Longer Just Email: It's Social*. Available: <https://www.akamai.com/us/en/multimedia/documents/white-paper/phishing-is-no-longer-just-email-its-social-white-paper.pdf>
- [28] Microsoft. (2021, October, 2021). *Catching the big fish: Analyzing a large-scale phishing-as-a-service operation*. Available: <https://www.microsoft.com/security/blog/2021/09/21/catching-the-big-fish-analyzing-a-large-scale-phishing-as-a-service-operation/>
- [29] Valimail. (2019, August, 2021). *More than 3 billion fake emails are sent worldwide every day, Valimail report finds*. Available: <https://www.valimail.com/newsroom/more-than-3-billion-fake-emails-are-sent-worldwide-every-day-valimail-report-finds/>
- [30] A. Bhowmick and S. M. Hazarika. (2016, November, 2016). Machine Learning for E-mail Spam Filtering: Review, Techniques and Trends. Available: <https://arxiv.org/pdf/1606.01042.pdf>
- [31] B. M. Bowen, R. Devarajan, and S. Stolfo, "Measuring the human factor of cyber security," in *Proc. IEEE International Conference on Technologies for Homeland Security (HST)*, Waltham, MA, USA, 2011, pp. 230-235.
- [32] E. J. Williams, J. Hinds, and A. N. Joinson, "Exploring susceptibility to phishing in the workplace," *International Journal of Human-Computer Studies*, vol. 120, pp. 1-13, 2018/12/01/ 2018.
- [33] B. Reinheimer, L. Aldag, P. Mayer, M. Mossano, R. Duezguen, B. Lofthouse, *et al.*, "An investigation of phishing awareness and education over time: When and how to best remind users," in *SOUPS @ USENIX Security Symposium*, 2020.
- [34] A. Berggren. (2013, March, 2017). *Nearly all Americans spammed with malware & viruses*. Available: <https://halon.io/blog/nearly-all-americans-spammed-with-malware-viruses/>
- [35] L. Irwin. (2020). *The effects of phishing awareness training wear off over time*. Available: <https://www.itgovernance.co.uk/blog/the-effects-of-phishing-awareness-training-wear-off-over-time>
- [36] D. Piscitello. (2017, September, 2021). *Reputation Block Lists: Protecting Users Everywhere*. Available: <https://www.icann.org/en/blogs/details/reputation-block-lists-protecting-users-everywhere-1-11-2017-en>
- [37] L. Wenyin, G. Liu, B. Qiu, and X. Quan, "Antiphishing through phishing target discovery," *IEEE Internet Computing*, vol. 16, pp. 52-61, 2012.
- [38] P. Barraclough, M. A. Hossain, M. Tahir, G. Sexton, and N. Aslam, "Intelligent phishing detection and protection scheme for online transactions," *Expert Systems with Applications*, vol. 40, pp. 4697-4706, 2013.

- [39] G. Aaron and R. Rasmussen. (2017, December, 2017). *Global Phishing Survey: Trends and Domain Name Use in 2016*. Available: http://docs.apwg.org/reports/APWG_Global_Phishing_Report_2015-2016.pdf
- [40] Webroot. (2016, November 2017). *Webroot Phishing Web Threat Trends: An Update to the 2016 Threat Brief*. Available: https://webroot-cms-cdn.s3.amazonaws.com/7314/8070/2914/Webroot_Threat_Trends_December_2016.pdf
- [41] Webroot. (2018, May, 2020). *2018 Webroot Threat Report*. Available: https://www-cdn.webroot.com/9315/2354/6488/2018-Webroot-Threat-Report_US-ONLINE.pdf
- [42] E. Medvet, E. Kirda, and C. Kruegel, "Visual-similarity-based phishing detection," in *Proc. 4th international conference on Security and privacy in communication networks*, Istanbul, Turkey, 2008 p. 22.
- [43] H. Zhang, G. Liu, T. W. Chow, and W. Liu, "Textual and visual content-based anti-phishing: a Bayesian approach," *IEEE Transactions on Neural Networks*, vol. 22, pp. 1532-1546, 2011.
- [44] K.-T. Chen, J.-Y. Chen, C.-R. Huang, and C.-S. Chen, "Fighting phishing with discriminative keypoint features," *IEEE Internet Computing*, vol. 13, 2009.
- [45] P. Knickerbocker, D. Yu, and J. Li, "Humboldt: A distributed phishing disruption system," in *Proc. eCrime Researchers Summit, 2009. eCRIME'09.*, Tacoma, WA, USA, 2009, pp. 1-12.
- [46] H. Shahriar and M. Zulkernine, "Trustworthiness testing of phishing websites: A behavior model-based approach," *Future Generation Computer Systems*, vol. 28, pp. 1258-1271, 2012.
- [47] H. Zuhair, A. Selamat, and M. Salleh, "New Hybrid Features for Phish Website Prediction," *International Journal of Advances in Soft Computing & Its Applications*, vol. 8, 2016.
- [48] H. K. Shirazi H., Ray I., "Fresh-Phish: A Framework for Auto-Detection of Phishing Websites," presented at the 2017 IEEE International Conference on Information Reuse and Integration (IRI), San Diego, CA, USA 2017.
- [49] A. K. Jain and B. B. Gupta, "Towards detection of phishing websites on client-side using machine learning based approach," *Telecommunication Systems*, vol. 68, pp. 687-700, August 01 2018.
- [50] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from URLs," *Expert Systems with Applications*, vol. 117, pp. 345-357, 2019/03/01/ 2019.
- [51] Y. Li, Z. Yang, X. Chen, H. Yuan, and W. Liu, "A stacking model using URL and HTML features for phishing webpage detection," *Future Generation Computer Systems*, vol. 94, pp. 27-39, 2019.
- [52] APWG. (2016, December 2016). *Phishing Activity Trends Report 4th Quarters 2016*. Available: http://docs.apwg.org/reports/apwg_trends_report_q4_2016.pdf
- [53] FBI. (2020, August, 2021). *Internet Crime Report*. Available: https://www.ic3.gov/Media/PDF/AnnualReport/2020_IC3Report.pdf

- [54] III. (2020, August, 2021). *Facts + Statistics: Identity theft and cybercrime*. Available: <https://www.iii.org/fact-statistic/facts-statistics-identity-theft-and-cybercrime>
- [55] A. Caglayan, M. Toothaker, D. Drapaeau, D. Burke, and G. Eaton, "Behavioral Patterns of Fast Flux Service Networks," in *2010 43rd Hawaii International Conference on System Sciences*, 2010, pp. 1-9.
- [56] G. Ollmann, "The Phishing Guide: Understanding and Preventing Phishing Attacks," 2004.
- [57] A. N. V. Sunil and A. Sardana, "A pagerank based detection technique for phishing web sites," in *2012 IEEE Symposium on Computers & Informatics (ISCI)*, 2012, pp. 58-63.
- [58] V. S. Lakshmi and M. Vijaya, "Efficient prediction of phishing websites using supervised learning algorithms," *Procedia Engineering*, vol. 30, pp. 798-805, 2012.
- [59] Sophos. (2017, June 2020). *Don't take the bait* Available: <https://www.cygnussystems.com/wp-content/uploads/2017/08/dont-take-the-bait.pdf>
- [60] PhishLabs. (2018, January 2017). *2017 Phishing Trends and Intelligence Report: Hacking the Human*. Available: <https://info.phishlabs.com/2017-phishing-trends-and-intelligence-report-pte>
- [61] A. Emigh, "Online Identity Theft: Phishing Technology, Chokepoints and Countermeasures," 2005.
- [62] N. Y. Conteh and P. J. Schmick, "Cybersecurity: risks, vulnerabilities and countermeasures to prevent social engineering attacks," *International Journal of Advanced Computer Research*, vol. 6, p. 31, 2016.
- [63] T. Seals. (2016) Spear Phishing Incident Average Cost is \$1.6M. *Info Security*. Available: <https://www.infosecurity-magazine.com/news/spear-phishing-incident-average/>
- [64] UC Berkeley, "Phishing Example: PayPal - We need your help," 2016.
- [65] ProofPoint. (2016). *Q4 2016 & Year in Review*. Available: https://www.proofpoint.com/sites/default/files/proofpoint_q4_threat_report-final.pdf
- [66] L. La Porta. (2018, April, 2020). *Phishing attacks are moving to messaging and social apps at an alarming rate*. Available: <https://www.wandera.com/mobile-phishing-attacks/>
- [67] D. Jevans, "Phishing Goes Mobile – The Rise of SMS Phishing," in *Threat Insight*, ed: ProofPoint, 2017.
- [68] E. Volkman, "The Rise in Mobile Phishing Attacks," in *The PhishLabs Blog*, ed: PhishLabs, 2019.
- [69] Lockout. (2018, April, 2020). *Mobile phishing 2018: Myths and facts facing every modern enterprise today*. Available: <https://info.lockout.com/rs/051-ESQ-475/images/Lookout-Phishing-wp-us.pdf>

