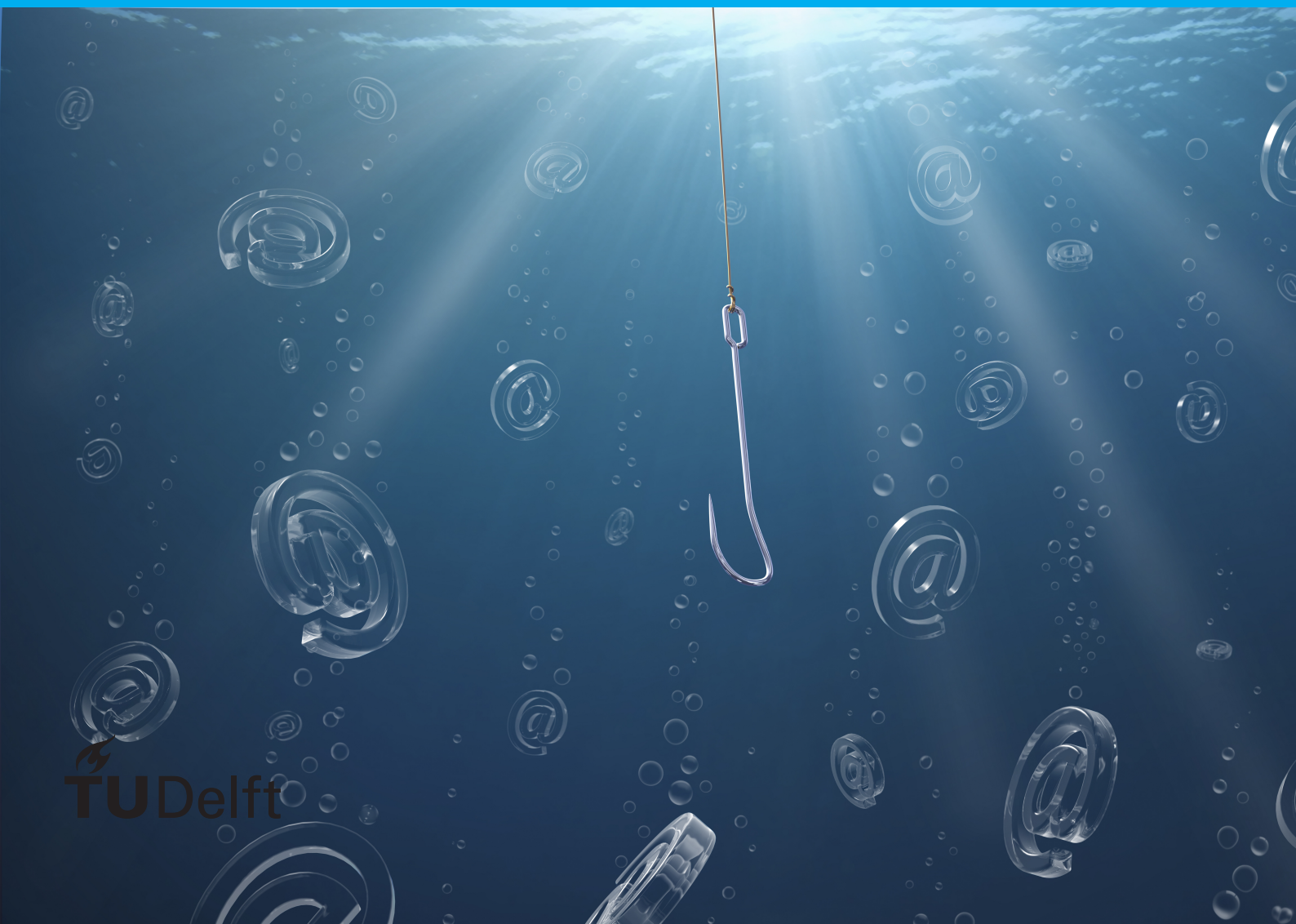
HoPhiDeS

Holistic Phishing Detection System

A.T. Mulder

Master Thesis report
Computer Science
Faculty of EEMCS
Cyber Security Group



TU Delft

HoPhiDeS

Holistic Phishing Detection System

by

A.T. Mulder

Student number: 4372166
Project duration: May 3, 2021 – May 20, 2022
Thesis committee: Assistant Prof. Dr. A. Zarras, TU Delft, supervisor
Prof. dr. ir. G. Smaragdakis, TU Delft
Associate Prof. dr. ir. A. Anand, TU Delft
Cover image: Jan Willem [58]

TU Delft

Abstract

From 2019 into 2021, phishing incidents tripled, while phishing was found to be the largest used cyber-attack vector. One commonly used type of phishing is a credential phishing attack over email. This is an act where an attacker tries to steal credential information from a target via a phishing website. This phishing website is delivered to the target over email. Even though much research has been done on the detection of such attacks, this rise in incidents was not prevented. In this thesis, it was found that a credential phishing attack over email consists of four phases, with an important phase being the attack conduction phase. This attack conduction phase comprises four stages. Most phishing detection systems focus on two of these stages, the 'email received' stage and the 'website visited' stage. Though, all research focused on either of the two stages, not a combination of both. Every research handled phishing emails and phishing websites as isolated entities. Also for any other stage, no research on detection was found comprising more than one stage in any of the phases in the credential phishing attack chain.

To perform this research, a dataset containing both benign and phishing emails has been selected. The dataset originates from a combination of an openly available dataset and private datasets. Each email used originates from the period from 2015 up to 2022.

To increase phishing detection capabilities, a more holistic approach to the phishing attack chain was introduced in this thesis. Emails and any (phishing) websites in that email were handled as one entity. This opened up new phishing detection capabilities. Features have been identified that made, amongst other aspects, use of brand recognition in both email and website. To extract these features, a brand recognition system has been created. This system is able to detect the logos of brands in emails and websites and name the brand associated with that logo. The system consists of a combination of neural networks and Google reverse image search. An XGBoost algorithm has been built on top of these brand recognition features. This resulted in a phishing detection system which was able to predict phishing emails from the dataset with a precision of .834 and a recall of .836. The extraction of the brand recognition features took on average 70.18 seconds per email. Though, this execution time grows linearly with the number of URLs present in an email. By building a phishing detection system of features using information from a combination of email domains, website domains, website URL redirects, certificates, email content such as brand names in text, domain registration dates, DMARC and SPF, an XGBoost model could be created with a precision of .957 and a recall of .943. The extraction of those features took 25.57 seconds on average. By combing all feature types (both brand recognition and non-brand recognition types), a precision of .967 and a recall of .947 could be obtained with an XGBoost model. This was called the 'Fusion module'. This Fusion module thus shows very decent phishing detection capabilities on the dataset.

Using the same holistic approach as before, another phishing detection system was proposed. For this holistic approach, systems performing phishing detection at several stages in the credential phishing attack chain were combined into a single classification system. A Structure module has been designed and created, performing phishing detection solely on emails. This 'Structure module' used 68 features focusing on the structural aspects of an email. The best classification performance of this model was obtained with an XGBoost algorithm, which had a precision of .977 and a recall of .957. A 'Website module' performed phishing detection on whether a website was likely to be a phishing website. It used a simpler version of the brand recognition system used in the Fusion module. This module obtained a precision score of .661 and a recall of .775. The prediction of these two models for an email, along with the prediction of the Fusion module, was used as input for another machine learning model. A Support Vector Machine model was selected, and this model was then able to perform phishing detection on the dataset with a precision of .985 and a recall of .974. This holistic system thus showed improved performance over the individual prediction systems. The holistic system was able to increase overall phishing detection by combining phishing detection systems from different stages in a credential phishing attack over email.

Contents

1 Introduction **1**
1.1 Definition of phishing . . . 1
1.1.1 Credential phishing attack over email . . . 2
1.2 Research questions . . . 3
1.3 Contributions . . . 4
1.4 Thesis outline . . . 4

2 Background **5**
2.1 Brand impersonation in email . . . 5
2.2 Machine learning . . . 5
2.2.1 Features . . . 5
2.2.2 Ensemble classifiers . . . 5
2.2.3 Meta-learning . . . 6
2.2.4 Machine learning models . . . 6
2.2.5 Class weight using machine learning . . . 8
2.3 Validation methods . . . 9
2.3.1 Precision-recall curve . . . 9
2.3.2 Validation methods for object detection . . . 10
2.3.3 Average Precision for Object detection systems . . . 11
2.3.4 k-fold cross-validation . . . 11
2.4 Email authentication protocols . . . 12
2.4.1 Sender Policy Framework (SPF) . . . 12
2.4.2 DomainKeys Identified Mail (DKIM) . . . 12
2.4.3 Domain-based Message Authentication, Reporting and Conformance (DMARC) . . . 12
2.5 Phishpedia . . . 12
2.5.1 Detection of logos . . . 12
2.5.2 Brand identification . . . 13
2.5.3 Domain validation . . . 15
2.6 Feature Analysis . . . 15
2.6.1 General statistics . . . 15
2.6.2 Feature ranking . . . 16
2.6.3 Impact on the performance of removing features . . . 16

3 Related work **17**
3.1 Phishing attack process . . . 17
3.1.1 Phishing phases in a general phishing attack . . . 17
3.1.2 Phishing stages of a phishing website . . . 19
3.2 Phishing detection . . . 19
3.2.1 Phishing email detection . . . 19
3.2.2 Phishing web page detection . . . 20

4 Phishing attack chain for a credential phishing attack over email **23**
4.1 Knowledge gap . . . 23
4.2 Phishing attack anatomy . . . 23
4.3 Attack phases . . . 23
4.3.1 Planning phase . . . 24
4.3.2 Preparation phase . . . 24
4.3.3 Attack conduction phase . . . 24
4.3.4 Valuables Acquisition phase . . . 24

4.4 Detection possibilities . . . 25
4.5 No research which overlaps stages in the attack conduction phase . . . 25
4.6 Combining detection models in a holistic detection system . . . 26

5 Datasets **27**
5.1 Email datasets used for Fusion module and Holistic System Validation . . . 27
5.1.1 Nazario . . . 27
5.1.2 External . . . 27
5.1.3 Eye . . . 28
5.1.4 Personal . . . 28
5.1.5 Dataset enrichment . . . 29
5.2 Fusion Module datasets . . . 30
5.2.1 Logo detection dataset . . . 30
5.2.2 Brand identification dataset . . . 31
5.2.3 Brand recognition dataset . . . 32
5.2.4 Reverse image search datasets . . . 33

6 Phishing attack detection by merging attack stages **35**
6.1 Methodology . . . 35
6.1.1 Brand recognition system . . . 35
6.1.2 Feature engineering . . . 38
6.1.3 Machine learning model . . . 46
6.2 Results . . . 47
6.2.1 Brand recognition system validation . . . 47
6.2.2 Feature analysis Fusion module . . . 54
6.2.3 Performance Fusion module . . . 58

7 Holistic detection system **69**
7.1 Methodology . . . 69
7.1.1 General architecture of the holistic detection system . . . 69
7.1.2 Email structure module . . . 71
7.1.3 Website module . . . 71
7.2 Results . . . 74
7.2.1 Structure module validation . . . 74
7.2.2 Website module validation . . . 79
7.2.3 Holistic system validation . . . 84

8 Discussion and conclusion **93**
8.1 Answering the research questions . . . 93
8.2 Conclusion . . . 94
8.3 Discussion . . . 94
8.3.1 Phishing email dataset . . . 95
8.3.2 Adversarial attack on the brand recognition system . . . 95
8.3.3 Whitelisting domains for the Website module . . . 95
8.3.4 Computation time of the Fusion module . . . 95
8.4 Future work . . . 96

Appendices **97**

A General feature analysis structure module **99**

B Common email service providers used in Fusion module **107**

C Social media brands used in Fusion module **109**

D General feature analysis Fusion module **111**

E Deep neural network for structure module **117**

F Unused features for the Fusion module **121**

List of Figures

1.1 Phishing attack types and techniques [2] 2

2.1 Intersection over Union is the area of overlap between two shapes, divided by the area of union between these two shapes [27] 10
2.2 Phishpedia object detection model [35] 13
2.3 Region Proposal Network [46] 14
2.4 Siamese Model from [35] 14
2.5 Logo Variants for Adobe [35] 15

3.1 Phishing phases by [2] 18
3.2 Phishing stages by [42] 19

4.1 Phishing attack phases for credential phishing attack over email 24
4.2 Attack conduction phase enlarged 24
4.3 Attack phases and phishing detection mechanisms Stringhini et al.[50], Lee et al.[34], Smadi et al.[49], Fang et al.[22], Jain et al. [30], Oest et al. [42], Zhou et al. [63], Xiang et al. [60], Tan et al. [52], Ding et al. [18], Sahingoz et al. [47], Zhang et al. [62], Drury et al. [19], Lin et al. [35], Bozkir et al. [7] 25

5.1 Screenshot of an email from the personal dataset 29
5.2 Flow for deciding which URL to enrich with screenshot 30
5.3 Phishing page example 30
5.4 Phishing page example 31
5.5 Screenshot of web page without logo from logo detection dataset 31
5.6 Screenshot of web page with a logo from the logo detection dataset 32
5.7 Example of a ”non-logo” from the created dataset, detected by the logo detection system. 33
5.8 Example of a logo, detected by the logo detection system 33

6.1 Brand recognition system architecture 36
6.2 Logo detection model. The code for the purple blocks has been created during this thesis. The code for the red blocks has been created by Phishpedia [35], and the code for blue block is code from Phishpedia, with improvements made during this thesis. 36
6.3 Logo vector representation model 37
6.4 Brand identification model 37
6.5 Extended brand recognition from a high level 38
6.6 Extended brand recognition model 39
6.7 precision-recall curve for the logo detection model 49
6.8 precision-recall curve for the brand identification model 50
6.9 Poor quality phishing example 52
6.10 Performance of reverse image search on the dataset with logos and random pictures . . . 54
6.11 Performance of reverse image search on the dataset with logos and predicted boxes produced by the logo detection model 55
6.12 Feature importance for all features Fusion module 56
6.13 Feature importance with most important features removed for the Fusion module 56
6.14 Precision development when removing features for the Fusion module. Experiment has been done 5 times, the results are averaged. 59
6.15 Recall development when removing features for the Fusion module. Experiment has been done 5 times, the results are averaged. 59

6.16 Accuracy development when removing features for the Fusion module. Experiment has been done 5 times, the results are averaged. 60
6.17 Precision recall curve for Fusion module while adjusting the weights for the positive classification class 63
6.18 Average execution time of the Fusion module for Phish, benign and combined 64
6.19 Distribution of samples with an execution time within a time interval for the Fusion module. 64
6.20 The ratio of total execution time for each sub-part in the Fusion module. 65
6.21 The number of URLs present in an email shows a correlation with the execution time of the Fusion module on the email. Do note that several anomalies have been removed for the visibility of the graph. 67

7.1 The architectural overview for the holistic system 69
7.2 Regression score for different formulas for $g(\alpha)$ 74
7.3 Feature importance for all features for the Structure module 75
7.4 Feature importance with most important feature removed for the Structure module . . . 75
7.5 Precision development when removing features for the Structure module. The experiment has been done 5 times, the results are averaged. 76
7.6 Recall development when removing features for the Structure module. The Experiment has been done 5 times, the results are averaged. 76
7.7 Accuracy development when removing features for the Structure module. The Experiment has been done 5 times, the results are averaged. 77
7.8 Precision recall curve for Structure module while adjusting the weights for the positive classification class 79
7.9 Average time per email class for the Structure module to create its features 80
7.10 Distribution of samples with an execution time within a time interval for the Structure module. 80
7.11 Prediction loss for different similarity thresholds. Run with a similarity cap of .85. 82
7.12 Average time per email class for the Website module to make a prediction 83
7.13 Distribution of samples with an execution time within a time interval for the Website module 84
7.14 The feature importance for the meta classifier, where each feature is the regression score of one of the modules. From left to right: structure module, fusion module and website module 84
7.15 Precision recall curve for holistic module while adjusting the weights for the positive classification class. 87
7.16 Email used for qualitative analysis. Email from every scenario has an identical layout . . 89

E.1 Count of recall values per configuration for the lower segment of the recall range 119
E.2 Count of recall values per configuration for the higher segment of the recall range. 119

List of Tables

5.1 Email datasets used for validation of different systems 27
5.3 Non-email datasets used during the validation of sub-systems used in the Fusion module 31
5.4 Number of emails per source in the dataset for Brand Recognition 32

6.1 Sub-categories for Fusion module features 41
6.2 Feature list for the fusion module 44
6.3 Logo detection performance results for different IoU thresholds 48
6.4 Brand identification performance results for different similarity thresholds 49
6.5 Brand identification performance results for different similarity thresholds, zoomed in on range 0.8-0.845 50
6.6 Per sample category for combined performance test, number of samples and the number of correct predictions done by the Brand recognition system 51
6.7 Confusion matrix for the test on the combined performance of the brand recognition system 51
6.8 Per sample category for test 2, number of samples and the number of correct predictions done by the Brand recognition system 53
6.9 Analysed statistics for test 2 53
6.10 Zero occurrences for three continuous features from the Fusion module. 56
6.11 Top 5 best performing features from the Fusion module 57
6.12 Certificate set types present in the different classes of the dataset 58
6.13 Average feature ranking per sub-category 58
6.14 Performance per machine learning model for the Fusion module 60
6.15 Misclassified samples per machine learning model for the Fusion module 61
6.16 Performance of the Fusion module without using the brand recognition features 61
6.17 Performance of the Fusion module while using only brand recognition features 62
6.18 Overview of the performance when splitting the features into different machine learning models. 62
6.19 Performance of the machine learning models for the Fusion module with the best performing parameter settings 63
6.20 Feature ranking of features which use either the brand recognition or the logo detection system 66

7.1 Desired outcome of the regression formula 72
7.2 Performance per machine learning model for the Structure module 77
7.3 Misclassified samples per machine learning model for each dataset for the Structure module 78
7.4 Performance of the machine learning models for the Structure module with the best performing parameter settings 79
7.5 Performance for different maximum similarity threshold 81
7.6 Performance for varying similarity thresholds 81
7.7 Results for using different regression score formulas. 82
7.8 Misclassified samples for the Website module per dataset. 82
7.9 Top 10 brands producing false positives. 83
7.10 Performance per machine learning model for the holistic system 85
7.11 Misclassified samples per machine learning model for each dataset for the holistic system 85
7.12 Performance of the machine learning models for the holistic system with the best performing parameter settings 86
7.13 Performance when weighting classes. The columns represent for which machine learning models the weighting has been applied. 87

7.14 Performance of a Random Forest regression predictor for varying threshold 88
7.15 Predictions for the modified emails for the qualitative analysis 90

A.1 General statistics structure module features. 99
A.2 Feature ranking structure module. 103

D.1 General statistics Fusion module features. 111
D.2 Feature ranking for Fusion module. 114

E.1 Averaged performance of random layer configurations. 118
E.2 Maximum values of performance of random layer configurations. 118

F.1 Unused features for the Fusion module. 121

1 Introduction

At the dawn of the invention of the internet in the 1960s, electronic mail (email) was already in limited use as a means of communication. At that time, 'email' functioned by giving a user the rights to append text to a dedicated file, but not the rights to read or modify that file. Thus, an 'email' could only be sent to users on the same computer [54]. In 1971, Ray Tomlinson, a researcher at MIT combined a file-transferring program with the protocol of the early email, making it possible to send emails to other machines. By means of identifying to which network an email should be sent, he distinguished the 'user' from the 'host' with the now commonly used @ symbol. Email was born. Until the late 1980s, email was used primarily in computer science, governmental and business circles. Subsequently, email became popular among other users, leading to wider use [20]. Somewhere in the 1990s though, the widely used communication method was being abused by malicious users in the first "phishing attacks". In 2000, the larger public came to know about the new type of attack with the Love Bug worm[13] [15]. Now, 30 years later, phishing is far from distortion. In fact, according to the FBI, phishing attacks are rising in numbers and impact [28], and are the largest used attack vector for gaining a foothold in a cyber attack. The FBI reported almost three times as many incidents for 2021 compared to 2019 [29]. Compared to 2017, the number of incidents even rose with 1200%. To be able to cope with this rise, phishing detection capabilities should be increased.