- [70] D. Palmer. (2017, June 2017). *TrickBot banking Trojan steps up attacks against UK targets*. Available: <https://www.zdnet.com/article/trickbot-banking-trojan-steps-up-attacks-against-uk-targets/>
- [71] Symantec, "Symantec Internet Security Threat Report 2013," 2013.
- [72] PandaLabs. (2013, November 2016). *PandaLabs Annual Report 2012*. Available: <http://www.pandasecurity.com/mediacenter/src/uploads/2014/07/PandaLabs-Annual-Report-2012.pdf>
- [73] S. Stamm, Z. Ramzan, and M. Jakobsson, "Drive-By Pharming," in *Proceedings Information and Communications Security: 9th International Conference, ICICS 2007, Zhengzhou, China, December 12-15, 2007*. , S. Qing, H. Imai, and G. Wang, Eds., ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 495-506.
- [74] E. Maelstrom, "Phishing with Wildcard DNS Attacks and Pharming," in *The PhishLabs Blog*, ed, 2017.
- [75] E. Kovacs. (2019, February, 2021). *DHS Warns Federal Agencies of DNS Hijacking Attacks*. Available: <https://www.securityweek.com/dhs-warns-federal-agencies-dns-hijacking-attacks>
- [76] J. Zhang, C. Yang, Z. Xu, and G. Gu, "Poisonamplifier: A guided approach of discovering compromised websites through reversing search poisoning attacks," in *International Workshop on Recent Advances in Intrusion Detection*, 2012, pp. 230-253.
- [77] P. O'Connor. (2020). *An Introduction to Black Hat SEO*. Available: <https://blog.hubspot.com/marketing/black-hat-seo>
- [78] J. Wang, "Ubiquitous SEO Poisoning URLs," 2018.
- [79] M. Bartley. (2019, May, 2020). *What Is SEO Spam and How to Remove It?* Available: <https://www.malcare.com/blog/seo-spam/>
- [80] D. Patel, Niv, N. (2016, May, 2020). *Vector Attacks Prey on Thousands of Legitimate Websites*. Available: <https://www.imperva.com/blog/black-hat-seo-attacks/>
- [81] D. Airehrour, N. Vasudevan Nair, and S. Madanian, "Social engineering attacks and countermeasures in the new zealand banking system: Advancing a user-reflective mitigation model," *Information*, vol. 9, p. 110, 2018.
- [82] X. Luo, R. Brody, A. Seazzu, and S. Burd, "Social Engineering: The Neglected Human Factor for Information Security Management," *Inf. Resour. Manage. J.*, vol. 24, pp. 1–8, 2011.
- [83] Z. Alkhalil, C. Hewage, L. Nawaf, and I. Khan, "Phishing Attacks: A Recent Comprehensive Study and a New Anatomy," *Frontiers in Computer Science*, vol. 3, 2021-March-09 2021.
- [84] B. Wallace. (2019, April, 2020). *The Cost of Email Phishing*. Available: <https://www.dumblittleman.com/phishing-attacks/>
- [85] Avanan. (2019, April, 2020). *The History & Future of Phishing*. Available: <https://www.avanan.com/resources/infographics/the-history-and-future-of-phishing>

- [86] Verizon. (2019, April, 2020). *2019 Data Breach Investigations Report*. Available: <https://enterprise.verizon.com/resources/reports/2019-data-breach-investigations-report-emea.pdf>
- [87] E. Volkman. (2019, April, 2020). *Why Social Media is Increasingly Abused for Phishing Attacks*. Available: <https://info.phishlabs.com/blog/how-social-media-is-abused-for-phishing-attacks>
- [88] ProofPoint. (2017, May, 2020). *Social Media Protection Brand Fraud Report* Available: <https://go.proofpoint.com/rs/309-RHV-619/images/Social%20Media%20Protection%20Brand%20Fraud%20Report%20v2.pdf>
- [89] A. Castle. (2018, May, 2020). *Just Keep Swimming: How to Avoid Phishing on Social Media*. Available: <https://www.webroot.com/blog/2018/01/22/how-to-avoid-phishing-social-media/>
- [90] ProofPoint. (2020, May, 2020). *2020 State of Phish*. Available: <https://www.proofpoint.com/sites/default/files/gtd-pfpt-us-tr-state-of-the-phish-2020.pdf>
- [91] A. Kumar, Rosso, K.,. (2020, May, 2020). *Lookout Phishing AI provides an inside look into a phishing campaign targeting mobile banking users*. Available: <https://blog.lookout.com/lookout-phishing-ai-reveals-mobile-banking-phishing-campaign>
- [92] APWG, "Phishing Activity Trends Report, 4th Quarter 2016," 2017.
- [93] APWG. (2020, April, 2020). *Phishing Activity Trends Report: 4th Quarter 2019*. Available: https://docs.apwg.org/reports/apwg_trends_report_q4_2019.pdf
- [94] Kavya. (2020, April, 2020). *Types of SSL Certificates for a Secure Business Website*. Available: <https://serverguy.com/ssl/types-of-ssl-certificates/>
- [95] Robertckl, "Types of SSL certificates – choose the right one," in *Symantec Official Blog*, ed: Symantec, 2014.
- [96] Global Sign. (n.d, June 2018). *What Are The Different Types of SSL Certificates?* Available: <https://www.globalsign.com/en/ssl-information-center/types-of-ssl-certificate/>
- [97] Acmetek. (n.d, June 2018). *What is the difference between Domain Validated (DV), Organization Validated and Extended Validation (EV) SSL?* Available: <https://www.ssldesk.com/what-is-the-difference-between-domain-validated-dv-organization-validated-and-extended-validation-ev-ssl/>
- [98] D. Warburton, Pompon, R.,. (2019, April, 2020). *2019 Phishing and Fraud Report*. Available: <https://www.f5.com/labs/articles/threat-intelligence/2019-phishing-and-fraud-report>
- [99] PhishLabs. (2019, April, 2020). *2018 Phishing Trends and Intelligence Report*. Available: https://info.phishlabs.com/hubfs/2018%20PTI%20Report/PhishLabs%20Trend%20Report_2018-digital.pdf

- [100] Webroot. (2019, May, 2020). *The 2019 Webroot Threat Report*. Available: https://www-cdn.webroot.com/9315/5113/6179/2019_Webroot_Threat_Report_US_Online.pdf
- [101] R. Kurtus. (2014, June 2018). *Types of HTML Hyperlinks*. Available: https://www.school-for-champions.com/web/html_hyperlinks.htm
- [102] S. Ghobril, "What is href="#" and why is it used?," in *Stack Overflow*, ed, 2015.
- [103] Omg, "What does "javascript:void(0)" mean?," in *Stack Overflow*, ed: Stack Overflow, 2009.
- [104] J. Valk. (2016, November 2017). *re=canonical: The ultimate guide*. Available: <https://yoast.com/rel-canonical/#when-to-canonicalize>
- [105] C. Meier. (2014, December 2017). *A Beginners Introduction To The Canonical Tag*. Available: <https://unamo.com/blog/general/beginners-introduction-canonical-tag>
- [106] N. Eubanks. (2012, December 2017). *A Simple Guide to Using rel="alternate" hreflang="x"*. Available: <https://searchenginewatch.com/sew/how-to/2232347/a-simple-guide-to-using-rel-alternate-hreflang-x>
- [107] Microformats. (2016, December 2017). *rel-alternate*. Available: <http://microformats.org/wiki/rel-alternate>
- [108] Google. (2017, December 2017). *Separate URLs* Available: <https://developers.google.com/search/mobile-sites/mobile-seo/separate-urls>
- [109] W3Schools. (n.d, November 2017). *HTML Input Types*. Available: https://www.w3schools.com/html/html_form_input_types.asp
- [110] Universal Class. (n.d, November 2017). *User Input and Output in JavaScript*. Available: <https://www.universalclass.com/articles/computers/javascript/user-input-and-output-in-javascript.htm>
- [111] W3Schools. (n.d, November 2017). *JavaScript Popup Boxes*. Available: https://www.w3schools.com/js/js_popup.asp
- [112] N. Agarwal. (n.d, November 2017). *jQuery Get Value Of Input, Textarea and Radio Button*. Available: <https://www.formget.com/jquery-get-value-of-input/>
- [113] Universal Class. (n.d, November 2017). *How to Script Forms with JavaScript*. Available: <https://www.universalclass.com/articles/computers/javascript/how-to-script-forms-with-javascript.htm>
- [114] W3Schools. (n.d, November 2017). *HTML <label> Tag*. Available: https://www.w3schools.com/tags/tag_label.asp
- [115] C. Broadley. (n.d, November 2017). *<input value="">*. Available: <https://html.com/attributes/input-value/>
- [116] R. Zhao, S. John, S. Karas, C. Bussell, J. Roberts, D. Six, *et al.*, "The highly insidious extreme phishing attacks," in *Proc. 25th International Conference on Computer Communication and Networks (ICCCN)*, Waikoloa, HI, USA, 2016, pp. 1-10.

- [117] G. Xiang, J. Hong, C. P. Rose, and L. Cranor, "Cantina+: A feature-rich machine learning framework for detecting phishing web sites," *ACM Transactions on Information and System Security (TISSEC)*, vol. 14, p. 21, 2011.
- [118] PCHelp. (2002, December 2017). *How to Obscure Any URL*. Available: <http://www.pchelp.org/obscure.htm>
- [119] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: learning to detect malicious web sites from suspicious URLs," in *Proc. 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, Paris, France, 2009, pp. 1245-1254.
- [120] N. Poojary. (2019, May, 2020). *How DNS Works – the Domain Name System (Part One)*. Available: <https://cloudacademy.com/blog/how-dns-works/>
- [121] Microsoft, "How DNS Works," in *TechNet*, ed, 2003.
- [122] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel, "Exposure: A Passive DNS Analysis Service to Detect and Report Malicious Domains," *ACM Trans. Inf. Syst. Secur.*, vol. 16, pp. 1-28, 2014.
- [123] D. Barr. (1996, August, 2019). *Common DNS Operational and Configuration Errors*. Available: <https://www.ietf.org/rfc/rfc1912.txt>
- [124] Y. Chang, K. Yoon, and D. Park, "A Study on the IP Spoofing Attack through Proxy Server and Defense Thereof," in *2013 International Conference on Information Science and Applications (ICISA)*, 2013, pp. 1-3.
- [125] R. Prego. (2016, August, 2019). *5 Reasons Your Company Should Use Proxy Servers*. Available: <https://www.cmswire.com/information-management/5-reasons-your-company-should-use-proxy-servers/>
- [126] V. Stocker, G. Smaragdakis, W. Lehr, and S. Bauer, "The growing complexity of content delivery networks: Challenges and implications for the Internet ecosystem," *Telecommunications Policy*, vol. 41, pp. 1003-1016, 2017/11/01/ 2017.
- [127] H.-T. Lin, Y.-Y. Lin, and J.-W. Chiang, "Genetic-based real-time fast-flux service networks detection," *Computer Networks*, vol. 57, pp. 501-513, 2013/02/04/ 2013.
- [128] E. Passerini, R. Paleari, L. Martignoni, and D. Bruschi, "FluXOR: Detecting and Monitoring Fast-Flux Service Networks," in *Proc. 5th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, Paris, France, 2008, pp. 186-206.
- [129] A. Almomani, "Fast-flux hunter: a system for filtering online fast-flux botnet," *Neural Computing and Applications*, vol. 29, pp. 483-493, 2018.
- [130] R. Perdisci, I. Corona, D. Dagon, and W. Lee, "Detecting malicious flux service networks through passive analysis of recursive dns traces," presented at the Proc. 2009. ACSAC'09. Annual Computer Security Applications Conference Honolulu, HI, USA, 2009.