1.1. Definition of phishing

Phishing can be defined as follows: **Phishing is a scalable act of deception whereby impersonation is used to obtain information from a target [33]**. Since such an act has malicious intent, the act is often referred to as a phishing attack. The impersonation mentioned in this definition means that the attacker performing such a phishing attempt is trying to impersonate a source which the target knows, and relies on. By this impersonation, the attacker can try to extract information from the victim, e.g. login credentials. The attacker could also extend the trust that the target has in the impersonated source to extract information which is not at all related to the source. In such a case, the attacker could for example impersonate Microsoft and ask the target for its social security number.

Such an act of deception can happen through many mediums. A large part of phishing attempts use email as an attack vector for impersonation, but other phishing attempts happen over social media or via phone. In Figure 1.1, phishing attack types and techniques are categorised. When performing a phishing attack, there are two largely different attack types. 'Deceptive Attack' and 'Technical Subterfuge'.

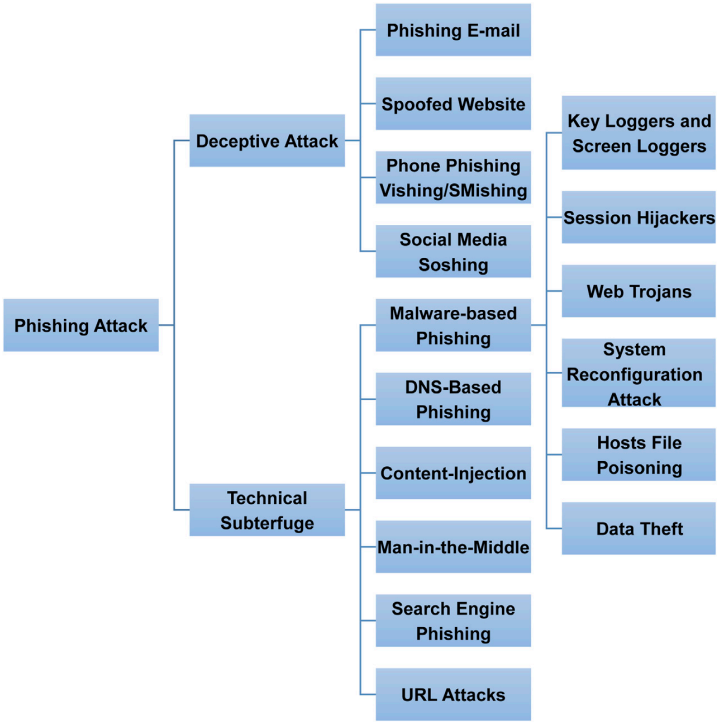


Figure 1.1: Phishing attack types and techniques [2]

1.1.1. Credential phishing attack over email

Taking into consideration the general definition of phishing, the term phishing in this thesis will be scoped. This thesis will focus on phishing attacks over email, with the attempt of tricking a user to visit a web page and fill in login credentials of some sort. This relates to a sub-category of the 'deceptive phishing' attack type from Figure 1.1 [2]. This research focuses on a phishing attempt where the user is tricked into visiting a hosted web page. Phishing emails without URLs or phishing emails making use of attachments are not taken into consideration.
Deceptive email phishing is the most common threat derived by an attacker [2]. Such an attack usually consists of 4 steps.

1. Setting up a fraudulent email containing link(s). The email will try and trick the user to believe its sender has good intent. The attacker can do this for example by making the email look like it is coming from a source the target probably knows and trusts. e.g. by placing logos of a large brand in the email. The goal of the attacker is to trick the target into clicking the link to the phishing site in the email.

2. Send the email.

3. If the link is clicked by the target, the target will be sent to a malicious website. This malicious website is operated by the attacker. This website will again try to trick the target into believing it has good intent. Phishing websites often look like benign websites of large brands.

4. The malicious website prompts for information or credentials. The target will be under the assumption that it fills in its credentials on the original website of the brand impersonated by the phishing website. In fact, it sends its credentials to the attacker.

After successful execution of an attack, the attacker has obtained credentials of a user to a certain website. These credentials can then be misused by the attacker for malicious intent. Due to the medium used by the attack and the goal of the attacker to obtain credentials from a user, this specific attack is a **credential phishing attack**, executed over **email**.

1.2. Research questions

Since phishing has been around for several decades, a lot of studies have been done on several aspects of a phishing attack. Ranging from the psychological investigations on the motivation of attackers to the best ways to detect phishing attacks. Phishing attacks are changing rapidly and attackers continuously adapt state-of-the-art techniques to accomplish their malicious goals. Even though email is relatively old, it ages well: the layout of emails is improving along with new frameworks and technology from general web applications. There is constant development in the commercial use of software such as newsletters containing sophisticated marketing tracking techniques. Next to changes in email, the quality of layout and complexity of web pages has been increasing. The layout quality of Web pages trying to trick victims has also been increasing accordingly.

Even though much research has been done on phishing detection, phishing attacks still remain a large and increasing threat. The evolution of phishing attacks is mainly driven by the continuous improvement of detection systems. This ever-continuing arms race drives innovation on both the attacker and defender sides.

Abundant research has resulted in excellent detection mechanisms which are able to analyse email received and predict whether it is a phishing email or not. Next to detection mechanisms on emails, many proposals for phishing detection on web pages have been done.

Emails and websites differ enormously in both architecture and inner workings. Therefore, it might be intuitive to perform phishing detection on either email or a web page. Even so, the phishing email and phishing web page have their use in different moments during a phishing attack. Though, when considering the whole email credential phishing attack, the phishing web page and phishing email are inseparable. In this specific attack, the attacker relies on the email to deliver the website URL to the victim. It then relies on the website to steal the credentials from the victim. To the best of my knowledge, no effort has been done to perform phishing detection on information available after fusing the isolated entities of website and email together.

This research will be performed with a single defined goal:

To investigate phishing detection capabilities of taking a holistic approach to a credential phishing attack.

To reach this goal, three research questions will be answered in this thesis.

Q1: What different phases occur in a credential phishing attack? First, a clear identification of the phishing attack chain will be performed. This research question will be answered in Chapter 4.

Q2: How to create a phishing detection mechanism which uses information from different stages of a phishing attack?. In Chapter 6, features will be extracted using information from different attack stages. On these features, several machine learning models will be trained and identified to perform phishing detection.

Q3: How to combine phishing detection mechanisms from different stages into a holistic detection mechanism? This research question will be answered in Chapter 7, in which several detection mechanisms from different stages in the attack conduction phase of a phishing attack are combined into a single classifier.

The outcome of this research will be a new phishing detection system. The new phishing detection system has been built from the idea that a good detection system considers the whole phishing attack as a single entity without isolated stages. By no longer performing phishing detection on either emails or websites, a new landscape of possibly interesting information opens up. Lastly, another phishing detection system is created by combining several phishing detection mechanisms together. This research will show that enabling phishing detection systems for email and websites to work together in combination with the use of the new landscape of information, results in a decent phishing detection mechanism. Also, a clear overview of the phishing attack chain for a deceptive email phishing attack will be introduced.

1.3. Contributions

The contributions of this study to the academic community could be summarised as a new approach to a widely investigated problem with proof of good results to this approach on our dataset. More specifically, the contributions are listed in the following list:

- Clear overview of the phishing attack chain for a deceptive phishing attack conducted over email.
- A new approach to phishing detection has been introduced. By withdrawing from the idea to perform phishing detection on websites and emails as isolated entities, a holistic detection approach has been created.
- By approaching emails and the websites in that email with a more holistic view, several new features have been introduced. A quantitative analysis of these features on both phishing and benign emails has resulted in observed behavioural differences between phishing and benign emails.
- A phishing detection system with new features showing detection capabilities very close to the capabilities of state-of-art systems has been introduced.
- A mathematical procedure has been introduced to make a phishing detection model which performs a phishing detection on one website applicable to perform phishing detection on a whole email message, including both benign and phishing websites.
- A holistic phishing detection system has been introduced. Using the holistic approach mentioned before, phishing detection systems performing phishing detection anywhere in the phishing attack chain can be combined. This enables detection systems for email and detection systems for websites to work together to form a single classification. This single classifier then performs better than any individual model, which makes it better than the state-of-the-art systems currently available.

1.4. Thesis outline

The next chapter is used to set some definitions clear for the rest of the research, as well as provide necessary background on machine learning, validation methods and email authentication protocols. Also, this chapter is used to dive into Phishpedia [35] a system used in this research. In Chapter 3, an overview of related work is created to give the reader an insight into recent development on topics investigated in this research. Then, in Chapter 4, the phishing attack chain is made specific. Chapter 5 introduces the datasets which are shared between different chapters. Some chapters also introduce their own datasets which are only used in that chapter. Section 2.6 introduces analysis techniques which will be used in Chapter 6 and Chapter 7 to introduce two different phishing detection systems. These two chapters start with the methodology used to perform the research. Then, results are presented for the tests which were run. These results will also hold analysis and insights on the obtained results. Finally, in Chapter 8, a discussion on the limitation of the research, suggestions for future work and conclusions will be stated.

2

Background

In the introductory chapter, the definition of credential phishing which will be used in this thesis has been defined. In this chapter, more definitions and general concepts used during the research are handled in-depth.

2.1. Brand impersonation in email

Impersonation is the act of sending an email with the name of someone, while the actual email is not sent by that person [3]. In brand impersonation, an attacker sends an email as if that email is originating from a brand, while in fact the email has not been sent on behalf of that brand. An example of an email with brand impersonation can be seen in Figure 6.9.

2.2. Machine learning

In machine learning, an algorithm is trained to improve. This training happens by providing the algorithm with sufficient data. A widely used task for such an algorithm is to be able to classify a data sample. The algorithm does this by comparing a new data sample (or its features representation) with the data (or their feature representations) it has been trained on. If the data sample is sufficiently similar, the algorithm will give the new data sample the same label as similar data samples or will group the data samples. During this research, the machine learning models will be charged with the task to distinguish phishing emails from benign emails. In this research, two types of machine learning models will be used; classification - and regression models. A classification model returns a class label for each sample, in this research that will be a ”1” for phishing and a ”0” for benign emails. A regression model provides a regression score. In this case a value between 0 and 1. The closer the regression is to 1, the more certain the model is that the email is phishing.

2.2.1. Features

The machine learning model must be able to compare data samples. For example, two emails should be compared where one email is phishing and the other one is a benign email. The email in this case is the ”raw” data for the model. The model will not be able to categorise the two emails by only looking at the raw, unprocessed emails. Simply comparing the content of the emails will (most likely) not result in a decent classification algorithm. To help the model process the raw data, features can be extracted from the raw data. Such a feature can interpret information from the raw data, and present the interpreted data to the machine learning model in a single value or variable. This makes it easier for the machine learning model to distinguish the raw data samples from each other. One example of a feature would be the type of language an email has been written in or the number of words used in the email.

2.2.2. Ensemble classifiers

In machine learning, one can combine several classification algorithms to perform one classification task. The combination of these algorithms forms a new classifier which performs better than any of

the individual classifiers. Such a combined classifier is called an ensemble classifier. There are several sorts of ensemble techniques, for example, bagging and boosting [45].

During bagging, multiple sequential iterations are done called trials. Before bagging, a training set is created from a dataset. The size of the training set is the same as the original dataset, though, some samples might have been left out, while other samples appear multiple times in the training set. The bagging classifier creates multiple classifiers with different training sets. When the system is being tested or used, every classifier is given a sample x to classify. The final classification of the whole bagging classifier is the class which has the most votes.

Boosting is a technique where several iterations of classifiers are trained sequentially. The booster holds a weight for every sample in the dataset. During training, it will update these weights based on the performance of classifiers classifying that sample correct or wrong. The higher the weight, the more focus the classifier will put to classify that sample correctly. On the first iteration, a boosting algorithm assigns a weight to every sample in the training set, where

$$w_x^1 = 1/N$$

with w_x^1 being the weight of sample x on iteration 1. On every iteration, a classifier will be constructed. based on w^t. The error of the classifier is the sum of the weights of the instances that the classifier misclassifies. If that error ever becomes larger than 0.5, the boosting algorithm will go back to the configuration of one iteration before: t-1. If the error is smaller than 0.5, the weights of the samples which were correctly classified will be updated. This will be done by multiplying it with: $\beta^t = \epsilon^t(1-\epsilon^t)$, where ϵ is the error. After this, all weights will be normalised. Thus, correctly classified samples will gain a lower weight, while misclassified samples will gain a higher weight for the next iteration. The final classification of the booster classifier will be by summing the classifiers of all iterations, with different vote weights. The vote weight is determined by: $log(1/\beta^t)$. Thus, better performing classifiers have more influence on the final prediction. [45]

2.2.3. Meta-learning

Meta-learning is a technique where a machine learning model can be trained on the output of other machine learning models. This learning-to-learn technique can be used to create multiple machine learning models which can be combined, without the need for basic voting or averaging techniques. For one data sample, several machine learning models can be constructed to classify the sample based on different aspects, but still, work together to obtain the best results [55]. Meta-learning is some sort of bagging classifier as discussed before, but then using machine learning models aimed at different tasks, instead of machine learning models performing the same classification task on the same features or input. [10]

Assume for example someone were to train three different machine learning algorithms. The first machine learning model will be trained for object detection to classify an animal-based on an image. The second machine learning model will be trained on sound, to classify an animal-based on the noise it makes. The third machine learning model will be created to detect an animal-based on its smell. Now assume combining the three models to classify an animal in real life. The smell of certain animals might be similar, so the vote of the third machine learning model should be less than the vote of the object detecting model when determining the final classification. In order to combine the three models into a single classifier, another simple machine learning model can be trained. This meta classifier takes the predictions from the distinct machine learning models as input and produces a single classification (or regression score) as output.

2.2.4. Machine learning models

There are many different machine learning models available. For this research, three models will be used for evaluation.

Random Forest Random Forest is a machine learning algorithm used for classification and regression. In a high-level description, it creates several decision trees and combines all trees for a prediction. A decision tree is a machine learning model which can be used for classification as well as regression. A decision tree has a root node, splitting nodes and leaves (decision nodes). For every input, the tree is traversed, starting at the root node until a leaf is reached. At every splitting node, either the left or

right branch is visited, depending on the condition at that node. A decision tree (T) can be assumed to be a column vector $\mathbf{x} = (\mathbf{x}[1], ...\mathbf{x}[n])^\top$ with features to a prediction value y. Each splitting node N_i at node i is defined by a (j_i, t_i) pair with j_i being an index in 1,...,n and $t_i \in \mathbb{R}$ being a threshold. At node i, if $\mathbf{x}[j_i] \geq t_i$, the right path will be taken. Each splitting node is associated with $N_i(\mathbf{x}) = \mathbf{e}_{j_i}^\top \cdot \mathbf{x} - t_i$. $\mathbf{e}_i$ is a column vector with a 1 in position i and zero otherwise. $\mathbf{e}_i^\top$ is the transpose of $\mathbf{e}_i$. [24]

Most decision trees are trained 'offline' which means a decision tree is built when all data is available. In order to train a tree 'online', i.e. update a tree when a new sample is available, an incremental decision tree approach should be taken. [57].