- [131] T. Holz, C. Gorecki, K. Rieck, and F. Freiling, "Measuring and Detecting Fast-Flux Service Networks," presented at the Proc. 16th Annual Network & Distributed System Security Symposium (NDSS), San Diego, CA, 2008.
- [132] G. Hogben, Gerhards-Padilla, E., Leder, F., (2011, February, 2017). *Botnets: Detection, Measurement, Disinfection & Defence*. Available: <https://www.enisa.europa.eu/publications/botnets-measurement-detection-disinfection-and-defence>
- [133] F. Hsu, C. Wang, C. Hsu, C. Tso, L. Chen, and S. Lin, "Detect Fast-Flux Domains Through Response Time Differences," *IEEE Journal on Selected Areas in Communications*, vol. 32, pp. 1947-1956, 2014.
- [134] S. Campbell, S. Chan, and J. R. Lee, "Detection of fast flux service networks," presented at the Proceedings of the Ninth Australasian Information Security Conference - Volume 116, Perth, Australia, 2011.
- [135] W. Salusky and R. Danford. (2008, April 2017). Know your Enemy: fast-flux service networks. *The Honeynet Project*. Available: https://www.researchgate.net/publication/328781281_Know_Your_Enemy_Fast-Flux_Service_Networks
- [136] C.-H. Hsu, C.-Y. Huang, and K.-T. Chen, "Fast-Flux Bot Detection in Real Time." vol. 6307, ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 464-483.
- [137] ThreatX Labs. (2017, August, 2019). *Malicious Bot Detection through A Complex Proxy Network*. Available: <https://blog.threatxlabs.com/malicious-bot-detection-through-complex-proxy-network>
- [138] J. Nazario and T. Holz, "As the net churns: Fast-flux botnet observations," presented at the 3rd International Conference on Malicious and Unwanted Software (MALWARE), Fairfax, VI, USA 2008.
- [139] Y. M. P. Pa, K. Yoshioka, and T. Matsumoto, "Detecting Malicious Domains and Authoritative Name Servers Based on Their Distinct Mappings to IP Addresses," *Journal of Information Processing*, vol. 23, pp. 623-632, 2015.
- [140] L. B. Metcalf and J. M. Spring. (2013). *Passive Detection of Misbehaving Name Servers*. Available: http://resources.sei.cmu.edu/asset_files/TechnicalReport/2013_005_001_65284.pdf
- [141] A. F. A. Kadir, R. A. R. Othman, and N. A. Aziz, "Behavioral Analysis and Visualization of Fast-Flux DNS," in *2012 European Intelligence and Security Informatics Conference*, 2012, pp. 250-253.

- [142] M. Konte, N. Feamster, and J. Jung, "Dynamics of online scam hosting infrastructure," presented at the Proc. International conference on passive and active network measurement, 2009.
- [143] E. Burns. (2021, September, 2021). *Machine Learning*. Available: <https://searchenterpriseai.techtarget.com/definition/machine-learning-ML>
- [144] J.-h. Li, "Cyber security meets artificial intelligence: a survey," *Frontiers of Information Technology & Electronic Engineering*, vol. 19, pp. 1462-1474, 2018/12/01 2018.
- [145] R. Mohammed, R. Vinayakumar, and K. Soman, "A short review on Applications of Deep learning for Cyber security," *arXiv e-prints*, p. arXiv: 1812.06292, 2018.
- [146] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, *et al.*, "Machine Learning and Deep Learning Methods for Cybersecurity," *IEEE Access*, vol. 6, pp. 35365-35381, 2018.
- [147] G. Apruzzese, Colajanni, M., Ferretti, L., Guido, A., Marchetti, M., "On the effectiveness of machine and deep learning for cyber security," in *2018 10th International Conference on Cyber Conflict (CyCon)*, Estonia, 2018, pp. 371-390.
- [148] M. Al-Garadi, M. Amr, A. Al-Ali, X. Du, and M. Guizani, "A survey of machine and deep learning methods for internet of things (IoT) security," *arXiv preprint arXiv:1807.11023*, 2018.
- [149] Internet Society. (2017, December 2017). *Artificial Intelligence and Machine Learning: Policy Paper*. Available: https://www.internetsociety.org/wp-content/uploads/2017/08/ISOC-AI-Policy-Paper_2017-04-27_0.pdf
- [150] F. Fruth. (2018, April 2018). *How different Kinds of Artificial Intelligence Will Shape Industries – the Case of Smart Production in Life Science*. Available: <https://www.capgemini.com/consulting-de/2018/01/ai-concept-frame/#>
- [151] FSB. (2017, Artificial intelligence and machine learning in financial services - Market developments and financial stability implications. Available: <http://www.fsb.org/wp-content/uploads/P011117.pdf>
- [152] K. J. Cios, R. W. Swiniarski, W. Pedrycz, and L. A. Kurgan, "Supervised Learning: Decision Trees, Rule Algorithms, and Their Hybrids," in *Data Mining: A Knowledge Discovery Approach*, ed Boston, MA: Springer US, 2007, pp. 381-417.
- [153] M. Somvanshi and P. Chavan, "A review of machine learning techniques using decision tree and support vector machine," presented at the 2016 International Conference on Computing Communication Control and automation (ICCUBEA) Pune, India, 2016.
- [154] M. Pineda. (2017, December 2017). *Benefits and limitations of machine learning*. Available: <https://www.profolus.com/topics/benefits-limitations-of-machine-learning/>
- [155] A. L'Heureux, K. Grolinger, and H. Elyamany, "Machine Learning With Big Data: Challenges and Approaches," *IEEE Access*, vol. 5, December 2017 2017.

- [156] K. Guruswamy (2015, December 2017). *Data Science: Machine Learning Vs. Rules Based Systems*. Available: <https://www.forbes.com/sites/teradata/2015/12/15/data-science-machine-learning-vs-rules-based-systems/#37f1265b2119>
- [157] J. Brownlee. (2016, April 2018). *Supervised and Unsupervised Machine Learning Algorithms*. Available: <https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/>
- [158] R. Brooks and k. Dahlke. (2017, April 2018). *Understanding the 3 Categories of Machine Learning – AI vs. Machine Learning vs. Data Mining 101 (part 2)*. Available: <http://guavus.com/ai-vs-machine-learning-vs-data-mining-whats-big-difference-part-2/>
- [159] J. Brownlee. (2017). *Difference Between Classification and Regression in Machine Learning*. Available: <https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/#:~:text=Fundamentally%2C%20classification%20is%20about%20predicting,is%20about%20predicting%20a%20quantity.&text=That%20classification%20is%20the%20problem,quantity%20output%20for%20an%20example.>
- [160] J. Brownlee. (2020, May, 2021). *4 Types of Classification Tasks in Machine Learning*. Available: <https://machinelearningmastery.com/types-of-classification-in-machine-learning/>
- [161] A. Band. (2020, May, 2020). *Multi-class Classification — One-vs-All & One-vs-One*. Available: <https://towardsdatascience.com/multi-class-classification-one-vs-all-one-vs-one-94daed32a87b>
- [162] J. Brownlee. (2020, May, 2021). *One-vs-Rest and One-vs-One for Multi-Class Classification*. Available: <https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/>
- [163] L. Merschmann and A. Freitas, "An extended local hierarchical classifier for prediction of protein and gene functions," presented at the International Conference on Data Warehousing and Knowledge Discovery, 2013.
- [164] C. N. Silla and A. A. Freitas, "A survey of hierarchical classification across different application domains," *Data Mining and Knowledge Discovery*, vol. 22, pp. 31-72, 2011.
- [165] M. Ceci and D. Malerba, "Classifying web documents in a hierarchy of categories: a comprehensive study," *Journal of Intelligent Information Systems*, vol. 28, pp. 37-78, 2007.
- [166] N. Weiss. (2020). *Hierarchical Classification with Local Classifiers: Down the Rabbit Hole*. Available: <https://towardsdatascience.com/hierarchical-classification-with-local-classifiers-down-the-rabbit-hole-21cdf3bd2382>
- [167] K. Kowsari, D. E. Brown, M. Heidarysafa, K. J. Meimandi, M. S. Gerber, and L. E. Barnes, "Hdltext: Hierarchical deep learning for text classification," presented at the 2017 16th IEEE international conference on machine learning and applications (ICMLA), 2017.