The process of combining several decision trees into a single (Random Forest) classifier is called 'bagging'.
While a Random Forest is growing trees, it searches for the most important feature from a random set of features while splitting nodes. This adds randomness to the model. The bagging of trees is generally done by averaging all predictions:

$$y = \frac{1}{m} \sum_{i=1}^{m} T_i(\mathbf{x}) \tag{2.1}$$

Where T_i is tree i and m is the number of trees.

Gini Importance To measure the feature importance per feature for a Random Forest, the Gini Importance can be used[37]. The Gini Importance metric builds on the Gini Impurity. The Gini Impurity, $i(\tau)$, can be calculated at every node τ. This impurity calculates how well a potential split separates the dataset into equally large parts. If the decision node split will result in an equal split, the impurity is at its maximum. The Gini Impurity is defined as:

$$i(\tau) = 1 - p_1^2 - p_0^2 \tag{2.2}$$

For any Random Forest with 2 classes, 0 and 1. $p_k = \frac{n_k}{n}$ is the fraction of n_k samples from the dataset with a total of n samples. On this node τ, a new split could be introduced for a certain feature (θ) with a threshold (t_τ). The difference in impurity for this new split is given by:

$$\Delta i(\tau) = i(\tau) - p_l i(\tau_l) - p_r i(\tau_r) \tag{2.3}$$

Where $p_l = \frac{n_l}{n}$ is the fraction of samples for a new left node introduced by the split, and p_r is the fraction for the right node. This Δi can be calculated for every θ, for different thresholds to split on. For every node, the optimal split ($\Delta i_\tau(\tau, T)$) can be calculated for all trees T. Thus, in order to determine the importance of every feature, the optimal split is calculated for every node, per θ. This results in the Gini Importance I_G.

$$I_G(\theta) = \sum_T \sum_\tau \Delta i_\theta(\tau, T) \tag{2.4}$$

XGBoost Another way of building a good classifier is by gradient boosting[12] [5]. Gradient boosting builds on the idea of boosting, which is discussed in Section 2.2.2. A gradient boosting algorithm combines several 'weak' learners into one 'strong' learner. A weak learner is a learner performing slightly better than a regression algorithm or classifier which classifies samples randomly. Assume a labelled dataset D where $D = \{\mathbf{x}_i, y_i\}_1^N$, $\mathbf{x}_i$ is a feature vector representation of data sample i, and y is the label, as an integer, of data sample i. Assume there is a function $F(x)$, which can take any x as an input and outputs the corresponding y. Gradient boosting does an approximation to this function with F. The loss between the approximation and the exact function is given by: $L(y, F^*(x))$. By minimising this loss, gradient boosting improves its $F^*(x)$ iteratively. A gradient boosting algorithm has been built up of several weak learners, i.e. decision trees. Each tree m has a weight, ρ_m, and a model function $h_m(x)$. This model function is the function of the ensemble, for example, the decision trees. The additive approximation function of the gradient boosting algorithm is built up of a weighted sum of the functions of each tree:

$$F_m^*(\mathbf{x}) = F_{m-1}^*(\mathbf{x}) + \rho_m h_m(\mathbf{x}) \tag{2.5}$$

The function for optimisation for the first model is given by:

$$F_0^*(\mathbf{x}) = \arg\min_{\alpha} \sum_{i=1}^{N} L(y_i, \alpha) \tag{2.6}$$

$L(y_i, F(\mathbf{x}))$ is a given loss function for the model. This can for example be the Gini Impurity (Section 2.2.4). Every model after that optimises:

$$(\rho_m, h_m(\mathbf{x})) = \arg\min_{\rho,h} \sum_{i=1}^{N} L(y_i, F_{m-1}^*(\mathbf{x}_i) + \rho h(\mathbf{x}_i)) \tag{2.7}$$

One example of a good gradient boosting algorithm is XGBoost [12]. XGBoost focuses on decision trees as a weak learner. The loss function of XGBoost is slightly different from the original loss function of gradient boosting, in order to control the complexity of the decision trees. The loss function of XGBoost is given by:

$$L_{xgb} = \sum_{i=1}^{N} L(y_i, F^*(x_i)) + \sum_{m=1}^{M} \Omega(h_m) \tag{2.8}$$

$$\Omega(h) = \gamma T + \frac{1}{2}\lambda ||w||^2 \tag{2.9}$$

Where γ determines the rate at which nodes will split during the training phase of the algorithm. A higher γ will result in simpler trees. T is the number of leaves in a tree and w is the score at the leaves, which will be the output of that specific model.

Support Vector Machine (SVM) A Support Vector Machine is a machine learning classification algorithm. It will try to divide the data samples into different clusters by creating a hyperplane. For a two dimensional problem, this can be visualised by imagining an x-y coordinate system where all the data samples are plotted. The support vector machine will create a hyperplane which can be seen as a line in the coordinate system, dividing the samples from each other. There are two sides of the line, one side for every category. The sample will be classified according to the side of the line it has been placed in the coordinate system. The Support Vector Machine is tasked with the problem to maximize the minimum distance between any datapoint and the hyperplane. Increasing the number of dimensions in the problem makes it harder to visualise, but the solution remains the same. The SVM will create a multidimensional hyperplane separating data samples as points in a multi-dimensional coordinate system [41].

The mathematical problem to determine where to create the hyperplane is to maximize the minimum margin from any point to the hyperplane, or in a two dimensional problem, just a line[51]. This formula to maximize is given by:

$$\min_{\mathbf{w},\gamma} \frac{\|\mathbf{w}\|^2}{2} \tag{2.10}$$

While:

$$s \times (\mathbf{w}\mathbf{x}^\top + \gamma I) \geq I \tag{2.11}$$

Where $\mathbf{w}$ is a vector perpendicular to the potential hyperplane. s is the class label (0 or 1 in this thesis). $\mathbf{x}$ is the data vector. γ is the intercept of the hyperplane, and I is the identity matrix.

2.2.5. Class weight using machine learning

When building a machine learning mechanism, one should take into consideration the meaning and importance of false positives and false negatives. In a phishing detection mechanism, a false positive is an email which is flagged as phishing, but actually is benign. When such a detection system would create many false positives, this would mean that benign emails will be flagged and blocked by the system. On the other side, false negatives would result in phishing emails being accepted in the mailbox. The model can be biased into avoiding either of the two classes by adjusting weights to the classes. When

the positive class is assigned more weight, false negatives will reduce. A higher weight for the negative class results in fewer false positives. When designing a machine learning model, one could penalise the misclassification of samples from a certain class more than samples from another class. This will make the model predict the lesser penalised class easier than the opposite. This penalisation can be done by providing the machine learning model with misclassification weights for any of the classes in the dataset.

2.3. Validation methods

In order to correctly compare different algorithms, measures of performance should be standardised [23]. A widely used basis for metrics in binary classification tasks is to define a sample to be in one of the four categories of True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN). True Positives are the number of samples from the 'positive' class a model has predicted correctly. True Negatives are the number of samples from the 'negative' class a model has predicted correctly. False Positives are the number of samples from the 'negative' class a model has predicted as being from the positive class, and False Negatives are the number of samples from the 'positive' class the model has predicted as being from the negative class. To give an example, assume a model which tries to perform the classification task of classifying an email as a phishing email. Whenever the model is given an email sample, it can output either a '1' when the model predicts the email is phishing, or a '0' when the email is not phishing. The performance of the model will now be validated on a dataset. The dataset consists of both phishing emails, and non-phishing emails. All emails are thus labelled with ground truth. Whenever a model defines a phishing email as phishing, this is a true positive. A non-phishing email labelled correctly is a true negative. A non-phishing email labelled as a phishing email is a false positive. A phishing email labelled as a non-phishing email is a false negative.

With this basis, several metrics can be calculated.

$$Precision = \frac{TP}{TP + FP} \tag{2.12}$$

$$Recall(TPR) = \frac{TP}{TP + FN} \tag{2.13}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.14}$$

$$False\_positive\_ratio(FPR) = \frac{FP}{TN + FP} \tag{2.15}$$

2.3.1. Precision-recall curve

As discussed before, precision is the number of true positives over all the samples the model has labelled positive. Recall is the proportion of the number of true positives, over all actually positive samples. [48] When a precision-recall curve for a model is created, the model will be run on a dataset. If the model is tasked with the classification of the datasets, this model might predict a class directly or predict the probability that a sample belongs to a certain class. This probability is a regression score. In a binary classification problem, one could say that the model predicts a sample to belong to the positive class if the regression score is 0.5 or higher. Each sample with a regression score lower than 0.5 would be classified as the negative class. Though, this 0.5 is a threshold which can be adjusted. By raising this threshold to 1, the number of false positives will be reduced, since the model will be less likely to label a sample to the positive class. Lowering the threshold will reduce the number of false positives. The performance of the model can be evaluated for different thresholds. So for different thresholds, the precision and recall can be calculated. The precision and recall points can be visualised in a coordinate system with precision on the y-axis and recall on the x-axis. A precision-recall curve is created by drawing a line through each of the precision-recall points for a model with varying thresholds. In this thesis, the focus will be on reducing false positives, therefore, the precision-recall curves will be created for a threshold of 0.5 and higher. Assume for example a regression model, which will be given 10 email samples. On all 10 samples, the model will create a prediction score on how likely the sample is phishing. A threshold is set at 0.5, so when the model returns a prediction score of ≥ 0.5, the email is predicted as phishing. When the prediction score is < 0.5, the email is predicted as non-phishing. By comparing the predicted class with the label given to the sample in the dataset, it can be determined

per sample whether the model is correct or not. For this threshold, the precision and recall can be calculated. Now, the threshold which was previously set to 0.5, is set to .55 and the test is run again. Again a precision and recall value can be calculated. By plotting every single precision-recall point for a threshold, a graph can be established. The area under this graph is a single value, which can be used to compare performance between different (machine learning) prediction models.

2.3.2. Validation methods for object detection

The validation methods for object detection are slightly different compared to general validation methods. An object detection mechanism is loaded with the task to detect objects in a visualisation. The validation of such a mechanism can be done by providing the mechanism with an image containing a certain object. The mechanism will try to find the coordinates of that object. The actual coordinates of the object are compared to the predicted coordinates by the system.

Validating predicted samples Assume an object detection model which is constructed to perform the task of detecting logos in an image. Upon being given an image, the model will produce a bounding box in that image where it predicts a logo is depicted. This is called the prediction. The image has been labelled, so it is known where the box should be created to encapsulate the logo. This is the ground truth. Using the prediction and the ground truth, we can calculate the intersection over union.

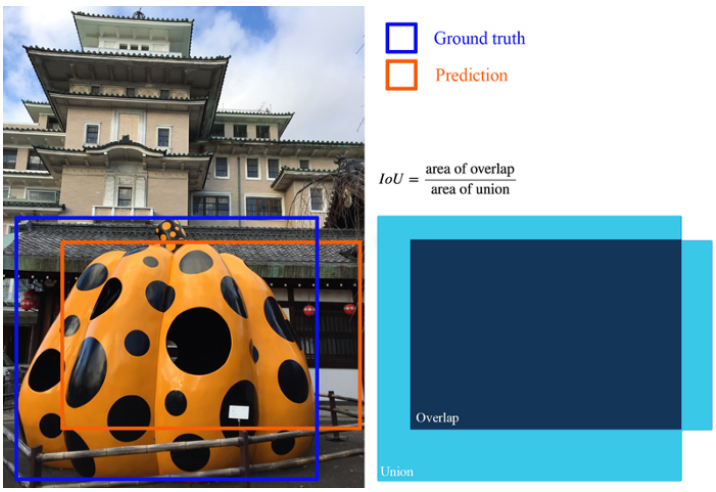


Figure 2.1: Intersection over Union is the area of overlap between two shapes, divided by the area of union between these two shapes [27]

Intersection over Union The IoU is the proportion of overlapping area over the total area of union.

$$IoU = \frac{Area\_of\_Overlap}{Area\_of\_Union}$$

This intersection over union is used to determine whether the model is actually correct in its prediction. We can define a minimum intersection of union which the model should have to actually predict a sample correctly. This is called the IoU threshold. For example, if the threshold is 0.7 IoU, and the model does a prediction which has 0.5 IoU, the model is wrong.

Performance measures for Object Detection Systems This IoU can then be used to define true positives, false positives and false negatives. The task at hand for object detection is that a model should be able to identify an object by drawing a box around the object. This will be compared with the ground truth, i.e. the image with the labelled boxes as discussed before.

- **True Positive** - The model predicts a box containing a logo, which has an IoU with a ground truth box higher than the threshold.

- **False Positive** - The model predicts a logo box with an IoU lower than the thresholds compared to the ground truth. This IoU could also be 0, meaning the model and the ground truth have no overlap.
- **False Negative** - The model does not predict any logo, while the ground truth does contain a logo.
- **True Negative** - The model does not predict any logo, and neither does the ground truth.

[44] [4]

Do note that every image can hold several boxes in both the model and the ground truth. This means that one image might result in several true positives, false positives and false negatives. Though, a true negative here is not possible, since then every pixel which is not in a box, could be calculated as a true negative. Using this, the precision-recall curve can be established, which in its turn can be used to calculate the Average Precision (Section 2.3.3).

Recall for object detection systems Remember that in recall for general classification, false negatives were used in the equation. Though due to the nature of object detection, false negatives are not used when calculating recall [43]. Instead, Recall is calculated as:

$$Recall = \frac{TP}{P} \tag{2.16}$$

Where P is the total number of ground-truth positives. In object detection, a 'positive' is a labelled object in the sample.

2.3.3. Average Precision for Object detection systems

The average precision is a metric used in measuring the accuracy of object detection systems. The Average Precision (AP) can be calculated by calculating the area under the precision-recall curve from Section 2.3.1. A mathematical formula for this is given by:

$$AP = \sum_{n} (R_n - R_{n-1}) * P_n \tag{2.17}$$

Where R_n is the recall for threshold n, and P_n is the precision for that corresponding threshold. The start ($n = 0$) recall is 0. $n = 1$ is the threshold with the lowest recall. Consecutive points will have an increasing recall.

2.3.4. k-fold cross-validation

The validation of a machine learning model is done by splitting the dataset into a training- and testing part. First, the training part is used to train the machine learning model. Then, the testing part is used to see how well the model performs. It is important that the samples from the training and testing set do not overlap, i.e. the samples from the test set are not used to train the model. By splitting the dataset into these two parts, one might note that some very distinctive samples might end up either in the train-, or test-set. The model then will never have the possibility to train on these samples, reducing the overall performance, just by an unlucky split. In order to avoid such a case, k-fold cross-validation is used. This technique requires running the cycle of training and testing several (k) times. First, the entire dataset is split into k-parts. One of these parts will function as the test part, while the others will compose the train part. The model is again trained on the train part and tested with the test part of the dataset. In the next iteration, the model will be trained and tested again, but with another test part. The part which has been used for testing before will now be used to train on, and one of the parts which were previously used for training will now be used as a test part. This happens k-times until every part of the dataset has been used for testing once. The final results of the cross-validation are comprised of the test scores from every iteration.

Stratified To ensure that the ratio between the different classes under investigation is the same in every test set, stratified k-fold can be used. Stratified k-fold will take, during the splitting process of the dataset, into account that every k-part holds the same ratio of every class of samples. During this research, this thus means that the ratio between phishing and benign emails is the same for every k-part.

2.4. Email authentication protocols

To prevent email spoofing and increase overall email security, three email authentication protocols can be integrated for an email domain. Two of these, DMARC and SPF, will be investigated during this research.

2.4.1. Sender Policy Framework (SPF)

Using the sender policy framework, domain owners can publish from which IP addresses that domain sends its emails [39]. The owner of the domain publishes an SPF DNS record, stating a valid IP address (blocks). This record also states what should happen to emails which do not originate from the valid IP addresses listed. E.g. emails from non-valid IP addresses could be allowed, dropped or marked as spam. An example of such an SPF record is:

“v=spf1 ip4:129.6.100.200 ip6:2610:20:6005:100::20 -all”

The first part, **v=spf1** indicates that this record is an SPF record. Next, two IP addresses are stated. If an email from this domain originates from either of these, the email is deemed to be benign. The last part **-all** indicates that email from this domain originating from any other IP address will be dropped. When an email is received, the SPF record for the domain is queried. The IP address from which the email originates is checked to the record. If the SPF record states that the email should be dropped, the ’SPF check’ fails.

2.4.2. DomainKeys Identified Mail (DKIM)

The DKIM protocol provides a framework to sign headers and content of an email message. This allows receiving parties to verify that the content of the email or the headers has not been changed between the sending of the message, and the message being received [39].

A DKIM record in DNS is given by:

“v=DKIM1 ; p=<encoded public key>

Where **v=DKIM1** states it is a DKIM record and **p=** is the public key used to verify the message.

2.4.3. Domain-based Message Authentication, Reporting and Conformance (DMARC)

A domain owner can configure the DMARC protocol to its domain. This protocol allows the receiver to authenticate email messages and send feedback on potentially failing emails received from the domain. Essentially, the DMARC protocol verifies both SPF and DMARC to an email. A DMARC record is published in DNS for a certain domain:

“v=DMARC1; adkim=r; aspf=s; p=none; pct=100; rf=afrf; ri=86400; ruf=mailto:forensics.example.com;”

v=DMARC1 states that this record is a DMARC record. The next, **adkim** states how strict the DKIM record should be evaluated. **aspf** does the same for SPF. **p** and **pct** state when to apply the DMARC policy, and **rf**, **ri** and **ruf** handle the feedback settings FOR the receiving party. Since this policy is published in DNS and is not being sent with an email, the DMARC to an email can be checked independently from the email received. Therefore, with DMARC configured correctly, this protocol prevents spoofing of email domains.