- [168] A. D. Secker, M. N. Davies, A. A. Freitas, J. Timmis, M. Mendao, and D. R. Flower, "An experimental comparison of classification algorithms for hierarchical prediction of protein function," *Expert Update (Magazine of the British Computer Society's Specialist Group on AI)*, vol. 9, pp. 17-22, 2007.
- [169] C. N. Silla and A. A. Freitas, "Novel top-down approaches for hierarchical classification and their application to automatic music genre classification," presented at the 2009 IEEE International Conference on Systems, Man and Cybernetics, 2009.
- [170] Kaushal113. (2020, May, 2021). *Hypothesis Generation for Data Science Projects – A Critical Problem Solving Step*. Available: <https://www.analyticsvidhya.com/blog/2020/09/hypothesis-generation-data-science-projects/>
- [171] F. Chollet, *Deep Learning with Python*, First ed. New York, USA: Manning Publications, 2017.
- [172] Y. Gavrilova, Bolgurtseva, O.,. (2020, May, 2021). *What Is Data Preprocessing in ML?* Available: <https://serokell.io/blog/data-preprocessing>
- [173] K. Goyal. (2020, May, 2021). *Data Preprocessing in Machine Learning: 7 Easy Steps To Follow*. Available: <https://www.upgrad.com/blog/data-preprocessing-in-machine-learning/>
- [174] IBM. (2020, May, 2021). *Exploratory Data Analysis*. Available: <https://www.ibm.com/cloud/learn/exploratory-data-analysis>
- [175] H. Heike, H. Wickham, and K. Kafadar, "Letter-Value Plots: Boxplots for Large Data," *Journal of Computational and Graphical Statistics*, vol. 26, 03/13 2017.
- [176] J. Brownlee. (2020, May, 2021). *How to Perform Data Cleaning for Machine Learning with Python*. Available: <https://machinelearningmastery.com/basic-data-cleaning-for-machine-learning/>
- [177] J. Brownlee. (2017, September, 2018). *How to Handle Missing Data with Python*. Available: <https://machinelearningmastery.com/handle-missing-data-python/>
- [178] G. Dhasade. (2020, May, 2021). *Ways To Handle Categorical Column Missing Data & Its Implementations*. Available: <https://medium.com/analytics-vidhya/ways-to-handle-categorical-column-missing-data-its-implementations-15dc4a56893>
- [179] M. Tripathi. (2020, May, 2021). *Knowing all about Outliers in Machine Learning*. Available: <https://datascience.foundation/sciencewhitepaper/knowing-all-about-outliers-in-machine-learning>
- [180] J. Brownlee. (2020, May, 2021). *Introduction to Dimensionality Reduction for Machine Learning*. Available: <https://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/>

- [181] R. a. W. Sillipo, M. (2019, May, 2021). *3 New Techniques for Data-Dimensionality Reduction in Machine Learning*. Available: <https://thenewstack.io/3-new-techniques-for-data-dimensionality-reduction-in-machine-learning/>
- [182] A. Müller and S. Guido, *Introduction to Machine Learning with Python*, 1 ed. U.S: O'Reilly Media, 2017.
- [183] M. Pathak. (2018). *Handling Categorical Data in Python*. Available: <https://www.datacamp.com/community/tutorials/categorical-data>
- [184] S. Ronaghan. (2019, May, 2021). *Data Preparation for Machine Learning: Cleansing, Transformation & Feature Engineering*. Available: <https://towardsdatascience.com/data-preparation-for-machine-learning-cleansing-transformation-feature-engineering-d2334079b06d>
- [185] J. Brownlee. (2019, May, 2021). *How to use Data Scaling Improve Deep Learning Model Stability and Performance*. Available: <https://machinelearningmastery.com/how-to-improve-neural-network-stability-and-modeling-performance-with-data-scaling/>
- [186] J. Brownlee, *Machine Learning Mastery With Python*, 1.4 ed., 2016.
- [187] J. Brownlee. (2019, May, 2021). *A Gentle Introduction to Imbalanced Classification*. Available: <https://machinelearningmastery.com/what-is-imbalanced-classification/>
- [188] J. Brownlee. (2015, May, 2021). *8 Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset*. Available: <https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/>
- [189] J. Brownlee. (2014, August, 2018). *Classification Accuracy is Not Enough: More Performance Measures You Can Use*. Available: <https://machinelearningmastery.com/classification-accuracy-is-not-enough-more-performance-measures-you-can-use/>
- [190] S. Narkhede. (2018, October, 2018). *Understanding AUC - ROC Curve*. Available: <https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5>
- [191] J. Brownlee. (2014, August, 2018). *Assessing and Comparing Classifier Performance with ROC Curves*. Available: <https://machinelearningmastery.com/assessing-comparing-classifier-performance-roc-curves-2/>
- [192] J. Brownlee. (2020, May, 2021). *Hyperparameter Optimization With Random Search and Grid Search*. Available: <https://machinelearningmastery.com/hyperparameter-optimization-with-random-search-and-grid-search/>
- [193] P. Worcester. (2019, May, 2021). *Comparison of Grid Search and Randomized Search Using Scikit Learn*. Available: <https://blog.usejournal.com/a-comparison-of-grid-search-and-randomized-search-using-scikit-learn-29823179bc85>

- [194] C. Nicholson. (n.d, May, 2021). *Machine Learning Workflows*. Available: <https://wiki.pathmind.com/machine-learning-workflow>
- [195] D. Berman, A. Buczak, J. Chavis, and C. Corbett "A Survey of Deep Learning Methods for Cyber Security," *Information*, vol. 10, 2019.
- [196] J. Brownlee. (2016). *Logistic Regression for Machine Learning*. Available: <https://machinelearningmastery.com/logistic-regression-for-machine-learning/>
- [197] A. Thanda. (2020, May, 2021). *What is Logistic Regression? A Beginner's Guide*. Available: <https://careerfoundry.com/en/blog/data-analytics/what-is-logistic-regression/>
- [198] J. VanderPlas, *Python Data Science Handbook*, 1st ed. California, USA: O'Reilly Media, 2017.
- [199] S. Ray. (2017, May, 2021). *Understanding Support Vector Machine(SVM) algorithm from examples (along with code)*. Available: <https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>
- [200] N. Chauhan. (2020, May, 2021). *Decision Tree Algorithm, Explained*. Available: <https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html>
- [201] JavaTpoint. (n.d, May, 2021). *Decision Tree Classification Algorithm*. Available: <https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm>
- [202] P. Yadav. (2018, May, 2021). *Decision Tree in Machine Learning*. Available: <https://towardsdatascience.com/decision-tree-in-machine-learning-e380942a4c96>
- [203] N. Donges. (2021). *A complete guide to Random Forest Algorithm*. Available: <https://builtin.com/data-science/random-forest-algorithm>
- [204] T. Yiu. (2019, February 2020). *Understanding Random Forest*. Available: <https://towardsdatascience.com/understanding-random-forest-58381e0602d2>
- [205] C. Nicholson. (n.d, May, 2021). *A Beginner's Guide to Neural Networks and Deep Learning*. Available: <https://wiki.pathmind.com/neural-network>
- [206] B. Dickson. (2019, May, 2021). *What are artificial neural networks (ANN)?* Available: <https://bdtechtalks.com/2019/08/05/what-is-artificial-neural-network-ann/>
- [207] J. Moolayil, *Learn Keras for Deep Neural Networks*. Vancouver, Canada: Apress, 2019.
- [208] M. Nguyen. (2018, September, 2019). *Illustrated Guide to LSTM's and GRU's: A step by step explanation*. Available: <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>
- [209] M. Phi. (2018). *Illustrated Guide to LSTM's and GRU's: A step by step explanation*. Available: <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>

- [210] P. Karkare. (2019, May, 2021). *A 7 Minute Introduction to LSTM*. Available: <https://medium.com/x8-the-ai-community/a-7-minute-introduction-to-lstm-5e1480e6f52a>
- [211] S. Yan. (2016, May, 2021). *Understanding LSTM and its diagrams*. Available: <https://blog.mlreview.com/understanding-lstm-and-its-diagrams-37e2f46f1714>
- [212] S. Verma. (2019, March, 2020). *Understanding 1D and 3D Convolution Neural Network / Keras*. Available: <https://towardsdatascience.com/understanding-1d-and-3d-convolution-neural-network-keras-9d8f76e29610>
- [213] A. Dertat. (2017, September, 2019). *Applied Deep Learning - Part 4: Convolutional Neural Networks*. Available: <https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2>
- [214] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1D convolutional neural networks and applications: A survey," *Mechanical systems and signal processing*, vol. 151, p. 107398, 2021.
- [215] Google. (n.d, March 2017). *What is Safe Browsing?* . Available: <https://developers.google.com/safe-browsing/>
- [216] Google. (n.d, March 2017). *Google Safe Browsing*. Available: <https://safebrowsing.google.com/>
- [217] S. Somogyi, Miller, A.,. (2017, May, 2020). *Safe Browsing: Protecting more than 3 billion devices worldwide, automatically*. Available: <https://www.blog.google/technology/safety-security/safe-browsing-protecting-more-3-billion-devices-worldwide-automatically/>
- [218] Geotrust. (n.d, January 2017). *WEB SITE VERIFICATION SEARCH SITE AND TOOLBAR FIGHTS ONLINE FRAUD AND PHISHING SCAMS*. Available: https://www.trustico.co.uk/material/DS_TrustWatch.pdf
- [219] EC Council, *Ethical Hacking and Countermeasures: Threats and Defence Mechanisms*, 2nd ed. USA: EC-Council Press, 2017.
- [220] Kent. (2013, March 2017). *Bitdefender TrafficLight protects you from malware, phishing websites, and privacy trackers [Chrome, Firefox, Safari]* Available: <https://dottech.org/132882/review-bitdefender-trafficlight/>
- [221] Trend Micro. (n.d). *Trend Micro: Phishing*. Available: <http://www.antivirus.co.uk/Internet-Security-Info/Phishing/>
- [222] ESET. (2017, August 2017). *ESET Anti-Phishing*. Available: <https://www.eset.com/us/anti-phishing/>
- [223] Kaspersky. (2015, August 2017). *Secure Web Surfing With Kaspersky Lab Advanced Anti-Phishing Technology* Available: http://media.kaspersky.com/pdf/Kaspersky_Lab_Whitepaper_Anti_phishing.pdf

- [224] X. Dong, J. A. Clark, and J. L. Jacob, "Defending the weakest link: phishing websites detection by analysing user behaviours," *Telecommunication Systems*, vol. 45, pp. 215-226, 2010.
- [225] M. Hara, A. Yamada, and Y. Miyake, "Visual similarity-based phishing detection without victim site information," in *Proc. IEEE Symposium on Computational Intelligence in Cyber Security*, Nashville, TN, USA, 2009, pp. 30-36.
- [226] M. Aburrous, M. A. Hossain, K. Dahal, and F. Thabtah, "Intelligent phishing detection system for e-banking using fuzzy data mining," *Expert systems with applications*, vol. 37, pp. 7913-7921, 2009.
- [227] R. M. Mohammad, F. Thabtah, and L. McCluskey, "Predicting phishing websites based on self-structuring neural network," *Neural Computing and Applications*, vol. 25, pp. 443-458, August 01 2014.
- [228] W. Zhang, Q. Jiang, L. Chen, and C. Li, "Two-stage ELM for phishing Web pages detection using hybrid features," *World Wide Web*, vol. 20, pp. 797-813, 2017.
- [229] R. Gowtham and I. Krishnamurthi, "A comprehensive and efficacious architecture for detecting phishing webpages," *Computers & Security*, vol. 40, pp. 23-37, 2014.
- [230] AV-Comparatives. (2016). *Anti-phishing Test*. Available: https://www.av-comparatives.org/wp-content/uploads/2016/07/avc_phi_2016_en.pdf
- [231] C. Whittaker, B. Ryner, and M. Nazif, "Large-Scale Automatic Classification of Phishing Pages," in *Proc. Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, USA, 2010, p. 2010.
- [232] Y. Pan and X. Ding, "Anomaly based web phishing page detection," in *Proc. 22nd Annual Computer Security Applications Conference (ACSAC'06)*, Miami Beach, FL, USA, 2006, pp. 381-392.
- [233] M. He, S.-J. Horng, P. Fan, M. K. Khan, R.-S. Run, J.-L. Lai, *et al.*, "An efficient phishing webpage detector," *Expert Systems with Applications*, vol. 38, pp. 12018-12027, 2011.
- [234] L. Maciak. (2006, December 2017). *URL Obfuscation*. Available: <http://www.terminally-incoherent.com/blog/2006/09/03/url-obfuscation/>
- [235] W3Schools. (n.d, December 2017). *HTML URL Encoding Reference*. Available: https://www.w3schools.com/tags/ref_urlencode.asp
- [236] Microsoft. (n.d, December 2017). *Internet Explorer does not support user names and passwords in Web site addresses (HTTP or HTTPS URLs)*. Available: <https://support.microsoft.com/en-us/help/834489/internet-explorer-does-not-support-user-names-and-passwords-in-web-sit>
- [237] A. Hagen. (2003, December 2017). *Warn when HTTP URL auth information isn't necessary or when it's provided*. Available: https://bugzilla.mozilla.org/show_bug.cgi?id=232567

- [238] Google. (n.d, July 2018). *Best practices to avoid sending Personally Identifiable Information (PII)*. Available: <https://support.google.com/adsense/answer/6156630?hl=en>
- [239] P. Mitton. (2016, July 2018). *Never Put Secrets in URLs and Query Parameters*. Available: <https://www.fullcontact.com/blog/never-put-secrets-urls-query-parameters/>
- [240] Boldewyn, "Email addresses inside URL," in *Stack Overflow*, ed, 2013.
- [241] S. Garera, N. Provos, M. Chew, and A. D. Rubin, "A framework for detection and measurement of phishing attacks," presented at the Proceedings of the 2007 ACM workshop on Recurring malware, Alexandria, Virginia, USA, 2007.
- [242] Global Sign. (n.d, July 2018). *How to tell DV and OV SSL Certificates apart*. Available: <https://www.globalsign.com/en/ssl-information-center/telling-dv-and-ov-certificates-apart/>
- [243] Symantec, "May 2011 Intelligence Report," 2011.
- [244] W. Ashford. (2017, May, 2017). *Flawed GoDaddy security certificates show need for control*. Available: <https://www.computerweekly.com/news/450410837/Flawed-GoDaddy-security-certificates-show-need-for-control>
- [245] E. Ysasi. (n.d, November 2017). *How Long Does It Take Google to Index a New Site?*. Available: <https://www.theleverageway.com/blog/how-long-does-it-take-google-to-index-a-new-site/>
- [246] Kunal, "What should be the allowed percentage of Missing Values?," in *Analytics Vidhya*, ed, 2015.
- [247] P. Madley-Dowd, R. Hughes, K. Tilling, and J. Heron, "The proportion of missing data should not be used to guide decisions on multiple imputation," *Journal of Clinical Epidemiology*, vol. 110, pp. 63-73, 2019/06/01/ 2019.
- [248] MachMetrics, "Average Page Load Time in 2021," in *MachMetrics Speed*, ed, 2021.
- [249] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "BotHunter: detecting malware infection through IDS-driven dialog correlation," presented at the Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium, Boston, MA, 2007.
- [250] S. Khattak, Z. Ahmed, A. A. Syed, and S. A. Khayam, "BotFlex: A community-driven tool for botnet detection," *Journal of Network and Computer Applications*, vol. 58, pp. 144-154, 2015.
- [251] S. Kumar and B. Xu, "A Machine Learning Based Approach to Detect Malicious Fast Flux Networks," presented at the 2018 IEEE Symposium Series on Computational Intelligence (SSCI), 2018.
- [252] X. Chen, G. Li, Y. Zhang, X. Wu, and C. Tian, "A Deep Learning Based Fast-Flux and CDN Domain Names Recognition Method," presented at the Proceedings of the 2019 2nd International Conference on Information Science and Systems, Tokyo, Japan, 2019.

- [253] S.-Y. Huang, C.-H. Mao, and H.-M. Lee, "Fast-flux service network detection based on spatial snapshot mechanism for delay-free detection," presented at the Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, Beijing, China, 2010.
- [254] H. Wang, C. Mao, K. Wu, and H. Lee, "Real-Time Fast-Flux Identification via Localized Spatial Geolocation Detection," presented at the 2012 IEEE 36th Annual Computer Software and Applications Conference, 2012.
- [255] M. Stevanovic, J. M. Pedersen, A. D'alconzo, and S. Ruehrup, "A method for identifying compromised clients based on DNS traffic analysis," *International Journal of Information Security*, vol. 16, pp. 115-132, 2017.
- [256] C. Jiang and J. Li, "Exploring Global IP-Usage Patterns in Fast-Flux Service Networks," *Journal of Computers*, vol. 12, pp. 371-379, 2017.
- [257] S. Martinez-Bea, S. Castillo-Perez, and J. Garcia-Alfaro, "Real-time malicious fast-flux detection using DNS and bot related features," presented at the Eleventh Annual Conference on Privacy, Security and Trust, Tarragona, Spain 2013.
- [258] E. Stalmans, S. O. Hunter, and B. Irwin, "Geo-spatial autocorrelation as a metric for the detection of Fast-Flux botnet domains," presented at the 2012 Information Security for South Africa, 2012.
- [259] BuiltWith. (2020, March, 2020). *Content Delivery Network Usage Statistics*. Available: <https://trends.builtwith.com/CDN/Content-Delivery-Network>
- [260] SpamHaus. (2020, April, 2021). *Spamhaus Botnet Threat Update: Q1-2020*. Available: <https://www.spamhaus.org/news/article/798/spamhaus-botnet-threat-update-q1-2020>
- [261] T. Alsop. (2021, April, 2021). *Number of data centers worldwide in 2021, by country*. Available: <https://www.statista.com/statistics/1228433/data-centers-worldwide-by-country/>
- [262] MarketWatch. (2021, April, 2021). *Content Delivery Network (CDN) Market Size Analysis by Top Countries 2021: Share Insights with Covid-19 Impact on Global Industry Revenue, Future Growth Rate, Demand Trends and Forecast Research till 2025*. Available: <https://www.marketwatch.com/press-release/content-delivery-network-cdn-market-size-analysis-by-top-countries-2021-share-insights-with-covid-19-impact-on-global-industry-revenue-future-growth-rate-demand-trends-and-forecast-research-till-2025-2021-04-27>
- [263] A. Lipsman. (2019, May, 2020). *Global Ecommerce 2019*. Available: <https://www.emarketer.com/content/global-ecommerce-2019>
- [264] Internet World Stats. (2020, May, 2020). *TOP 20 COUNTRIES WITH THE HIGHEST NUMBER OF INTERNET USERS*. Available: <https://www.internetworldstats.com/top20.htm>

- [265] S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan, "Detecting algorithmically generated domain-flux attacks with DNS traffic analysis," *IEEE/ACM Trans. Netw.*, vol. 20, pp. 1663-1677, 2012.
- [266] Y. Fu, L. Yu, O. Hambolu, I. Ozcelik, B. Husain, J. Sun, *et al.*, "Stealthy domain generation algorithms," *IEEE Transactions on Information Forensics and Security*, vol. 12, pp. 1430-1443, 2017.
- [267] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, pp. 79-86, 1951.
- [268] H. Small, "Co-citation in the scientific literature: A new measure of the relationship between two documents," *Journal of the American Society for information Science*, vol. 24, pp. 265-269, 1973.
- [269] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, 1966, pp. 707-710.
- [270] A. Caglayan, M. Toothaker, D. Drapeau, D. Burke, and G. Eaton, "Behavioral analysis of botnets for threat intelligence," *Inf. Syst. E-bus. Manag.*, vol. 10, pp. 491-519, 2012.
- [271] H. Kordestani and M. Shajari, "An entice resistant automatic phishing detection," in *Proc. 5th Conference on Information and Knowledge Technology (IKT)*, Shiraz, Iran, 2013, pp. 134-139.
- [272] Y.-S. Chen, Y.-H. Yu, H.-S. Liu, and P.-C. Wang, "Detect phishing by checking content consistency," in *Proc. 15th IEEE International Conference on Information Reuse and Integration (IRI)*, Redwood City, CA, USA, 2014, pp. 109-119.
- [273] N. Martins, J. M. Cruz, T. Cruz, and P. H. Abreu, "Adversarial Machine Learning Applied to Intrusion and Malware Scenarios: A Systematic Review," *IEEE Access*, vol. 8, pp. 35403-35419, 2020.
- [274] G. Boesch. (2021, December, 2021). *What Is Adversarial Machine Learning? Attack Methods in 2021*. Available: <https://viso.ai/deep-learning/adversarial-machine-learning/>