2.5. Phishpedia

Lin et al. [35] proposed a system, Phishpedia, to identify phishing web pages from visual aspects. Phishpedia tries to detect logos on web pages. Whenever the logo of a company is detected on a web page while the web page does not belong to that company, the web page is deemed to be phishing.

2.5.1. Detection of logos

For the detection of logos, Phishpedia uses the Detectron2 library [59]. The model created can be seen in Figure 2.2.

The detection of a logo on a web page is done by a deep learning object detection model. The object detection model consists of three parts.

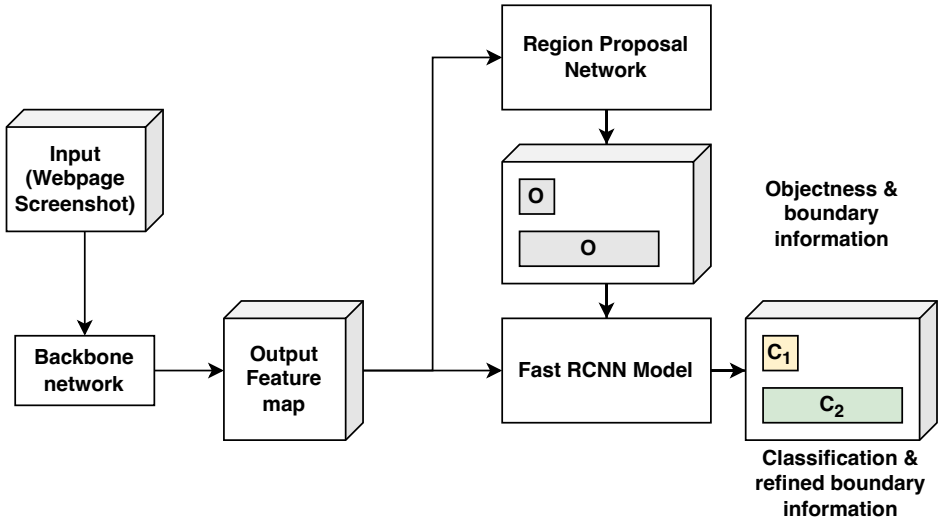


Figure 2.2: Phishpedia object detection model [35]

1. Backbone network (Phishpedia uses Resnet50)
2. Region Proposal Network (RPN)
3. Fast-RCNN model

The object detection model takes a screenshot from a web page as input. This image is translated into a feature map by the backbone network. This feature map is the input for both the RPN and Fast-RCNN models.

Region Proposal Network The Region Proposal Network [46] works on a feature map, for example, the output of a convolutional neural network. In this case, the RPN takes the output of the Resnet50 backbone network. The RPN will analyse the feature map and based on that feature map, it will produce 'proposals' for objects inside that map. These proposals will consist of a rectangular box which might contain an object, together with an 'objectness' score. This objectness score predicts the class of the object.

Fast-RCNN The Fast-RCNN in its turn will use the output of the RPN and the initial feature map to determine the likelihood of the objects in the boxes being a logo. Also, it refines the shape and size of each object. The outcome of the logo detection system is thus a set of candidate logos, including a confidence score. The candidate logos are represented by four coordinates per logo. The four coordinates span a box around the area where the candidate logo should be in. The coordinates are coordinates inside the image (in this thesis a screenshot). The confidence score is a regression score of how sure the model is that there is a logo inside the box predicted.

The Fast-RCNN model has been trained based on the Detectron2 framework [35]. The dataset used to train the model consists of screenshots of web pages, both phishing and benign, containing all different kinds of logos. The phishing part of the dataset has been gathered by the researchers from OpenPhish.

2.5.2. Brand identification

Whenever a logo has been detected by the previously described system, the brand identification system will try to identify the logo. This is done by a Siamese neural network model. A Siamese Neural Network model starts by transforming an image into a representative vector. In this thesis that image is a (candidate) logo. After this, the comparison between two images is calculated by the similarity between the two representative vectors of the images.

The Siamese model consists of a Resnetv2 network. Resnetv2 is a neural network which captures features of logo images. The output of the Resnetv2 is a feature map. The Resnetv2 is called the backbone network. During the training phase, the captured features of the logos will be the input for a fully

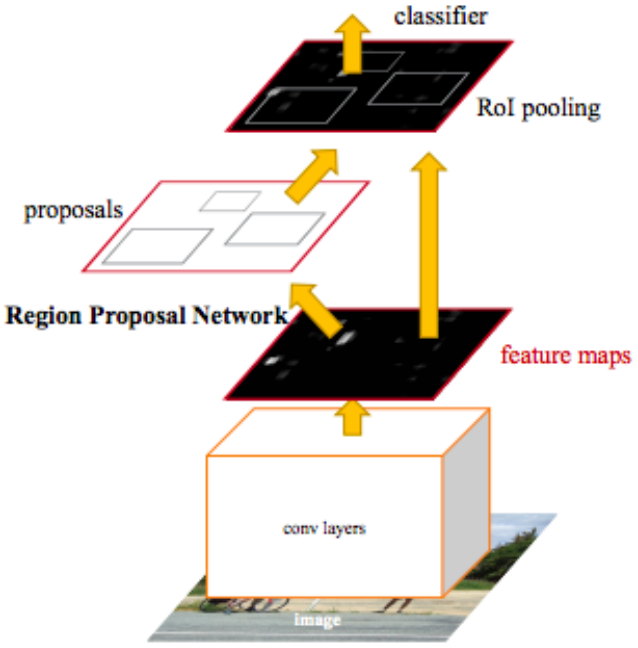


Figure 2.3: Region Proposal Network [46]

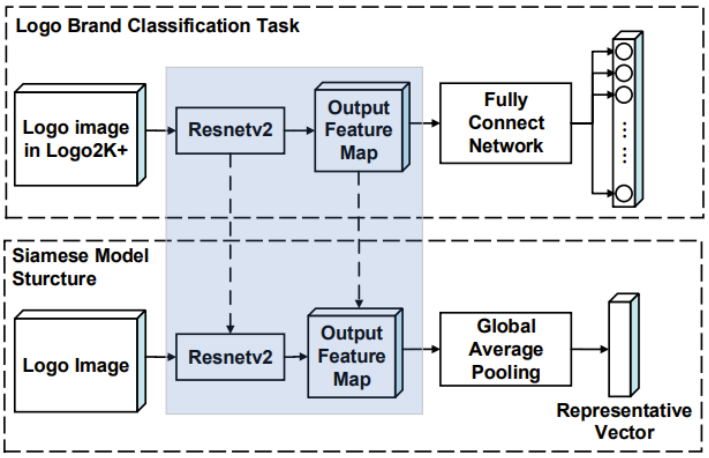


Figure 2.4: Siamese Model from [35]

connected network of one layer. This layer is trained with logos to be able to recognise the logos of 2341 brands (Logo2K+ dataset). This system can be seen in the upper half of Figure 2.4.

After the training phase, the backbone network is ready to serve in the Siamese model. In the Siamese model, a logo is fed to the previously trained backbone network. This network is followed by a global average pooling layer which produces a representative vector, representing the logo image. Phishpedia holds a set of over 200 brands, of which it has multiple logos. This is since brand logos might slightly vary from time to time, see Figure 2.5.

When the model is trying to identify a logo, say logo **X**, Phishpedia will create a representation vector for logo **X**. Then, it will create vector representations of all the other logos it already has in the same way. Phishpedia will then compare the vector representation of logo **X** to the representations of all the logos of all brands. The comparison is done based on cosine similarity. Though, if Phishpedia is investigating a logo of a brand which is not in the set used by Phishpedia, this logo will not be identifiable.

After Phishpedia has calculated the cosine similarity between logo **X** and all other logos, the four logos with the highest cosine similarity will be further investigated. Whenever a cosine similarity is

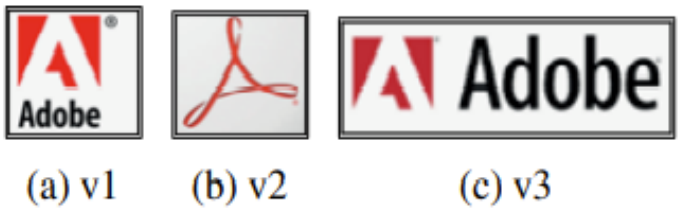


Figure 2.5: Logo Variants for Adobe [35]

found above a certain threshold (default to .835), logo **X** is deemed to be of the same brand as the logo to which it matches to. The web page is then considered to be impersonating that brand. This means that the web page is either the website of that brand, or a phishing page.

2.5.3. Domain validation

Phishpedia holds a small set of domains per brand. After a brand has been identified on a web page, the domain of the page is validated. If the domain is not in the set of domains which Phishpedia relates to that brand, the web page is considered a phishing page. Though, this check is not extensive enough. Companies can hold many domains. E.g.: Phishpedia would consider 'dhlparcel' as phishing since the only valid domains for DHL would be: dhl.

2.6. Feature Analysis

In the holistic approach, two modules (Chapter 6 and Section 7.1.2) are used which extract features from emails. In section 2.2.1, the inner workings of a machine learning model built from features have been discussed. The analysis of feature-based models is very important. When training a machine learning algorithm on manually extracted features on a fixed dataset, there is a common pitfall. When doing this, one or more of the selected features might be leaking the class label. As a result, by solely handling those features, the algorithm can predict data samples very accurately. To prevent this pitfall, three analysis steps are taken.

1. General statistics
2. Feature ranking
3. Impact on performance for removing features

The first analysis is conducted solely on the dataset, without the use of any machine learning algorithm. The feature ranking will be performed on a Random Forest classifier. This classifier will be trained on the combination of email datasets (Chapter 5), containing both phishing and benign email. The third analysis will also be done using a Random Forest classifier, trained on the same dataset as the feature ranking. During this analysis, the impact on the performance of removing a feature is investigated.

2.6.1. General statistics

The purpose of the general statistics analysis is to get a feeling for the features at hand. Even though the other (more mathematical) methods of the feature analysis might not detect any importance, plain reason could find biases in data. Interesting insights could be that the value of one feature is always the same for one category, but different over the two categories. This way, solely this feature could perfectly predict a phishing mail. Per feature and separated for both categories, the following metrics will be investigated:

1. Number of occurrences of a zero in the whole dataset
2. Feature is zero for all samples

3. The frequency of the most occurring element
4. The most occurring element is the same for both phishing and benign emails

2.6.2. Feature ranking

During the feature ranking analysis, the features will be ranked on importance by using a Random Forest. The importance of features to a Random Forest classifier can be calculated using the Gini Importance, as was described in Section 2.2.4.

2.6.3. Impact on the performance of removing features

Next to the ranking of the feature importance from the previous section, it can be interesting to quantify what the impact of a feature is on the performance of the model. For the modules for which features were engineered, experiments were run where the features were removed from the model to see the effect on the performance. Even though In those experiments, a model has been initialised in the ordinary way, trained on the dataset containing all features. In the next iteration, the best performing feature has been removed from all samples in the dataset. The model has been retrained and validated. This has been done iteratively until only one feature was left. This process has been described in Algorithm 1.

Algorithm 1 Calculate the impact of feature removal on the performance of a machine learning model

$model \leftarrow get\_initialised\_model()$
$N \leftarrow number\_of\_features\_in\_model$
$feature\_set \leftarrow all\_features$
while $N \geq 1$ **do**
  $performance \leftarrow calculate\_performance(model)$
  $most\_important\_feature \leftarrow determine\_most\_important\_feature(feature\_set)$
  $feature\_set \leftarrow feature\_set - most\_important\_feature$
  $model \leftarrow model.remove\_feature(most\_important\_feature)$
  $model.reset()$
  $N \leftarrow N - 1$
end while

3

Related work

This chapter contains an overview of literature on the process of a phishing attack, as well as literature on several phishing detection systems. The studies mentioned in this chapter are just a subset of the many studies done on these topics. Only studies with a close relation to this research have been discussed.

3.1. Phishing attack process

From an attacker's point of view, a phishing attack consists of several phases. In this section, two interesting views will be mentioned, serving as a source in answering the first research question stated in Section 1.2.

3.1.1. Phishing phases in a general phishing attack

Alkhalil et al. [2] propose anatomy of phishing consisting of four main phases. This anatomy holds for a phishing attack in general and is thus not focused on the phishing attack considered in this paper. Those 4 phases are made up of several sub-parts which define the stage but are not necessarily actions succeeding each other.
The graph can be seen in Figure 3.1.

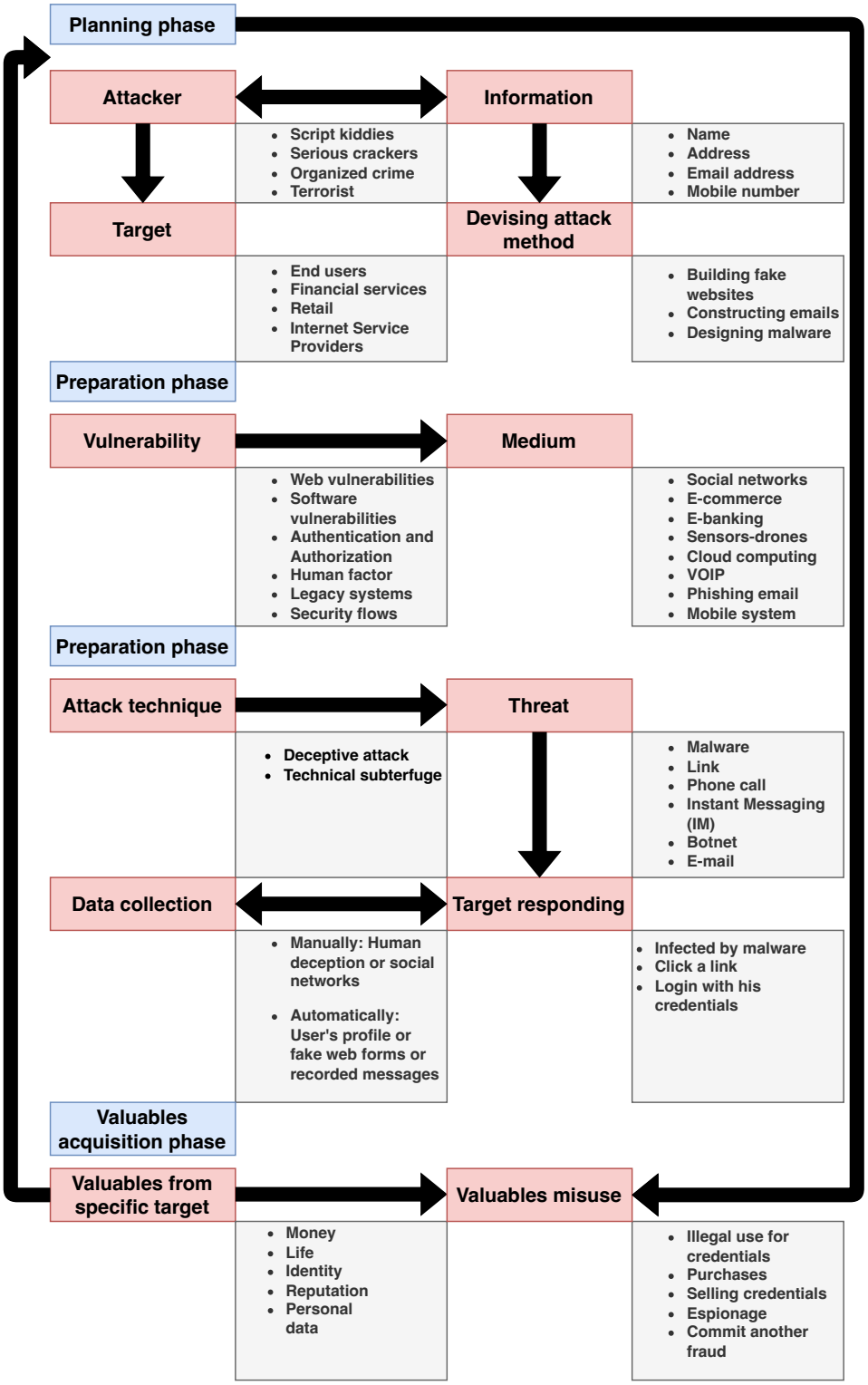


Figure 3.1: Phishing phases by [2]

1. Planning phase
The first phase of an attack is the planning phase. During this phase, an attacker determines a target or target group, gathers information and devises an attack method.
2. Preparation phase
The attacker tries to identify vulnerabilities. Those can range from technical such as zero-days to human factor vulnerabilities. When vulnerabilities are found or chosen by the attacker, the attacker chooses a

medium over which to perform the attack or deliver the threat.

3. Attack conduction phase

In this phase, attack techniques are used to deliver the threat to the victim. This can be either a deceptive attack or a technical subterfuge. After the use of the attack technique to deliver the threat, the target responding is an important part of phase 3. After the response of the target, the data collection is also part of the attack conducting phase.

4. Valuables acquisition phase

The last phase is the valuables acquisition phase. In this phase, valuables from the victim can be stolen, or any acquired position might be abused.

3.1.2. Phishing stages of a phishing website

Oest et al.[42] define high-level phishing stages of a phishing attack in seven stages (Figure 3.2). These are different from the paper discussed before since these stages include a stage for ”mitigation”. This stage overlaps with ”Victims Visit Phishing Website” and ”Monetization”.