Appendixes

Appendix I: Summary of Related Works in the Detection of Phishing Webpages

Category	Work	Feature # and Categories	Data Size (Webpages)	Evaluation Techniques/Algorithms	Performance
Blacklists and Whitelists	Dong, et al. [225]	2 whitelist features	1,463	Whitelist	Acc = 99.14 FPR = 0.14%
Visual similarity	Medvet, et al. [42]	14 webpage content features	82	Visual similarity score on texts and images	FPR = 0% FNR = 7.4%
	Hara, et al. [226]	1 webpage content and 1 URL structure features	2823	Visual similarity score on domains and images	Acc = 80% FPR = 17.5%
	Chen, et al. [44]	1 webpage content feature	306	Contrast Context Histogram (CCH) visual similarity technique	Acc = 95% - 98% FPR = 0.0% - 2.0% FNR = 0.1% - 2.1%
	Zhang, et al. [43]	2 webpage content features	10,272	Bayesian probability and EMD	Acc = 99.68% - 100% FNR = 0% - 1.96%

Offensive defence	Knickerbocker, et al. [45]	1 webpage content and 1 user credentials features	-	Probability computation	-
	Shahriar and Zulkernine [46]	8 webpage content features	52	Simple heuristic combinations	FPR = 0 FNR = 0
Rules	Aburrous, et al. [227]	16 webpage contents, 1 WHOIS records, 7 URL structure, 3 social human factor features	606	Fuzzy algorithms (Ripper, Part, Prism, C4.5)	Acc = 86.3% FPR = 13.6%
	Ma, et al. [120]	5 WHOIS records, 9 URL structure, 1 network, 1 geolocation, 14 webpage reputation (6 blacklists, 1 whitelist, 7 spamAssassin plugin) features	35,500	Naïve Bayes, SVM, Logistic Regression	Error rates = 0.90% – 44.02%
	Xiang, et al. [118]	7 URL structure, 4 webpage contents, 1 WHOIS record and 3 webpage reputation (search engine) features	13,064	SVM, Linear Regression, Bayesian Network, Decision Tree, Random Forest, Adaboost,	Acc = 92.3% FPR = 1.38 FNR = 0.95
	Lakshmi and Vijaya [58]	8 webpage contents, 4 URL structure, 1 WHOIS records, 2 reputation (search engines, blacklist) features	200	Neural Network (MLP), Decision Tree and Naïve Bayesian	Acc = 98.5% FPR = 1.5%

Mohammad, et al. [228]	10 URL structure, 14 webpage structure, 3 social human factor features	1,400	Self-structuring Full Connected Deep Neural Network	Acc = 92.2%
Zuhair, et al. [47]	48 webpage structure and 10 URL structure features	1,000	SVM	FPR = 1.17% FNR = 0.013%
Shirazi H. [48]	30 URL structure, webpage structure, DNS and webpage reputation (Alexa ranking and Google search engine) features	12,000	2 variations of SVM and 4 variations of FC Deep Neural algorithms	TP = 89% TN = 90.3% AUC = 0.9
Zhang, et al. [229]	4 URL structure, 5 webpage structure & contents, 2 WHOIS records, 1 webpage reputation (Alexa ranking) features	5,905	ELM, BP Neural Network, SVM, NB, k-NN, OPELM, Adaboost ELMs, MV-ELMs, LC-ELM	Acc = 99.0% FPR = 0.53%
Jain and Gupta [49]	8 URL structure, 11 webpage structure features	4,059	Random Forest, SVM, Neural Networks, Logistic Regression, Naïve Bayes	Acc = 99.1% FPR = 1.3%
Sahingoz, et al. [50]	1 webpage contents and 40 URL structure features	73,575	Naive Bayes, Random Forest, kNN, Adaboost, K-star, SMO and Decision Tree	Acc = 97.97% Prec = 0.97 Recall = 0.99 F1 = 0.98

	Li, et al. [51]	8 URL structure, 9 webpage structure and 3 webpage content features	2,000	Stack of Gradient Boosting (GB), Extreme GB (XGBoost) and LightGBM	Acc = 97.3% FPR = 4.46% FNR = 1.61%
Hybrid	Gowtham and Krishnamurthi [230]	3 webpage contents, 2 WHOIS records, 8 URL structure, 2 webpage reputation (whitelist, PageRank) features	3,164	SVM	Acc = 99.65% FPR = 0.42% F1 = 0.997

Appendix II: Prediction Features for Detection of Phishing Webpages

Feature #	Feature Name	Feature Description
1	Domain identity in a webpage	Number of times a domain appears in the webpage structure and contents.
2	Domain identity in copyright	Domain in a URL is checked if it matches with the copyright information in the contents or not. If the two are mismatching, the webpage is flagged as a phishing one otherwise it is legitimate webpage.
3	Domain in canonical URL	Domain in a URL is compared against a common domain retrieved from canonical URLs. If the two are mismatching, the webpage is flagged as a phishing one otherwise it is legitimate webpage.
4	Domain in alternate URL	Domain in a URL is compared against a common domain retrieved from alternate URLs. If the two are mismatching, the webpage is flagged as a phishing one otherwise it is legitimate webpage.
5	Foreign domains in links	Domain in a URL is compared against a common domain retrieved from all non-object hyperlinks. If the two are mismatching, the webpage is flagged as a phishing one otherwise it is legitimate webpage.
6	Proportion of void and same webpage links	Ratio of sum of number of void (empty) and number of links pointing to the same webpage divide by the total number of all non-object links.
7	Foreign form handler	Domain of a form handler of a webpage is compared against a domain in URL and a common domain in non-object links.
8	Encoded hostname	Presence of ASCII encoded characters (presented as % followed by two hexadecimal digits) in the hostname is checked. If found, the webpage is flagged as a phishing one otherwise it is legitimate webpage.
9	Encoded URL path	Presence of ASCII encoded characters (presented as % followed by two hexadecimal digits) in the URL path is checked. If found, the webpage is flagged as a phishing one otherwise it is legitimate webpage.

10	Use of @ character in a URL	Presence of @ character or its equivalent ASCII representation (%40) in the URL path is checked. If found, the webpage is flagged as a phishing one otherwise it is legitimate webpage.
11	Domain out of position	The characters <i>http://</i> , <i>https://</i> and <i>www</i> characters and generic or country code Top Level Domain are checked if they have been used more than once in a URL. If not, their positions in the URL will be determined if they are different from the standard ones. If any of the condition is true, the webpage is flagged as a phishing one otherwise it is legitimate webpage.
12	# dots in hostname	Number of dots in a hostname is counted.
13	# dots in the URL path	Number of dots in a URL path is counted.
14	Non-standard port number	For a URL that has used a port number, the number is compared against its http protocol. If the number is not 80 for http and 443 for https, the webpage is flagged as a phishing one otherwise it is legitimate webpage.
15	# obfuscation characters in hostname	Number of ‘_’, ‘-’ and ‘=’ characters in a hostname is counted.
16	# obfuscation characters in URL path	Number of ‘_’, ‘-’ and ‘=’ characters in a URL path is counted.
17	# forward slashes	Number of ‘/’ in a URL is counted.
18	# characters in hostname	Total number of characters in a hostname is counted.
19	# characters in URL path	Total number of characters in a URL path is counted.
20	IP address in a hostname	Presence of an IP address in a hostname is checked. If found, the webpage is flagged as a phishing one otherwise it is legitimate webpage.
21	Numeric in a hostname	Number of numeric characters in a hostname is counted.
22	Numeric in a URL path	Number of numeric characters in a URL path is counted.

23	Shortened URLs	Use of shortened URL is checked comparing a hostname of the URL against a list of 242 hostnames of the collected shortening URL services (see appendix III). If found, the webpage is flagged as a phishing one otherwise it is legitimate webpage.
24	Free domain services	Use of free domain is checked by comparing a domain of a webpage domain against a list of domains of the most abused free domain services we compiled from Anti-Phishing Working Group (APWG)'s reports on global phishing survey between 2008 and 2017. The list is in appendix IV.
25	Domain validity	An expiry date of a webpage's domain registration (from WHOIS database) is compared with the current date to check if it is still valid or not. If it overdue, the webpage is flagged as a phishing one otherwise it is legitimate webpage.
26	Form handler's domain validity	An expiry date of a form handler's domain registration (from WHOIS database) is compared with the current date to check if it is still valid or not. If it overdue, the webpage is flagged as a phishing one otherwise it is legitimate webpage.
27	Domain age	A difference between the current date and the webpage's domain first date of registration (from WHOIS database) is computed.
28	Form handler domain's age	A difference between the current date and the form handler's domain first date of registration (from WHOIS database) is computed.
29	Type of SSL certificate	A type of SSL certificate used by the webpage's domain is determined.
30	Domain, certificate and geolocation country matching	Country names in the ccTLD (for URLs with ccTLDs only), SSL certificate and location of the hosts are compared. If they do not match, the webpage is flagged as a phishing one otherwise it is legitimate webpage.
31	URL ranking in search engines	A URL is searched in the Google and Bing search engines. URLs in the top five results returned by each engine are compared against the searched URL. If none of the results are matching, the webpage is flagged as a phishing one otherwise it is legitimate webpage.

32	Hostname ranking in search engines	A hostname is searched in the Google and Bing search engines. Hostnames in the top five results returned by each engine are compared against the searched hostname. If none of the results are matching, the webpage is flagged as a phishing one otherwise it is legitimate webpage.
33	Domain ranking in search engines	A domain is searched in the Google and Bing search engines. Domains in the top five results returned by each engine are compared against the searched domain. If none of the results are matching, the webpage is flagged as a phishing one otherwise it is legitimate webpage.
34	Counts of matched hostname's IP address in a phishing blacklist of IP addresses	Number of times an IP address of a hostname appears in a list of IP addresses of blacklisted phishing URLs is counted. A 3-month-old data of a blacklist is used.
35	Counts of matched domain's IP address in a phishing blacklist of IP addresses	Number of times an IP address of a domain appears in a list of IP addresses of blacklisted phishing URLs is counted. A 3-month-old data of a blacklist is used.

Appendix III: List of Shortening URL services

1u.ro	cuthut.com	htxt.it	myurl.in	s2r.co	tinyarro.ws	url.inc-x.eu	budurl.co
1url.com	cutt.us	hugeurl.com	ncane.com	s3nt.com	tinytw.it	url4.eu	po.st
2pl.us	cuturl.com	hurl.ws	netnet.me	s7y.us	tinyurl.com	urlborg.com	rebrand.ly
2tu.us	decenturl.com	icanhaz.com	nn.nf	saudim.ac	tl.gd	urlbrief.com	brand.link
3.ly	df9.net	icio.us	o-x.fr	short.ie	tnw.to	urlcut.com	brand.cool
a.gd	digs.by	idek.net	ofl.me	short.to	to.ly	urlhawk.com	snip.ly
a.gg	doiop.com	is.gd	omf.gd	shortna.me	togoto.us	urlkiss.com	short.cm
a.nf	dwarfurl.com	it2.in	ow.ly	shoturl.us	tr.im	urlpire.com	
a2a.me	easyurl.net	ito.mx	oxyz.info	shrinkster.com	tr.my	urlvi.be	
abe5.com	eepurl.com	j.mp	p8g.tw	shrinkurl.us	trcb.me	urlx.ie	
adjix.com	eezurl.com	jijr.com	parv.us	shrtl.com	tumblr.com	uservice.com	
alturl.com	ewerl.com	kissa.be	pic.gd	shw.me	tw0.us	ustre.am	
atu.ca	fa.by	kl.am	ping.fm	simurl.net	tw1.us	virl.com	
awe.sm	fav.me	korta.nu	piurl.com	simurl.org	tw1.us	vl.am	
b23.ru	fb.me	l9k.net	plurl.me	simurl.us	tw2.us	wa9.la	
bc.vc	ff.im	liip.to	pnt.me	sn.im	tw5.us	wapurl.co.uk	
bacn.me	fff.to	liltext.com	poll.fm	sn.vc	tw6.us	wipi.es	
bit.ly	fhurl.com	lin.cr	pop.ly	snipr.com	tw8.us	wkrg.com	
bkite.com	flic.kr	linkbee.com	poprl.com	snipurl.com	tw9.us	wp.me	
blippr.com	flq.us	littleurl.info	post.ly	snurl.com	twa.lk	x.co	

bloat.me	fly2.ws	liurl.cn	posted.at	soo.gd	twd.ly	x.hypem.com	
bt.io	fuseurl.com	ln-s.net	ptiturl.com	sp2.ro	twi.gy	x.se	
budurl.com	fwd4.me	ln-s.ru	qurllyq.com	spedr.com	twit.ac	xav.cc	
buk.me	getir.net	lnkurl.com	rb6.me	starturl.com	twitthis.com	xeeurl.com	
burnurl.com	gl.am	loopt.us	readthis.ca	stickurl.com	twiturl.de	xr.com	
c.shamekh.ws	go.9nl.com	lru.jp	redirects.ca	sturly.com	twitzap.com	xrl.in	
cd4.me	go2.me	lt.tl	redirx.com	su.pr	twtr.us	xrl.us	
chilp.it	golmao.com	lurl.no	relyt.us	t.co	twurl.nl	xurl.jp	
chilp.it	goo.gl	memurl.com	retwt.me	takemyfile.com	u.mavrev.com	xzb.cc	
chs.mx	goshrink.com	migre.me	ri.ms	tcn.ch	u.nu	yatuc.com	
clck.ru	gri.ms	minilien.com	rickroll.it	teq.mx	ub0.cc	ye-s.com	
clicky.me	gurl.es	miniurl.com	rly.cc	thrdl.es	updating.me	yep.it	
cli.gs	hellotxt.com	miniurls.org	rsmonkey.com	tighturl.com	url.ca	yfrog.com	
clickthru.ca	hex.io	minurl.fr	rubyurl.com	tiny.cc	url.co.uk	zi.pe	
cort.as	href.in	moourl.com	rurl.org	tiny.pl	url.ie	zz.gd	

Appendix IV: Most Abused Free Domain Services Reported by APWG Between 2008 – 2017

No.	2008		2009		2010		2011	
	1 HF	2 HF	1 HF	2 HF	1 HF	2 HF	1 HF	2 HF
1	.pochta.ru	.ns10-wistee.fr	.ns10-wistee.fr	.t35.com	.t35.com	.co.cc	.co.cc	.osa.pl
2	.land.ru	.olympenetwork.com	.t35.com	.110mb.com	.110mb.com	.t35.com	.t35.com	.ce.ms
3	.ns8-wistee.fr	.by.ru	.nm.ru	.ns11-wistee.fr	.justfree.com	.110mb.com	.osa.pl	.cx.cc
4	.9k.com	.t35.com	.blackapplehost.com	.tripod.com	.notlong.com	.altervista.org	.cn.la	.co.cc
5	.altervista.org	.powa.fr	.110mb.com	.justfree.com	.tripod.com	.yourfreehosting.net	.mu.la	.cu.cc
6	.mtp.ru	.nm.ru	.pochta.ru	.co.cc	.altervista.org	.hdfree.com.br	.altervista.org	.bij.pl
7	.free.fr	.free.fr	.pop3.ru	.freehostia.com	.freewebhostx.com	.tripod.com	.co.tv	.cn.im
8	.nm.ru	.altervista.org	.justfree.com	.angelfire.com	.limewebs.com	.somee.com	.vv.cc	.altervista.org
9	.t35.com	.javabien.fr	.by.ru	.50webs.com	.eb2a.com	.my3gb.com	.ce.ms	.co.tv
10	.jexiste.fr	.ns8-wistee.fr	.free.fr	.dezipner.ru	.yourfreehosting.net	.freewebhostx.com	.cn.im	.gv.vg
11	.110mb.com	.cfun.fr	.freehostia.com	.freewebhostx.com	.co.cc	.hd1.com.br	.bee.pl	.cz.cc
12	.front.ru	.tripod.com	.tripod.com	.hostrator.com	.freehostia.com	.justfree.com	.somee.com	.0fees.net
13	.krovatka.su	.freehostia.com	.aplus.net	.free.fr	.50webs.com	.solidwebhost.com	.110mb.com	.tripod.com
14	.notlong.com	.pochta.ru	.land.ru	.pochta.ru	.hd1.com.br	.001webs.com	.gv.vg	.vv.cc
15	.freeweb7.com	.9k.com	.uol.com.br	.blackapplehost.com	.hdfree.com.br	.x10.mx	.5gbfree.com	.webs.com
16	.freehostia.com	.bluechiposting.com	.bplaced.net	.hd1.com.br	.webcindario.com	.webs.com	.co.be	
17	.us.com	.siteburg.com	.altervista.org	.atSPACE.com	.pochta.ru	.webcindario.com	.webcindario.com	

18	.de.com	.110mb.com	.co.cc	.psem.su	.x10hosting.com	.zxq.net	.hdfree.com.br	
19	.ifrance.com	.land.ru	.hostrator.com	.w.interia.pl	.my3gb.com	.notlong.com	.com3.tw	
20	.host.sk	.vndv.com	.50webs.com	.rbcmil.ru	.zapro.org	.hut2.ru	.free.fr	

No.	2012		2013		2014		2017
	1 HF	2 HF	1 HF	2 HF	1 HF	2 HF	
1	.bee.pl	.freeavailabledomains.com	.net.tf	.esy.es	.16mb.com	.altervista.org	.dyndns.org
2	.osa.pl	.1004web.com	.3owl.com	.popnic.com	altervista.org	.dyndns.org	.jino.ru
3	.freeavailabledomains.com	.000webhost.com	.usa.cc	.altervista.org	.tf	.1freehosting.com	.uCoz
4	.x90x.net	.altervista.org	.nazuka.net	.000webhost.com	co.vu	.2freehosting.com	.000webhost.com
5	.serversfree.com	.3owl.com	.altervista.org	.uni.me	de.vu	.000webhost.com	.co.ua
6	.nazuka.net	.1freehosting.com	.my3gb.com	.1freehosting.com	.96.it		.altervista.org
7	.altervista.org	.cydots.com	.kmdns.net	.2freehosting.com	.hol.es		.2freehosting.com
8	.blo.pl	.dyndns.org	.3eeweb.com	.serversfree.com	.pe.hu		.no-ip.com
9	.ias3.com	.likedns.cn	.p.ht	.ucoz.com	.890m.com		.runhosting.com
10	.linkpc.net	.co.cc	.cixx6.com	.pubyun.com			.websitewelcome.com
11	.pubyun.com	.ucoz.com	.wink.ws	.my3gb.com			.smarterasp.net
12	.azuka.biz	.homenic.com	.instantfreesite.com	.co.vu			
13	.co.cc	.main--hosting.com	.5gbfree.com	.3owl.com			
14	.loxblog.com	.alpennic.com	.chickenkiller.com	.nic.de.vu			
15	.cu.cc	.nazuka.net	.fav.cc	.at.vu			
16	.1freehosting.com	.pubyun.com	.ias3.com				