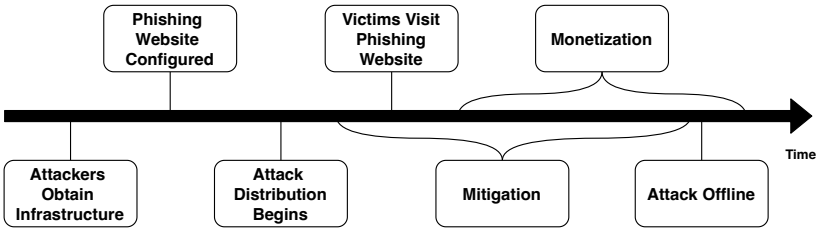


Figure 3.2: Phishing stages by [42]

A Attackers obtain the infrastructure to conduct an attack.

B Phishing website is configured

C The attacker delivers the URL to the phishing website to the target

D The victim visits the phishing website

E The mitigation phase starts as soon as the URL has been delivered to the target.

F The monetization can start as soon as the target has visited the phishing web page.

G Phishing website has been debunked and either taken offline or blocked.

3.2. Phishing detection

The aim of a phisher is to design its phishing attack as discrete as possible. The phisher tries to circumvent any phishing detection and totally blends in with benign emails. Defenders against phishing attacks, on the other hand, try to differentiate malicious emails from benign ones. Though, the defender also takes into account the availability of benign emails. When benign emails are not delivered due to phishing suspicion, the defender fails. This continuous arms race has been going on for a long time. Two important trends in the detection of phishing attacks can be distinguished: phishing detection on emails and phishing detection on web pages.

3.2.1. Phishing email detection

Detection of phishing emails is mainly focused on the structural or contextual elements of emails. Though, back in 2012, Stringhini et al. [50] investigated a way to perform phishing detection on how a message is sent. Their paper introduces a way to detect whether an email has been sent by a bot by identifying an SMTP ”dialect”. SMTP is the protocol which defines the delivery of an email. It works by sending a command followed by optional arguments. The server replies to messages from the client with a certain status code. Since servers typically also accept commands which are not necessarily part of the protocol, a sender might implement the protocol in a slightly wrong way. This might be used to identify an individual implementation. The identification happens by means of a state machine.

Smadi et al. [49] created a neural network to detect phishing emails. They trained a system with high accuracy, TPR, TNR: 98.63%, 99.07%, 98.19%. The model created makes use of different categories of features: features from email headers, URLs, email HTML content and main text. What is unique in their approach is that, first, their system marks emails as suspicious and then they let an expert look at the email. Several features from this system have been used in the feature engineering in Section 7.1.2.

After Smadi et al, Fang et al. [22] performed another research on phishing email detection. They called their phishing email detection mechanism THEMIS. THEMIS is built from Recurrent Convolutional Neural Networks (RCNN) with multilevel embedding and an attention mechanism. The paper first introduces multilevel embedding where both characters and words are being analysed. Then character level and word level vectors are created. After doing this, the paper combines the multilevel embedding with an RCNN with an attention mechanism. The whole model works on the email in separate parts being: email headers, email body, character level and word level. Research on phishing detection can be split into three different categories: Semantics, syntax and contextual information. Very interesting research came in 2021 by Lee et al. [34]. The authors created a system which consists of 3 different machine learning and deep learning mechanisms to detect phishing emails. Their system is divided into three parts. 1. Text modelling, 2. structure modelling and 3. URL modelling. The structure modelling module extracts text from both the headers and HTML part of an email. Features for this module are selected by hand and serve as input for both a supervised learning algorithm, as well as a deep learning algorithm. Several of the features created by Lee et al. were used in the creation of the email structure module as described in Section 7.1.2.

3.2.2. Phishing web page detection

The detection of phishing web pages is being investigated extensively. In 2011, Cantina+ from Xiang et al. [60] was introduced. It extracts features from a web page, on which they build a machine learning algorithm to perform detection. The features used in their research were URL-based. HTML-based and Web-Based. The prediction model was state of the art with excellent performance and their paper influenced the research community. Oest et al. [42] defined the ”Golden Hour” framework to identify phishing sites using Web requests, URLs and content analysis. The Golden Hour framework consists of 5 stages. In the first stage, web requests are analysed. Attributes of interest are, IP address, timestamp, user agent, session identifier, referring URL, and main page URL. In step 2 the URLs are held against any whitelists to eliminate benign requests. In the next step, parts of the URL are compared with recent lists of known malicious URLs. In step 4, the events are analysed. Events marked as phishing are immediately taken care of. In step 5, new information is held against the old detection to be able to identify the phishing at a later stage if needed.

Where Xiang et al. and Oest et al. used URL based detection inside their system, Sahingoz et al. [47] performed a phishing detection system solely on the text of URLs, building a Random Forest with natural language processing features. Wei et al. [56] also designed a system to detect phishing on URLs. They analyse phishing URLs to distinguish them from benign URLs. The paper has a dictionary of 70 symbols. By putting the symbols as rows and the URL as a column, they create a picture with pixels at the presence of a character. The architecture of their module consists of: 1. A system which creates a bitmap from URLs, 2. Deep learning model.
Also in 2019, Ding et al. [18] tried to detect phishing pages in three steps. The first step uses a search engine based technique. They use the title tag content of the potential phishing site as keywords in a search query. If the website is within the top 10 search results, the website is deemed not to be phishing. As a second step, the URL is analysed to detect URL obfuscation techniques. This is done by splitting the URL and detect whether common names are used as identification names. For example: pay.pal would try to disguise as paypal. The third step is building a logistic regression classifier. They extract features from the site from the DNS, Whois, similarity with phishing vocabulary, lexical features and HTML. DNS doubt is proposed to check whether the DNS is used by many phishing sites. If so, suspicion is raised.

A new approach was introduced by Drury et al. [19]. They investigated whether there are distinctions between SSL certificates for benign- and phishing domains. They found no significant difference, however.

Back in 2016, Tan et al. [52] proposed a system to identify whether a web page is compared with the company website of which it appears to be. In the first stage, input field is detected. If there are input fields, plain text and URL are extracted. In the third stage, the system tries to identify target words

which might indicate the identity of the website. In the last module, those keywords found will be fed to a search engine to find the domain name of the true domain. The domain name found and the one under investigation are then compared. An interesting anti-phishing technique categorisation is also done: 1. Text-based detection 2. Visual based detection 3. Feature-based detection 4. Identity-based detection 5. Prevention-based technique.

Later in 2020, Bozkir et al. [7] created LogoSENSE which identifies phishing pages by recognising brands. Bozkir et al. created a system to detect company logos on websites to detect phishing sites. Though there are several limitations. They already discuss the possibilities of adversarial attacks. But also a large page size results in very long run time. One year later, Lin et al. [35] took a similar approach with a new design. They developed a deep neural network which can recognise phishing pages based on company logos being present. The system they created consists of a Siamese deep learning network which first tries to identify logos by means of object detection, and then compares that logo to one of 181 used logos. If the logo appears to be one of a large brand, the page is identified as phishing. The recall of defining a phishing page is 88.67%. Recall here is reported true positive divided by total number of phishing pages. This system has also been discussed in Section 2.5, since it is used and extended in this thesis.

The before mentioned work of Lee et al., Smadi et al. and Lin et al. ([34], [49], [35]), were adopted and extended to be used in this thesis. Already one can see that the related work can be separated in work between detection on phishing emails and webpages. This thesis differs from the related work by taking a more holistic approach to phishing detection. This difference will be discussed further in the next chapter.

4

Phishing attack chain for a credential phishing attack over email

It is important to get a good understanding of how a credential phishing attack over email is conducted. Even though no two phishing attacks are the same, the research on existing literature described in Chapter 3 has shown that phishing attacks can consist of several, generic phases. This chapter will define the phishing attack chain for credential phishing attack by email.

4.1. Knowledge gap

In Figure 3.1 and Figure 3.2, two suggestions on phases and stages in a phishing attack are depicted. The phishing stages from Oest et al. [42] can be mapped to the phases of Alkhalil et al. [2]. To prevent confusion here, we must define the difference between a stage and a phase. Stages and phases have almost the exact same meaning [17][16]. Though, the stages of Oest et al. are more specific on the type of action taken by the attacker than the phases from Alkhalil et al.. Therefore, for the sake of clarity, the word ”stages” will be used to refer to a more concrete part of an activity than ”phases”. **Phases** - high-level step in the process of a phishing attack. **Stages** - low-level step in the process of a phishing phase (might also be explained as a sub-phase). The attack stages as proposed by Oest et al. are primarily focused on stages in the lifecycle of a generic phishing website. As mentioned in Chapter 1, this thesis focuses on deceptive phishing attacks. In such an attack, a phishing email plays a huge role. Therefore, new stages must be refined, while taking the phishing attack phases from Alkhalil et al. into consideration.

4.2. Phishing attack anatomy

During a credential phishing attack, two parties are involved: the **phisher** and the **target**. As discussed in the introduction in Chapter 1, the phisher will try to trick a target into exposing credentials or other sensitive information.

As discussed in Section 1.1, a credential phishing attack over email will generally follow the same four steps. In these four steps, the phisher should create a phishing web page, host the phishing web page, craft a phishing email, send the email to the target and use the stolen credentials. In a successful attack, the target takes the following steps: open the email, visit the phishing website and fill in its credentials.

4.3. Attack phases

The phishing attack anatomy should be captured in the full phishing attack chain. When mapping the deceptive attack by email to the phases by Alkhalil et al., several stages inside the different phases can be identified, Figure 4.1. This can be also referred to as the **phishing attack chain**.

Figure 4.1: Phishing attack phases for credential phishing attack over email

4.3.1. Planning phase

During the planning phase, the phisher makes a plan for the attack to come. The phisher determines its target. How phishers choose their target can greatly vary depending on the type of attack. Spear phishing attacks, for example, are a form of deceptive credential phishing where the attack is crafted for a single target [11]. In a spear phishing attack, a target might be selected via social media or other open-source intelligence because it is part of some organisation which might be of interest to the phisher. In non-spear phishing attacks, the phisher might have multiple attacks at once. In such an attack, phishers might use lists of email addresses which are known to be valid. In a third type of attack, a phisher might have gained access to an email inbox in a legitimate organisation. It might then select any contacts of the compromised user as new targets for further phishing attacks.

The second part of the planning phase is the crafting of a phishing email. That phishing email will try to trick the target into visiting the phishing web page which will be hosted by the phisher. This can be done in many ways. A commonly used attack vector is by impersonating a brand. In this case, the phisher will craft an email which looks like it is coming from a legit source. For the third part of the planning phase, the phisher will create a website to trick the target into filling in credentials. Such a phishing website often tries to impersonate the website of a well-known brand. Phishers do not always create an entirely new phishing web page for every single attack. There are phishing kits available which can easily deploy a phishing web page [6].

4.3.2. Preparation phase

In the preparation phase, the phisher will host the phishing web page. Generally, phishing web pages are not online for very long [42], thus the phisher is keen to make its targets bite the bait as soon as possible. As a last step of the preparation phase, the phisher sends the crafted phishing email. After the email has been sent, the attack conduction phase starts. From that point, the phisher has no more control over the phishing email and has to wait for the target to bite the bait.

4.3.3. Attack conduction phase

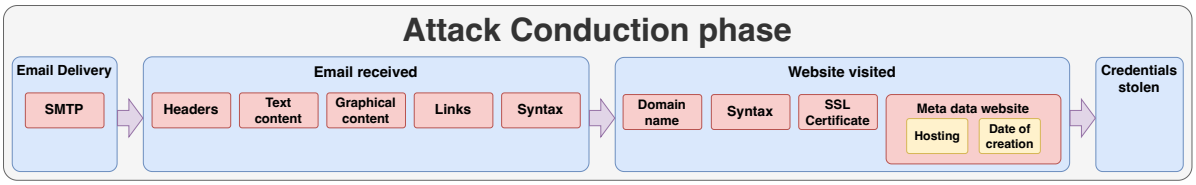


Figure 4.2: Attack conduction phase enlarged

During the attack conduction phase, the crafted email has been sent. The delivery of the email from the phisher to the target is the first stage in this phase. If the email has been delivered successfully and has not been blocked by any email security in place, the ”email received” stage is entered. In this stage, the target has received the email of the phisher where a link to a phishing web page has been placed. If the target clicks on this link, it is taken to a phishing web page. There, the target believes the web page to be something else than a phishing page, and the target fills in its credentials. The attack conduction phase has been shown in more detail in Figure 4.2

4.3.4. Valuables Acquisition phase

In the last phase, the phisher abuses the stolen credentials of the target. Depending on the stolen credentials, the abuse can be many things. Stolen login credentials for a Microsoft 365 environment, for example, can give the attacker access to company documents, email boxes and many more. Creditcard credentials could lead to the theft of financial resources. Stolen credentials for a webshop could lead

to unauthorised purchases by the phisher in name of the target, and many more different vectors are known to be taken.

4.4. Detection possibilities

Detection possibilities are very different per attack phase. During the planning phase, the phisher can operate stealthily and hardly expose any information about its bad intentions. During the preparation phase, the phisher publishes a web page, exposing the web page to the world. Detecting a phishing web page in this stage is still hard since many web pages are registered at any given moment. One approach could be to check every newly registered domain, demanding a vast amount of resources.

The two most interesting phases from a detection perspective are the attack conduction phase and the valuables acquisition phase. During the attack conduction phase, the email has been sent by the phisher. This means that the email can be analysed by the receiving party, the target. This email will also hold the link to the phishing web page, making it possible for the receiving party to analyse the site. During the valuables acquisition phase, the phisher tries to abuse the stolen credentials from the target. In recent years, a lot of detection possibilities have been opened up by analysing the behaviour of the phisher while conducting this phase. Since the phisher is not the same person as the target, the phisher might show different online behaviour compared to the original owner of the credentials. Also, it is very likely the phisher is in a different place than the target, and is using different devices than the target normally uses. Cloud monitoring analyses these and other components[1]. This has resulted in increased detection possibilities in cloud-based environments during the valuables acquisition phase.

4.5. No research which overlaps stages in the attack conduction phase

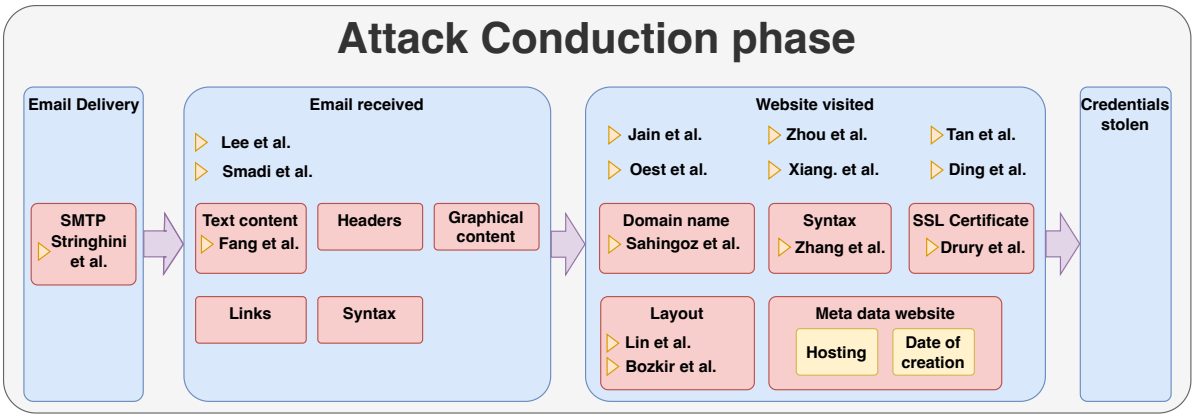


Figure 4.3: Attack phases and phishing detection mechanisms Stringhini et al.[50], Lee et al.[34], Smadi et al.[49], Fang et al.[22], Jain et al. [30], Oest et al. [42], Zhou et al. [63], Xiang et al. [60], Tan et al. [52], Ding et al. [18], Sahingoz et al. [47], Zhang et al. [62], Drury et al. [19], Lin et al. [35], Bozkir et al. [7]

Since the attack conduction phase is susceptible to detection, much research has been done in this phase. Figure 4.3 is just a subset of all phishing detection research performed. When looking at the figure, one might note that much research overlaps within a stage, but there is no research which overlaps several stages. One possible reason that no research is identified which overlaps both the email received and website visited stages, could be the fundamental difference between the two. Emails and websites often have different purposes, designs and protocols to adhere to. Performing phishing detection on either might be easier than combining the two. Though, seeing a phishing email and a phishing web page within that email as two isolated entities might result in information loss. Even though the two entities are very different, they also share informational aspects. This does not only hold for phishing emails, but for benign emails and domains as well. For example, the domain of a web page and an email domain might be the same. But some more non-trivial aspects could also be shared between an email and domains linked within that email. So could the email and a website include the same logo or share a certain writing style. By identifying many of the possible aspects shared, information from email and

website can be used in phishing detection which has never been used before. This thesis will suggest a model which performs phishing detection by combining the email- and website stages in one detection system.

4.6. Combining detection models in a holistic detection system

Next to the insight that no phishing stages were combined in previous research, Figure 4.3 also gives insight into how vast the research on phishing detection has been. Some of this research has been done very recently and performs well. When extending the view of combining different stages from the attack conduction phase we argue that the detection of different stages should be used in classification. With this approach, phishing detection systems created by different researchers can be combined into a single phishing detection system. This system will take into account multiple stages in the attack chain. Such a single classifier might very well outperform the individual phishing detection systems which work on isolated parts of the phishing attack chain. This approach will be investigated in the chapters to follow.