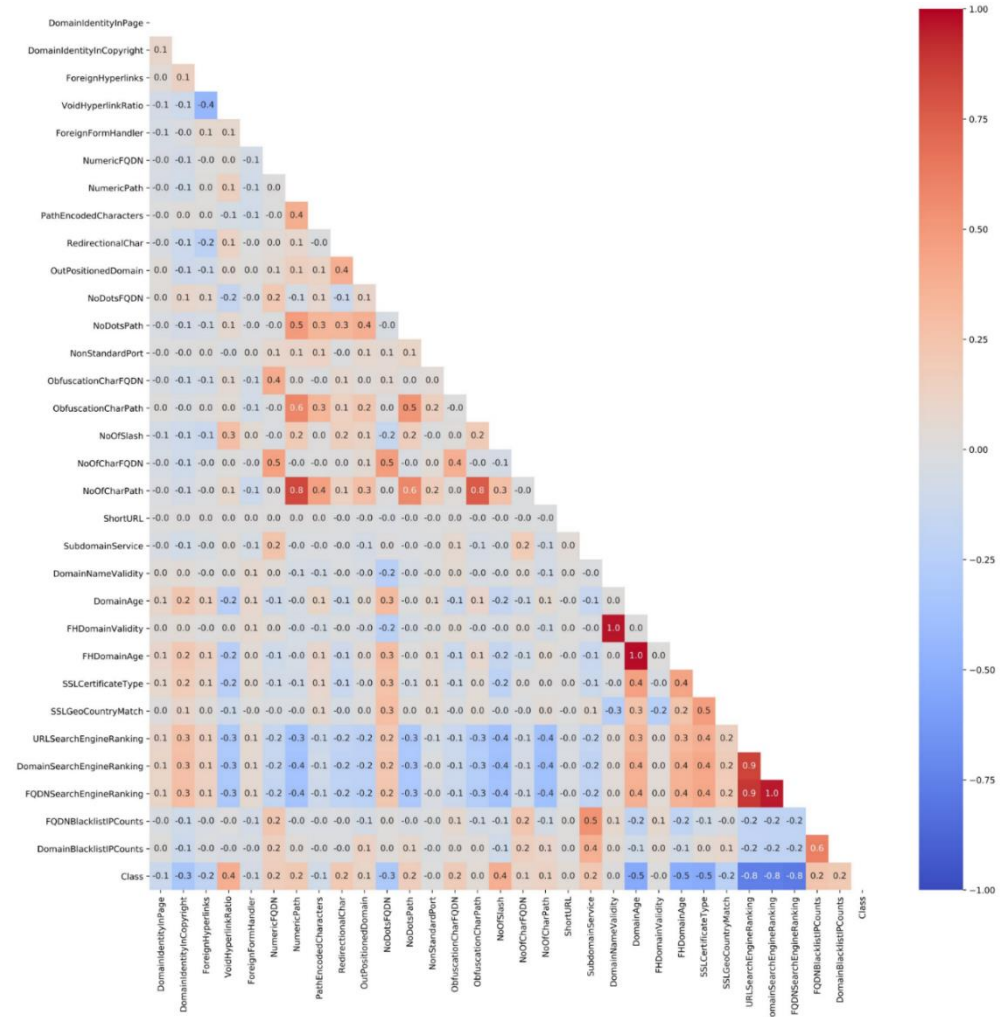
17	.r.gd	.5gbfree.com	.co.vu				
18	.tripod.com	.ias3.com	.oicp.net				
19	.3owl.com	.blogspot.com	.hol.es				
20	.ce.ms	.ripod.com	.vicp.cc				
21		.no-ip.com					

Other Known Abused Free Domain Services

.asia.gp	.eu.org	.1sta.com	.funurl.com	.at.tf	.us.tt	.freesubs.de
.online.gp	.cu.cc	.24ex.com	.alturl.com	.ch.tf	.uk.tt	.js4.de
.biz.uz	.freeavailabledomains.com	.hitart.com	.hereweb.com	.edu.tf	.ca.tt	.lz3.de
.co.gp	.usa.cc	.bigbig.com	.co.cc	.ru.tf	.eu.tt	.mp3d.de
.com.nu	.flu.cc	.2freedom.com	.co.nr	.pl.tf	.es.tt	.rocken.de
.eu.nu	.nut.cc	.2fortune.com	.dom.ir	.cs.tf	.fr.tt	.rockt.de
.mobi.ps	.igg.biz	.2truth.com	.2ir.ir	.bg.tf	.it.tt	.rockz.de
.name.vu	.4fd.us	.2savvy.com	.coo.ir	.sg.tf	.se.tt	.rulen.de
.pro.vg	.aa.am	.2tunes.com	.ne1.net	.de.lv	.dk.tt	.rult.de
.tv.gg	.tk	.2fear.com	.0vr.net	.at.lv	.be.tt	.rulz.de
.us.nf	.ml	.2hell.com	.r8.org	.ch.lv	.de.tt	.s3p.de
.web.gg	.ga	.mirrorz.com	.net.tf	.ar.at.lv	.at.tt	.sucken.de
.info.uu	.cf	.echos.com	.eu.tf	.co.at.lv	.au.tt	.suckt.de
.ovh.co.uk	.gq	.ebored.com	.us.tf	.at.gg	.co.uk.tt	.suckz.de

.ovh.ie	con.nr	.antiblog.com	.int.tf	.ch.gg	.com.au.tt	.uk8.de
.me	.shorturl.com	.dealtap.com	.ca.tf	.de.gg	.4bundeskazler.de	.000webhostapp.com
.2ya.com	.vze.com	.filetap.com	.de.tf		.cybermonn2000.de	

Appendix V: Correlation Matrix of Features for Detection of Phishing Webpages



Appendix VI: Summary of Related Works in the Detection of Malicious Fast Flux Hostnames

Work	Feature # and Categories	Data size (URLs)	Classification Type	Evaluation Algorithms	Performance	Detection Time
Passerini, et al. [129]	5 network, 2 temporal, 4 DNS and 1 reputation features	125	Binary classification (FF hostnames versus benign hostnames)	Naïve Bayes	FPR = 0% FNR = 0%	3 hours
Perdisci, et al. [131]	2 temporal, 5 DNS, 2 network and 1 spatial features	40k to 60k per day	Binary classification (FF hostnames versus non-FF hostnames)	Decision Tree	Acc = 99.7% FPR = 0.3% AUC = 0.992	24 hours
Kumar and Xu [253]	2 temporal, 1 network, 1 reputation and 3 DNS features	93,283	Binary classification (FF hostnames versus non-FF hostnames)	SVM	Acc = 88.03%	24 hours
Chen, et al. [254]	3 DNS features	6,986	Multi-class classification (FF hostnames versus non-FF malicious, CDN and non-CDN hostnames)	LSTM	Acc = 95.39%	Unknown
Huang, et al. [255]	2 temporal, 2 spatial, 2 DNS features	133,629	Binary classification (FF hostnames versus benign hostnames)	Bayesian network	Acc = 98.16% FPR = 0.39% AUC = 0.984	0.5 sec

Hsu, et al. [137]	2 temporal features, 4 host features	17, 214	Binary classification (FF bots versus benign server)	SVM	Acc. = 95% AUC = 0.993	Unknown
Wang, et al. [256]	2 temporal, 2 DNS and 2 spatial features	58,723	Binary classification (FF hostnames versus benign hostnames)	Bayesian network	Acc = 96.7% FPR = 4.4% F1 = 0.971 AUC = 0.982	Unknown
Lin, et al. [128]	3 DNS features, 1 temporal feature	12,952	Binary classification (FF hostnames versus benign hostnames)	Genetic algorithm	Acc. = 98.2% FPR = 1.78%	18.54 sec
Hsu, et al. [134]	1 temporal	150	Binary classification (FF hostnames versus benign hostnames)	FF-score computation	FPR = 0.3% FNR = 2%	Unknown
Stevanovic, et al. [257]	25 DNS, 8 network and 6 reputation features	2,756	Binary classification (FF hostnames versus benign hostnames)	Random Forest	Prec. = 0.90 Recall = 0.804 F1 = 0.849	Unknown
Jiang and Li [258]	2 DNS features, 1 network feature, 1 temporal feature, 1 spatial feature	26,873	Binary classification (FF hostnames versus benign hostnames)	SVM, Naïve Bayes, K-NN	Acc. = 96.7% Prec. = 0.965 Recall = 0.981 F1 = 0.973 AUC = 0.989	400 sec

Almomani [130]	11 DNS features, 3 temporal features	7,615	Binary classification (FF hostnames versus benign hostnames)	Adaptive evolving fuzzy neural network, SVM, Naïve Bayes	Acc. = 98% Error rate = 1 – 16 (%)	Unknown
----------------	--------------------------------------	-------	--	--	---------------------------------------	---------

Appendix VII: Summary of Related Works in Detection of Malicious NS Flux Hostnames

Work	Feature # and Categories	Data size (URLs)	Classification Type	Evaluation Algorithms	Performance	Detection Time
Salusky and Danford [136]	1 DNS feature	-	Binary classification (NS IP flux hostnames versus non-NS IP flux hostnames)	-	-	-
Konte, et al. [143]	2 temporal, 3 DNS, 2 network, 1 spatial features	3,860	Multi-class classification (NS IP flux and NS name flux hostnames versus benign hostnames)	-	-	-
Metcalfe and Spring [141]	1 DNS and 1 temporal features	4,722,495	Binary classification (NS IP flux hostnames versus non-NS IP flux hostnames)	-	-	-
Kadir, et al. [142]	7 DNS features	500	Binary-class classification (NS IP flux and versus benign hostnames)	KNN	FPR = 0% FNR = 0%	3 months
Pa, et al. [140]	3 DNS features	50,030	Binary classification (NS IP flux hostnames versus non-NS IP flux hostnames)	Mappings of IP and hostnames of NSs	FPR = 0.8%	6 months

Appendix VIII: Correlation Matrix of Features for Detection of Phishing NS Flux Hostnames