5 Datasets

5.1. Email datasets used for Fusion module and Holistic System Validation

During this research, a lot of effort has been put into gathering the right datasets. A blend of publicly available and private data was used. The datasets described in this section are the datasets holding phishing emails containing phishing links. These datasets are used during the construction of both Chapter 6 and Chapter 7. They are thus shared between other chapters. It is very important to note, that the results found in this research, are based on these datasets. Even though the datasets were selected with great care, it is likely that these datasets do not span the entire phishing landscape, meaning that there are unique phishing samples which are not represented by any phishing examples used to perform the research. Expanding the methodology as performed in this research to other datasets or phishing samples might thus result in different results. Table 5.1 shows the datasets used in this thesis. Columns 4 to 7 show which dataset is used during the validation of which method.

Table 5.1: Email datasets used for validation of different systems

Dataset	Number of phishing emails	Number of benign emails	Used for Brand recognition	Used for Fusion module	Used for Structure module	Used for Holistic system
Nazario	1555	0	TRUE	TRUE	TRUE	TRUE
EYE	178	591	FALSE	TRUE	TRUE	TRUE
Personal	0	1821	TRUE	TRUE	TRUE	TRUE
Total	**1733**	**2412**				

5.1.1. Nazario

From the Nazario phishing dataset [38], only the phishing emails after 2015 are used. From 2015 up to 2021, the Nazario dataset holds 1915 phishing emails. Though, due to the malformation of some of these emails, not every email is usable for each of the modules proposed in this research. Therefore, **1555** emails are used from the Nazario dataset. Those are phishing emails collected by Jose Nazario. The phishing emails contain an excellent layout and original links.

5.1.2. External

For this research, we reached out to many companies to ask whether they would be able to provide phishing emails for this research. It turns out that some organisations have a dedicated email address

where employees can report phishing emails. Several organisations were willing to cooperate on this research. Some organisations also pledged to collaborate on the research. Though the conversations with the companies started in the early phase of this thesis, it was not managed to set up the correct legal contract or infrastructure in time to actually use the data. Only Eye Security[21] managed to provide phishing data in time. Though, all preparations and infrastructure for the use of the data from external companies were finished and ready for use. During the talks with the external companies, it was clear they wanted the privacy of their employees to be preserved. In order for them to collaborate on this research, it was needed to anonymize their data. The methodology for this has been described in the next subsection.

Data exfiltration Organisations are, to put it mildly, not keen to open up their mailboxes, even if the emails are only phishing. They are also not always legally allowed to do so. The reluctance is often primarily focused on internal email addresses or sensitive information. Though, these standalone information points might not be needed to perform phishing detection on. As will be laid out indeed in Chapter 6 and Chapter 7, individual email addresses or other sensitive information is not evaluated during the detection process. To convince third parties to collaborate on this thesis and provide phishing emails, a straightforward procedure has been put in place. External parties can be provided with the material to perform feature extraction or phishing prediction (depending on the module) from emails. The third parties can then run this code on their internal networks, without any researcher having to see the actual sensitive data. As will be discussed in Chapter 6 and Chapter 7, two out of the three machine learning models are built from features. These models work by analysing an email and represent the email as a feature vector, where every feature holds interesting information for the machine learning model to analyse. This translation from email to feature vector can be done at the third parties. Then, for each sample, a feature vector can be extracted from the third parties, and used for further analysis in this research. For the third model, the website module Section 7.1.3, no feature vector is extracted, but a (set) of regression scores are correlated with every sample. These scores can be exported from the network of the third party.

Since we were thus not able to inspect the emails ourselves, the phishing emails sometimes needed some pre-processing. The emails in the dedicated ”report phishing” inbox, were always sent by legitimate users since they were reported. It turned out that the emails in the dedicated email inbox could be categorised in three ways:

- The legitimate user sent the phishing email as an attachment
- The legitimate user forwarded a phishing email
- A phishing email was reported, a response was sent from the ”report phishing” inbox email address, and the user send another response. i.e. an email thread occurred.

For every email in the ”report phishing” inbox, it should be analysed which of the three categories the mail belonged to. This was needed so the system could pick the right email to analyse as a phishing mail.

5.1.3. Eye

The dataset provided by Eye Security[21] was first labelled by customers of Eye. Eye is a cyber security company offering managed detection and response services to companies. The employees of these companies have the opportunity to report to Eye any emails which they suspect to be phishing. These emails were manually relabelled to separate phishing from spam.

5.1.4. Personal

In order to obtain a lot of benign emails, several personal inboxes can be used. This results in a dataset of 0 phishing emails and 1821 benign emails, starting from 2016 up to 2022. That the number of phishing emails in the personal dataset is zero is based on personal experience. In the time from which the dataset originates, no phishing email has been received or observed. Though, not every email has been reevaluated when this dataset was selected. There is a very slight chance that the number of phishing emails is not exactly zero.

Figure 5.1: Screenshot of an email from the personal dataset

5.1.5. Dataset enrichment

The Fusion module makes use of several datasets, which all consist of emails. Those emails could either be phishing or benign. The emails in the datasets are all from between 2015 and the present. As discussed before, a phishing email as considered in this research, contains a link to a phishing web page. Though, phishing websites do generally not have a long lifetime [42]. Therefore, many phishing websites in the emails from the dataset will probably no longer be available to analyse. One part of the Fusion module (Chapter 6) tries to identify phishing emails based on brand recognised in screenshots. Therefore, emails with offline phishing pages are enriched with 'new' pages. The URLs in these samples will be kept as they are, but the visuals analysed are taken from a dataset containing screenshots from almost 30.000 phishing pages. To make sure the phishing pages match with the email, the brand of the email is determined first.

For the replacement process, it should first be identified which brand the email is trying to impersonate. If this can be established, by use of the brand recognition system Section 6.1.1, the phishing URL can be enriched with a screenshot from another phishing website. Though not all emails will impersonate a brand, or the system will not be able to identify which brand the email is impersonating. Those emails will be given a phishing web page of a random brand. By statistics gathered after the experiments had been conducted, it turned out that the brand recognition system recognised a brand in roughly 20% of the emails.

As a next step, the URL which originally led to the phishing website should be identified. Emails can hold many URLs. Not all URLs in a phishing email are malicious. For example, some phishing emails place URLs to benign websites in order to evade detection.

Therefore, a decision process has been made to decide which of the URLs is most likely phishing. We will call a URL which points to an online website a valid URL, and a URL which points to an offline website an invalid URL. This validity check is done on the original URL from the email and not on the domain. As can be seen in Figure 5.2, whenever an invalid URL is found, this URL will be labelled as the phishing URL. In this research, screenshots will be taken of web pages. For this URL, no screenshot shall be taken. This URL will be given a screenshot from a dataset containing screenshots of phishing web pages. Hypothetically, it might be the case that the invalid URL used to point to a benign website and that the phishing website is still online. This would result in a phishing email sample with two phishing websites. Though, since the lifetime of a phishing web page is far smaller than that of benign web pages, this is very unlikely. Therefore the assumption is made that this cannot happen.

Phishing pages dataset The phishing pages used for enriching the dataset consist of 29496 screenshots of phishing pages of 276 different brands, which together comprise more than 99.1% [35] of all

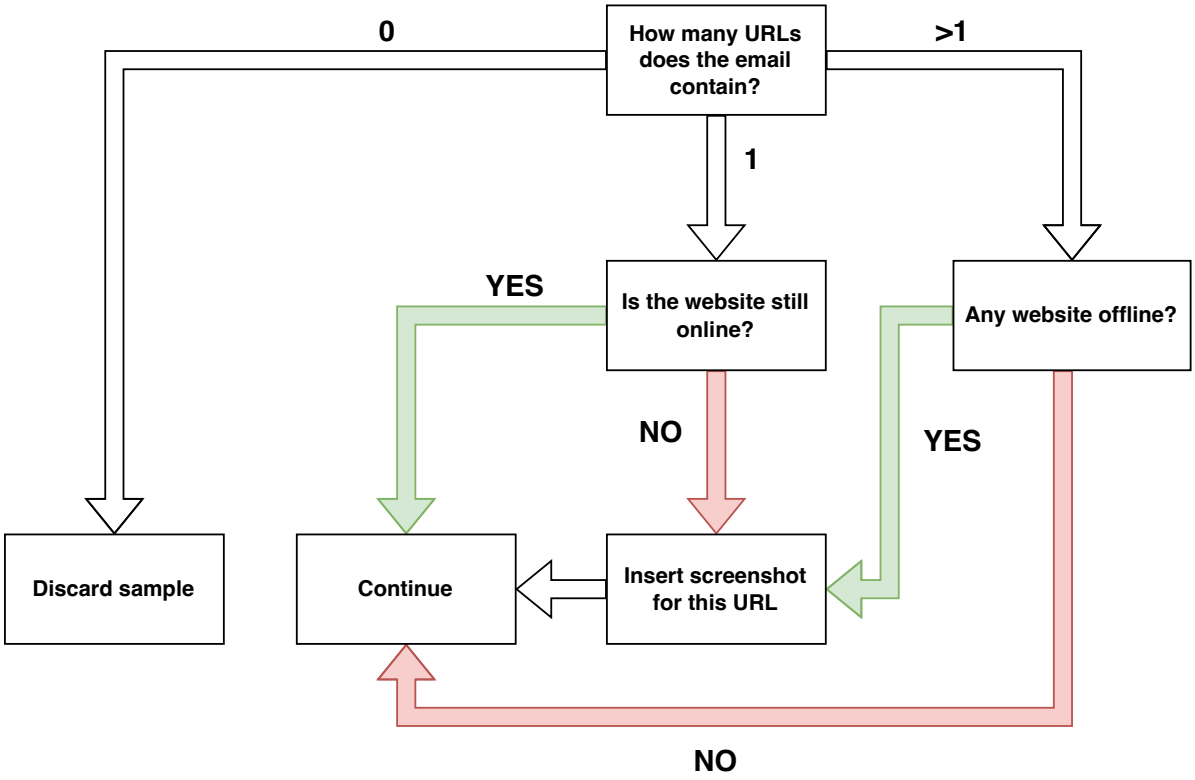


Figure 5.2: Flow for deciding which URL to enrich with screenshot

brand impersonation phishing attacks. This dataset has been scraped by Lin et al. [35] from Openphish in 2019 and late 2020. This dataset holds a wide variety of phishing emails, both of high- and low quality.

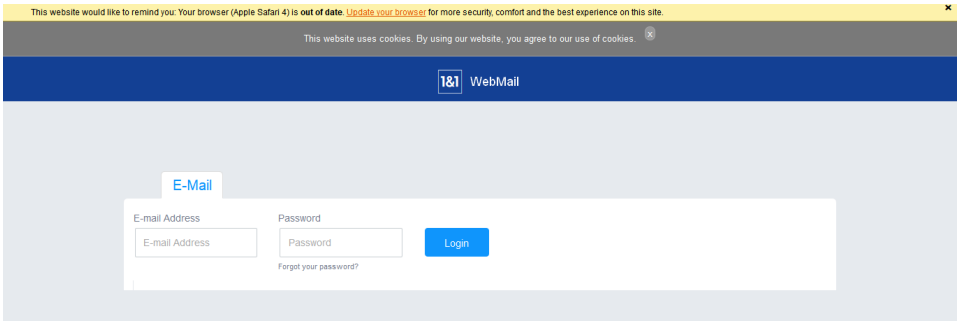


Figure 5.3: Phishing page example

5.2. Fusion Module datasets

In Chapter 6, several sub-system will be evaluated. The validation will be done with the use of datasets which do not contain emails. For the validation of the logo detection, brand identification and whole brand recognition systems, several datasets were used. Also, the performance of reverse image search is validated using a newly crafted dataset

5.2.1. Logo detection dataset

The logo detection model has been pre-trained on a dataset of screenshots of web pages[35]. Those web pages might contain none, one or more logos. For every logo in a screenshot, the coordinates of the logo are known. This dataset has been split into a training and a test set. The training set has been

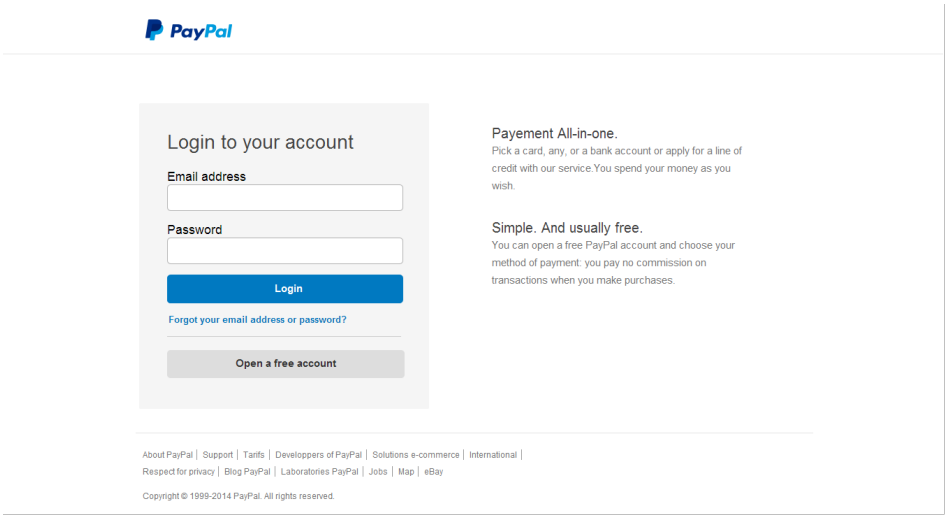


Figure 5.4: Phishing page example

Table 5.3: Non-email datasets used during the validation of sub-systems used in the Fusion module

Dataset	Content	Samples used for testing	Used for validation of Logo Detection	Used for validation of Brand Identification	Used for Reverse image search validation
Phishpedia	Screenshots of webpages	1600	TRUE	FALSE	FALSE
Phishpedia	Logos	592	FALSE	TRUE	FALSE
Handekar	Logos	125	FALSE	FALSE	TRUE
COCO	Random pictures	125	FALSE	FALSE	TRUE
Output of logo detection model	Predicted logos	60	FALSE	FALSE	TRUE

used by Lin et al. to train the logo detection model. The test set is used for validation. This test set will also be used during this research. The test set contains 1600 images.

Figure 5.5: Screenshot of web page without logo from logo detection dataset

5.2.2. Brand identification dataset

As for the logo detection model, the brand identification model has been pre-trained on a dataset[35]. This dataset consists of 2963 logos for a total of 181 brands. 2372 of those logos are used by Lin et al. to train the model, 592 of those logos can be used for testing.

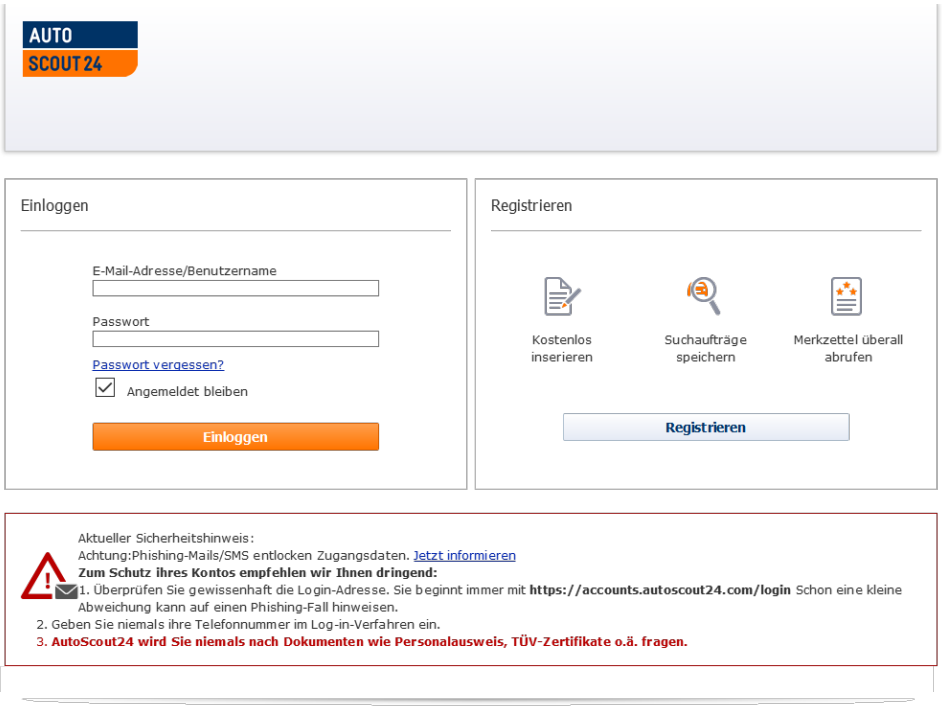


Figure 5.6: Screenshot of web page with a logo from the logo detection dataset

5.2.3. Brand recognition dataset

For testing the brand recognition model as a whole, a part of the Nazario dataset (Chapter 5), has been used. Also, emails from a personal inbox have been used. The whole dataset has been labelled manually with a brand as a label per email. The labelling of the email means that an email was opened, and whenever the email contained visual objects of a certain brand, the label given to this email would be that brand. The task of the brand recognition system is to identify this brand.

The dataset picked for the first test is the combination of the Nazario 2015 and 2020 datasets. Not all samples are useful as some emails impersonate a brand on which the model has not been tested. Other emails obviously impersonate a brand, but the logos are not available anymore. (e.g. email tries to download the file from a deprecated link)

Table 5.4: Number of emails per source in the dataset for Brand Recognition

Source	Number of emails
Nazario	289
Personal	39

5.2.4. Reverse image search datasets

For the validation of reverse image search, two datasets were used.

Logos and random pictures The first dataset consists of two parts:

- 125 logos from different brands [32]
- 125 random pictures [14]

The random pictures can be a picture taken by a camera, as long as the picture does not contain a logo. The pictures do not include computer generated pictures.

Prediction by Logo Detection model: logos and parts of emails or websites The second dataset is created by running the logo detection model from Section 6.1.1 on parts of the datasets from Chapter 5. This logo detection system will detect possible logos in websites and emails, but not every prediction is correct. From these predictions, 60 predicted boxes were selected and labelled. The dataset consists of 60 pictures:

- 30 fully visible logos from brands
- 30 pictures without a logo

Figure 5.7: Example of a ”non-logo” from the created dataset, detected by the logo detection system.

Figure 5.8: Example of a logo, detected by the logo detection system

6

Phishing attack detection by merging attack stages

After the analysis from Chapter 4, it is now clear where a detection capability gap is showing. In this chapter, the attack stage comprising the phishing email and the attack stage comprising the phishing web page will no longer be handled as isolated entities. Instead, this chapter will investigate shared informational aspects between the two stages to find features indicative of an email being phishing or not. After several features have been engineered, a phishing detection system is designed and tested. The performance of the phishing detection system is analysed.

6.1. Methodology

This section describes the methodology for the fusion module. First, the datasets used for evaluation will be described, then the design of the brand recognition system is explained, the design and explanation of the features used for the fusion module are set out. Finally, the machine learning models are listed.

6.1.1. Brand recognition system

Barracuda networks analysed more than 12 million email attacks over 17,000 organisations between May 2020 and June 2021. 49% of those were emails trying to impersonate another brand, so the receiver of the email would think the email originated from a legitimate source[3]. With such an enormous impact on the cyber security landscape, it would be a great contribution to shrinking this problem. Phishing emails try to impersonate a brand by adjusting the name of the email address owner, using domains similar to those of a brand, spoofing the brand, placing logos of a brand inside the email and many more. If one would be able to predict whenever an attacker is pretending to send an email in name of a brand, this would be the end of a large attack vector in phishing campaigns. One approach would be to identify logos of a certain brand in email, and upon detection of the logo, verify whether the email is actually sent by that brand. One part of the Fusion Module will thus be a brand recognition system. This brand recognition can be separated into two stages.

1. Logo detection
2. Brand identification

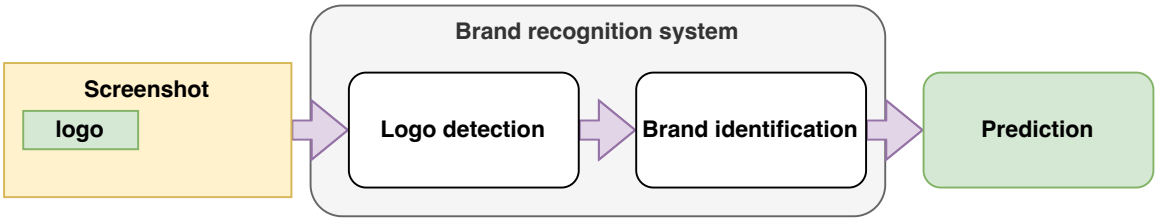


Figure 6.1: Brand recognition system architecture

Logo detection in email The goal of the logo detection model is to identify any possible logos in an email. The logo detection in email has been built from the logo detection system from Phishpedia (Section 2.5.1).

The system works in the following way:

- Save an email in the form .eml to .html format.
- Open email.html in headless selenium browser.
- Take a screenshot of the email
- Feed the screenshot to the logo detection model from Section 2.5.1

The full chain has been visualised in Figure 6.2.

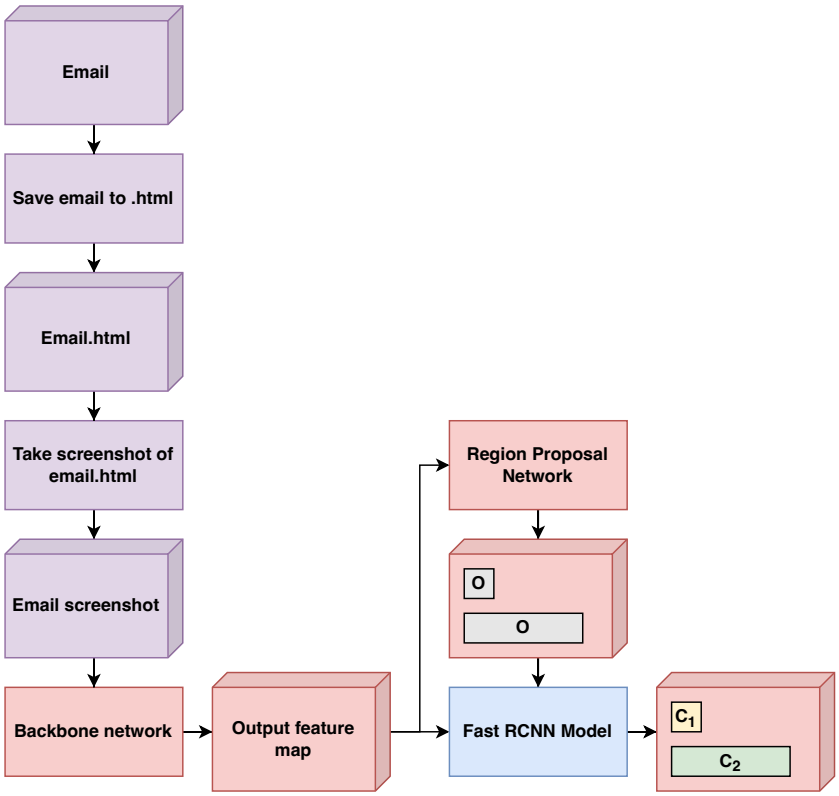


Figure 6.2: Logo detection model. The code for the purple blocks has been created during this thesis. The code for the red blocks has been created by Phishpedia [35], and the code for blue block is code from Phishpedia, with improvements made during this thesis.

The input of the logo detection system is an email, and the output is a screenshot of the email, along with coordinates of bordered boxes. The system expects that the bordered boxes contain logos.

Brand identification Whenever a logo has been detected as described in the previous section, the model will try to identify the name of the brand of that logo. The brand identification makes use of the brand identification of the Phishpedia model, as described in Section 2.5.2.

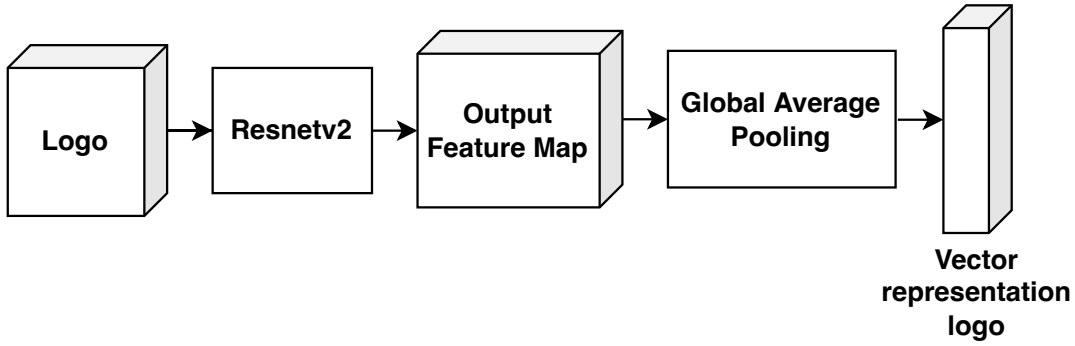


Figure 6.3: Logo vector representation model

The detailed workings of the brand identification can be seen in Figure 2.4 and read in Section 2.5.2, but for the sake of clarity, a quick summary is given here. First, a model has been trained by Lin et al. [35] on a dataset of logos of over 2000 brands, the Logo2K+ dataset. After this, that trained model can be used to create a vector representation of a logo image, as can be seen in Figure 6.3. The brand identification model is able to identify a fixed set of brands. For all those brands, several logos are collected. These logo images are run through the network from Figure 6.3, to create vector representations. After the vector representations are created, the model is ready to identify the first logo. This will result in N vector representations of logos, for which the brand identification model knows the exact brand.

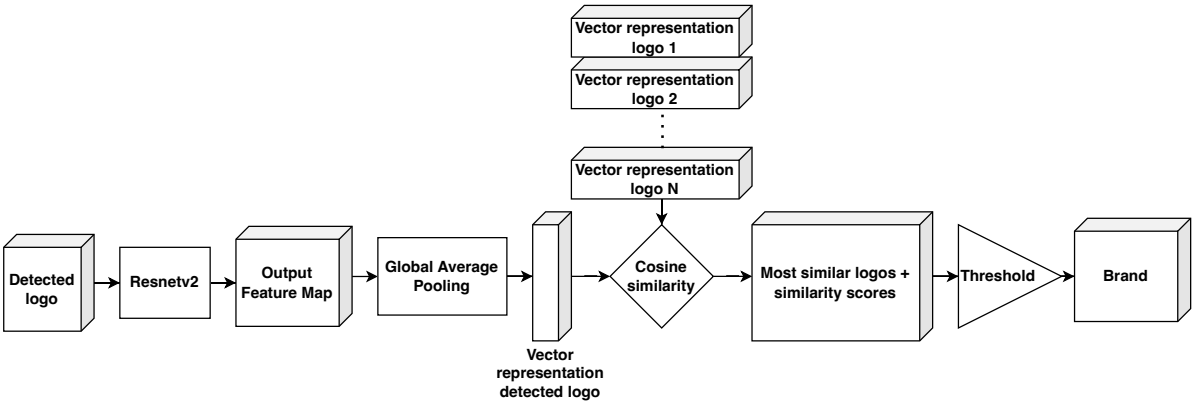


Figure 6.4: Brand identification model

The brand identification model will be given a logo to identify. Let's call this logo image **A**. Upon receiving logo **A** to identify, the model will first create a vector representation of the logo image. This vector representation will be compared to the vector representations of every logo the brand identification model holds. The comparison is done by means of calculating the cosine similarity between the vector representations. Thus, the cosine similarity between **A** and all N vector representations is calculated. The vector representations with the highest cosine similarity are further investigated, say for example that this is the vector representation of logo **X**. When the cosine similarity between **A** and **X** is above a certain threshold, logo **A** and **X** are considered to be sufficiently similar. Logo **A** will then be identified as the brand corresponding to the brand of logo **X**. If there is, however, no logo in all N logos which are sufficiently similar, logo **A** will be identified to have 'no brand'.

The brand identification model will never be able to classify a brand which is not in the pre-defined brand set.

Identifying brands outside the training set From a phishing point of view, it is not a large problem that the brand identification system is only able to detect brands which are in a pre-defined dataset.

According to Lin et al. [35], 99.1% of the phishing emails target 181 brands. Therefore, by being able to detect those 181 brands, brands in most phishing emails can be detected. Though, when looking at benign emails, it is important to also be able to detect brands which are not in this set. During the feature engineering (Section 6.1.2), it might be important to be able to tell the difference between no brand detected, and a brand detected which is out of the scope of known brands. Therefore, whenever the brand recognition system is not able to recognise a brand, there is one more step taken to try and recognise the brand. From a high level, this can be depicted as done in Figure 6.5. After a logo has

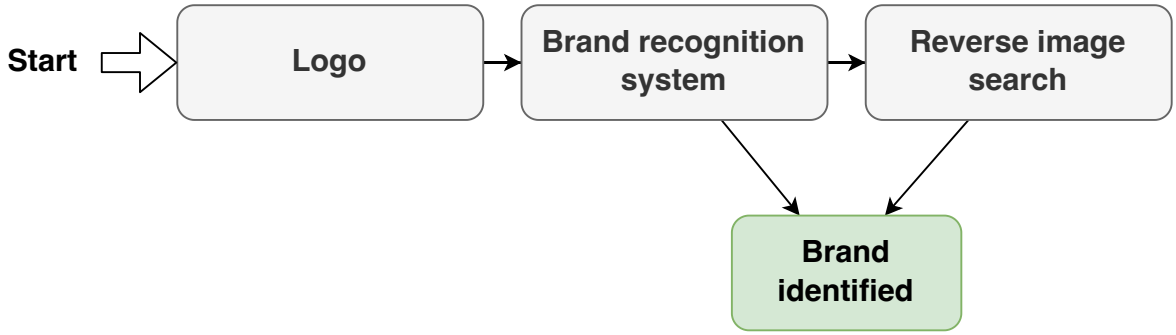


Figure 6.5: Extended brand recognition from a high level

been identified, the model will run the brand recognition system to try and recognise the brand. If this is unsuccessful, the system will proceed to the second step. The second step is by saving the logo found and performing a reverse image search on Google. Google will then return a prediction of what the image contains or represents, along with similar looking pictures. When performing reverse image search on a logo, the prediction of what the image contains or represents can be the name of the brand. Therefore, after the reverse image search has been conducted, the system will perform a google search on the term found with the image. The most similar picture as presented by Google will be translated into a representative feature vector by using the brand identification system. This vector will, again, be compared to the representative feature vector of the logo found by cosine similarity. If these two vectors are similar enough, i.e. cosine similarity is above a certain threshold, the term found by the reverse image search will be adopted as the name of the brand. The reverse image search will be done using the Cloud Vision API[25], which provides an API for Google reverse image search.

The steps are depicted in more detail in Figure 6.6. While the brand recognition model is only able to detect brands on which it has been trained, this system will be able to recognise brands outside of this set. Thus **making it a very scalable, and globally usable system.**

The Google Cloud Vision API also offers a service of labelling any logos present in pictures, which we call the ”Google logo Detection” method. This service takes as input any picture and will detect a logo inside that picture, and label it with the brand. The performance of this service will be compared to the performance of the previously described reverse image search.

Identifying similar logos without recognising brand One of the features described in Section 6.1.2 makes use of the brand recognition system, without actually identifying a brand. This feature compares the feature vector representations of logos found on websites and emails. Using this technique, a brand is not necessarily identified. Though, the similarity between the two logos does imply that the same logo has been identified on both the websites and emails.

6.1.2. Feature engineering

After the analysis of Section 3.1, two stages within the phishing attack chain were identified for which many phishing detection mechanisms were designed to work independently. The goal of this chapter is to explore the possibilities of performing phishing detection on information shared between the two stages. Thereby creating a phishing detection mechanism which combines the two stages. These two stages each comprise a different medium, a phishing email and a phishing website respectively. First, aspects should be defined which occur in either of the two stages and might distinguish a phishing email with a phishing website from a benign email containing a benign website. By means of **brainstorms in combination with the experience of senior cyber security specialists**, a

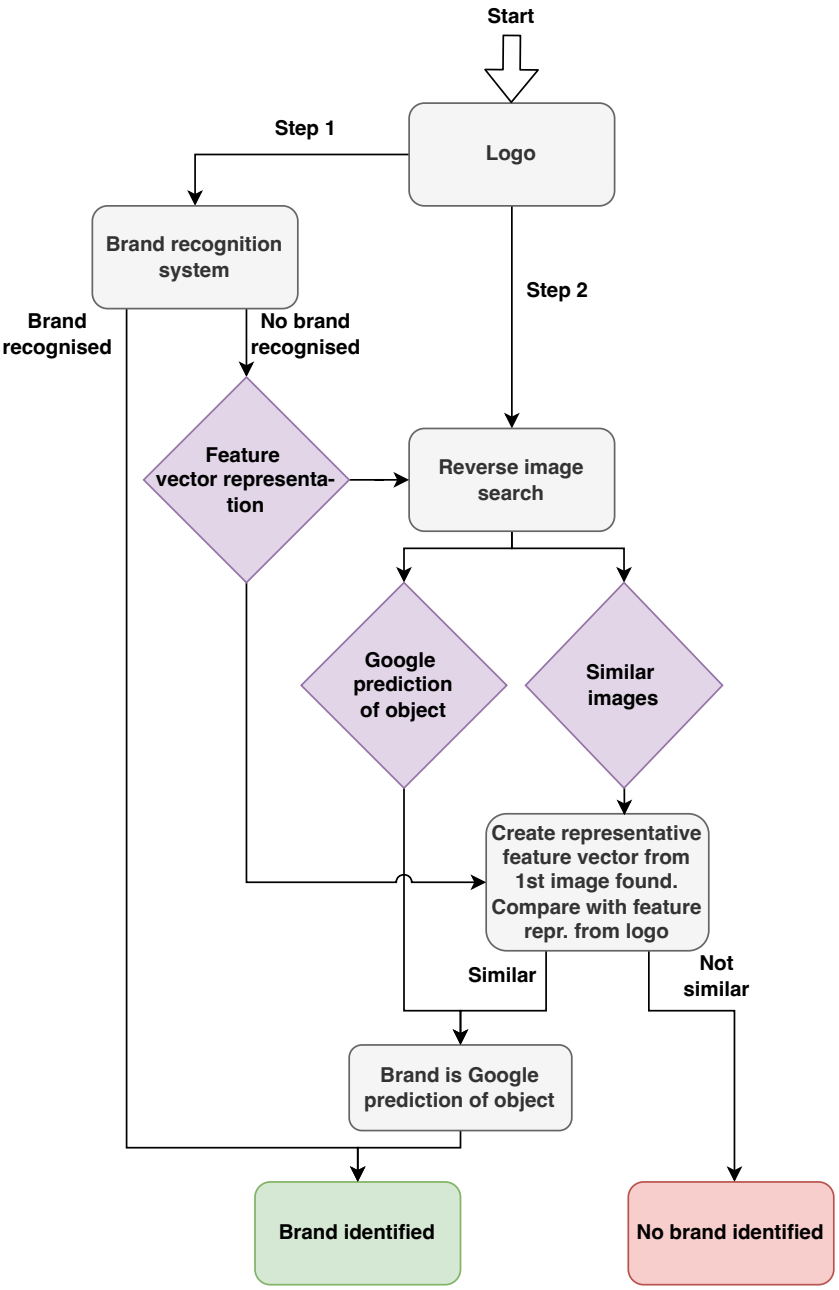


Figure 6.6: Extended brand recognition model

list (see Table 6.2) of possible indicators of an ongoing phishing attack was conceived. This list was mainly aimed at indicators which use information from both the email and websites combined. Those indicators can be seen as ”features” which can be extracted by investigating and processing emails and websites which are linked in the email. A feature works in the following way, the system gets an email containing links. For every feature, the system will process information from the email, into a value. For example, the system gets an email as input. For the 6th feature in Table 6.2, the system will run a brand recognition module to try to recognise a brand in the email. If that is the case, the value of this feature for the email sample will be set to 1. The system then continues to extract the information from the email and its websites for every feature. This way, the email is represented as a vector containing numbers, which we will call a feature vector. The expected behaviour is that benign and malicious emails and websites hold different values for certain features or combinations of features. During this research, a machine learning model will be built that trains on feature vectors for both benign and be-

nign emails (containing websites). By training this network, it will be able to recognise certain patterns in the feature vectors for benign and malicious emails. This model will then be able to detect those patterns in emails, resulting in a phishing detection mechanism.

The features are all divided into categories and sub-categories. A feature can have only one category, but multiple sub-categories. In Table 6.1, the different sub-categories are listed, along with the number of appearances.

Each feature has been described below. Three feature types can be distinguished. Binary features, producing either a ”1” or a ”0”, continuous features, producing a value and categorical features, producing a value which represents a certain pre-defined category.

1. **Any links redirects to main domain of large company** (*Binary*) Some phishing campaigns target specifically on certain targets. If someone who is not specifically targeted or a crawler clicks the phishing link, the link will redirect to a benign web page.

2. **At least one website with no logos** (*Binary*) There is at least one website where the logo detection system does not detect any logos. This might indicate that the website is empty. Some phishing attacks might be redirecting a crawler from a certain location to an empty website.

3. **Average number of logos detected by logo detection per website** (*Continuous*) The average number of logos detected per website. A low average number might indicate relatively empty websites.

4. **Brand but from common email provider** (*Binary*) A brand is identified in the email, but the email is sent from a common email service provider. Combination of features 6 and 13.

5. **Brand in email is social media** (*Binary*) The brand recognition system is used to identify a brand in the email. If a brand is identified and this brand is a social media brand, this feature is ”1”. If there is not a social media brand detected, or no brand is identified this feature is ”0”. This feature is introduced to try to prevent false positives introduced by feature 6. It is very legitimate and not uncommon to include the logos of social media brands in emails.

6. **Brand is recognised in email** (*Binary*) The brand recognition system is used to identify a brand in the email. If a brand is identified, this feature will have value ’1’, otherwise ’0’. This feature alone can not be used to distinguish phishing from benign emails. The presence of the logo of a brand in an email can be both benign and phish behaviour. Though, this might enrich other features in the fusion module.

7. **Brand of website is mentioned in email** (*Binary*) All links in an email are visited and on all websites, the brand recognition system is used. If the name of any identified brands is mentioned as text in the email, this feature is ”1”. If brand(s) are identified but not mentioned, this feature is ”0”. Else: ”2”. This adds to the possibility of brand impersonation in an email. Though this is very common behaviour, this feature might enrich other features in the fusion module.

8. **Certificate type of domains of links in email** (*Categorical*) The certificate type of the domains of all the URLs in an email. Value is as a 3-bit binary representation where the left bit shows the presence of a domain with DV certification, the middle bit is a domain with OV certification, and the right bit represents the EV certificate. The integer value of this binary representation is now the value for this feature. e.g. when there are two domains with DV and EV certificates: 101 -> 5. Only OV certificate: 010-> 2. The assumption here is that benign emails show more complex certificates such as Domain Validation and Extended Validation

9. **Certificate type of email domains** (*Categorical*) The set of certificate types of the certificates of the email ”from:” domains in the email. Value is as a 3-bit binary representation where the left bit shows the presence of a domain with DV certification, the middle bit is a domain with OV certification, and the right bit represents the EV certificate. The integer value of this binary representation is now the value for this feature. e.g. when there are two domains with DV and EV certificates: 101 -> 5. Only OV certificate: 010-> 2. Multiple ”from:” domains are only present in an email thread.

Table 6.1: Sub-categories for Fusion module features

Sub-categories	Count	Tool	Description
Email brand	9	brand recognition system	The recognition of a brand in an email is done by the brand recognition system.
Website brand	6	brand recognition system	The recognition of a brand on a website is done by the brand recognition system
Email logo	1	logo detection system	The detection of a logo in an email is done by using a sub-part of the brand recognition system, the logo detection system
Website logo	1	logo detection system	The detection of a logo on a website is done by using a sub-part of the brand recognition system, the logo detection system
Website layout	3	logo detection system	For the website layout, it will be investigated whether the website is empty. This is done using the logo detection system.
Website URL redirect	2	requests	When a URL is visited, the URL might point to another URL, redirecting the visitor to a new URL. Tools such as the Requests library for Python can register such a redirection and output which URLs are all visited.
Certificate	6	sockets/crt.sh	SSL certificates are used to authenticate a domains identity and enables security for communication. There are three types of certificate types, Domain Validation (DV), Organisation Validation (OV) and Extended Validation (EV). Those certificates can hold important information on the website owner. For this sub-category, either sockets are used for websites which are still online or the api to the website crt.sh is used to see certificate history.
website domain	7		Features of this sub-category use domains of URLs in the emails.
Email 'from:' domain	11		Makes use of the domain of the email address from which the email originated, the 'from:' domain.
email content	2		Uses the content of an email.
domain registration date	5	whois	The registration date of a domain is publicely available knowledge. This is used in features from this sub-category.
DMARC	1	checkdmarc	DMARC is used to validate the authenticity of an email. When a DMARC fails, this could indicate spoofing of an email.
SPF	1	checkdmarc	Makes use of the SPF of an email domain. For this, the same tool as the DMARC check can be used.

10. **Days since newest mail domain registration** (*Continuous*) The number of days since the oldest domain of the "from" email address has been registered. Multiple "from:" domains are only present in an email thread. A newly registered "from:" domain is suspicious due to fast rotating phishing domains.

11. **Days since oldest mail domain registration** (*Continuous*) The number of days since the old-

est domain of the ”from” email address has been registered. Multiple ”from:” domains are only present in an email thread.

12. **DMARC check success** (*Binary*) The DMARC record is checked for all the ”from:” domains in the email. If any of the domains fails the DMARC check, this value is ”0”, else ”1”.

13. **Domain is of common email provider** (*Binary*) The ’from:’ domain of the email address is that of a common email service provider (Gmail, outlook etc.) (See Chapter B for the full list). A specific type of phishing email is often sent from emails registered at common email service providers. This is often observed for a different type of phishing attack than a credential phishing attack. Though, it is still interesting to see whether the abuse of common email service providers happens in credential phishing emails.

14. **”From:” email address is owned by brand in email** (*Categorical*) When a brand is recognised in email, the domain of the email address is checked. If the domain has a certificate, the owner of that domain will be verified. When a domain has a SSL certificate with Organization Validation or Extended validation, the organization that registered the domain is mentioned in the SSL certificate. The brand recognition system might produce a slightly different brand name than the official brand name which is mentioned on the domain. E.g., ’Airbnb’ might be recognised, but ’Airbnb, inc.’ is the name on the certificate. To verify whether the brand recognised actually is the brand which has registered the domain, it is checked if the name of the recognised brand from the email is a subset of the organization name on the certificate. There are 6 possible classes for this feature. This feature will output the number of the class. 0: ’from:’ domain owned by another brand or when the email contains a brand, but the domain does not. 1: Correct brand between email and domain. 2: If a domain is owned by cloudflare. Cloudflare provides a service where the certificate will be registered by Cloudflare. In this case, the brand name on the certificate is ”Cloudflare”, but that is not the actual brand name. 3: Email has no brand, but the domain does. 4: If the email has no brand, and domain does not have a brand name. 5: If the email does not have a ”from:” email address. An email might be an email thread, containing several ”from:” domains. In that case, the check is done on all domains. If any of the domains result in 1 or 2, the function is terminated and the result is obtained. Otherwise, continue to the next domain.

15. **Maximum days between registration of mail domain and website** (*Continuous*) The maximum difference (in days) between the registration dates of any of the email ’from:’ domain and any of the websites. Multiple ”from:” domains are only present in an email thread.

16. **Minimum days between registration of mail domain and website** (*Continuous*) The minimum difference (in days) between the registration dates of any of the email ”from:” domain and any of the websites. Multiple ”from:” domains are only present in an email thread. The difference in domain registration time is interesting to investigate. This is done to see the whether domains from which an email originates and the URLs it states are equal or registered around the same time.

17. **Minimum days since registration of any link** (*Continuous*) The minimum days since registration of any of the domains which are listed in the email. The assumption is that phishing domains are registered fairly recently.

18. **Nr of brands recognised in email** (*Continuous*) The brand recognition system is used to identify brands in the email. The output of this feature is the total number of brands recognised. When the same brand is identified multiple times, this will not result in multiple detections. The reason for investigating this feature closely relates to the reason of feature 6.

19. **Nr of times brand in website but different from email** (*Continuous*) All links in an email are visited and on all websites, the brand recognition system is used. The value of this feature is the total number of websites with at least one identified brand, where none of the identified brands is identified in the email. (This value of this feature is the value of feature 23 minus feature 20).

20. **Nr of times brand in website is same as brand in email** (*Continuous*) All links in an email are visited and on all websites, the brand recognition system is used. The value of this feature is

the total number of websites where the identified brand is the same as any of the brands in the email. This feature is investigated to see whether any distinctive behaviour between benign and phish can be seen here. A phisher might impersonate one brand and phish for another (e.g. Microsoft). But on the other hand, a user sharing a document might show similar behaviour.

21. **Nr of times logo in website is same as logo in email** (*Continuous*) All links in an email are visited and on all websites, the logo detection system (step 1 of the Brand recognition system) is used on all websites and on email. Any logos detected by the detection system is fed to a Resnetv2 network to extract features from the logo. Per website, the feature extractions from all the logos are compared to the detected logos in the email. The value of this feature is the number of websites where a feature vector from the website was similar enough to any of the feature vectors from the email. This feature is not able to recognise a brand, but it will be able to identify whether the logos from a website and email are the same.

22. **Nr of URLs in email** (*Continuous*) The number of URLs is counted. This feature has the total number of all links. This is a measure of how complicated emails are. It might be the case that phish and benign emails differ here.

23. **Nr of websites with brand recognised** (*Continuous*) All links in an email are visited and on all websites, the brand recognition system is used. The value of this feature is the total number of websites where a brand has been identified.

24. **SPF check success** (*Binary*) The SPF record is checked for all the ”from:” domains in the email. If any of the domains fail the SPF check, this value is ”0”, else ”1”. A failing SPF check might indicate either spoofing or a badly configured email server.

25. **URL in email performs redirect** (*Binary*) All links in an email are visited. If any URL in the email performs a redirect, this feature has a ”1”, and a ”0” otherwise. Some malicious behaviour here would be that phishers use a URL shortener to obfuscate phishing domains. Though, benign emails might use redirects in marketing services. It is interesting to see if any of these hypotheses will be indicative of phishing.

26. **URL with same root domain as from address** (*Binary*) All links in an email are visited. After any redirects, the final domain is compared to all the ”from:” domains of the email. If any of the root domains is the same as any of the domains of the ”from” email address, this feature is ”1”. Else: ”0”. Multiple ”from:” domains are only present in an email thread. For much legitimate usage, a company sends an email from the company domain. It is often observed that the company domain is also in either the footer or elsewhere in the email. Does phishing behave the same?

27. **URLs of unknown brands exist** (*Binary*) All links in an email are visited and on all websites, the brand recognition system is used. If there is no brand identified on the websites, the certificate of that domain is also checked. If this certificate is also not registered by a company, the URL is deemed to have no brand. If there exists at least one URL without a brand, this feature is ”1”, else ”0”.

28. **URLs of unknown brands exist and brand recognised** (*Binary*) All links in an email are visited and on all websites, the brand recognition system is used. If there is no brand identified on the websites, the certificate of that domain is also checked. If this certificate is also not registered by a company, the URL is deemed to have no brand. If there exists at least one URL without a brand but the email does contain a brand, this feature is ”1”, else ”0”. This feature is a combination of features 6 and 27. The assumption here is that when a phisher sends an email with a brand, all URLs in that email should lead to either a company domain or domains of other brands (e.g. file sharing). Therefore, we are interested in seeing how much of both benign and phishing emails show this behaviour.

29. **Website without logo but brand in email** (*Binary*) One of the links in the email leads to an empty website but there is a brand detected in the email. (This feature combines features 2 and 6). Sometimes, a phishing attack is targeted at a specific user or in a specific region. If anyone else than the predicted target visits the phishing page, the domain returns as an empty page.

Table 6.2: Feature list for the fusion module

Nr	Category	Sub-categories	Low-level technique	Name
1	Website	Website URL redirect, certificate	Selenium, requests, sockets/crt.sh	any links redirects to main domain of large company
2	Website	Website layout	Selenium, logo detection system	at least one website with no logos
3	Website	Website layout	Selenium, logo detection system	average nr logos detected by logo detection per website
4	Email	Email brand, email 'from:' domain	Selenium, brand recognition system	brand but from common email provider
5	Email	Email brand	Selenium, brand recognition system, list_of_social_m	brand in email is social media
6	Email	Email brand	Selenium, brand recognition system	brand is recognised in email
7	Website + email	Website brand, email content	Selenium, brand recognition system	brand of website is mentioned in email
8	Website	Website domain, certificate	sockets/crt.sh	certificate type of domains of links in email
9	Email	Email 'from:' domain, certificate	sockets/crt.sh	certificate type of email domains
10	Email	Email 'from:' domain, domain registration date	whois	days since newest mail domain registration
11	Email	Email 'from:' domain, domain registration date	whois	days since oldest mail domain registration

12	Email	Email 'from:' domain, DMARC	checkdmarc	Dmarc check success
13	Email	Email 'from:' domain	sockets/crt.sh, email_service_pr	domain is of common email provider
14	Email	Email brand, email 'from:' domain, certificate	Selenium, brand recognition system, sockets/crt.sh	from email address is owned by brand in email
15	Website + email	Email 'from:' domain, website domain, domain registration date	whois	maximum days between registration of mail domain and website
16	Website + email	Email 'from:' domain, website domain, domain registration date	whois	minimum days between registration of mail domain and website
17	Website	Website domain, domain registration date	whois	minimum days since registration of any link
18	Email	Email brand	Selenium, brand recognition system	nr of brands recognised in email
19	Website + email	Email brand, website brand	Selenium, brand recognition system	nr of times brand in website but different from email
20	Website + email	Email brand, website brand	Selenium, brand recognition system	nr of times brand in website is same as brand in email
21	Website + email	Email logo, website logo	Selenium, logo detection system	nr of times logo in website is same as logo in email
22	Email	Email content		nr of URLs in email

23	Website	Website brand	Selenium, brand recognition system	nr of websites with brand recognised
24	Email	Email 'from:' domain, SPF	checkdmarc	SPF check success
25	Website	Website URL redirect	requests	URL in email performs redirect
26	Website + email	Website domain, Email 'from:' domain	Selenium	URL with same root domain as from address
27	Website	Website brand, website domain, certificate	Selenium, brand recognition system, sockets/crt.sh	URLs of unknown brands exist
28	Website + email	Website brand, email brand, website domain, certificate	Selenium, brand recognition system, sockets/crt.sh	URLs of unknown brands exist and brand recognised
29	Website + email	Email brand, website layout	Selenium, brand recognition system, logo detection system	website without logo but brand in email

6.1.3. Machine learning model

As discussed in Section 6.1.2, the fusion module will be built on top of features. The machine learning models used for the fusion module will be selected among classical machine learning models and will not make use of deep learning models. Deep learning is not used due to several reasons. First, the number of data samples is limited. Deep learning models generally need more data to train than classical machine learning models. Second, a deep learning model is far more complex and requires more computational power compared to classical machine learning. Though, the feature selection has already been done by hand. Training a deep learning model on the feature vector from Section 6.1.2 would make it needlessly complex. Third, by creating a classical machine learning model, the behaviour of the model is better to understand. By creating a more or less white-box algorithm, the model will be able to provide feedback on the reason why a sample is classified the way it is. If an email is labelled as phishing because the domain corresponding to the email address has been registered one day ago, it would increase the usability of the system.

Several machine learning models will be trained and tested for the fusion module. All systems will be evaluated, and the best performing model is selected for the final prediction. The models to try will be Random forest, XGBoost and Support Vector Machine.

Hyper parameter optimisation A machine learning model is configured using several settings. The optimal settings for every model depend heavily on the dataset which is used to train the model. The optimal settings can not be determined upfront, but are evaluated by trying many settings and

optimising for a certain statistic. During this research, a random search has been used. This means that a set of all possible settings is created, and a subset of x (in this case x=100) settings are tried. The random search outputs the optimal settings for a model, optimised for a certain statistic, in this case, accuracy. Per model, the superset is created from the following settings:

XGBoost

- Learning rate - values between .01-1
- Maximum depth of a tree used in the XGBoost - values between 2-100
- Number of trees used in the XGBoost - values between 50-450
- Minimum sum of weight needed in a child node - 1 or 5
- Minimum loss reduction needed before making a split in a tree (gamma) - values between 0-1

Random Forest

- Number of trees - values between 50-450
- The number of features to consider when looking for the best split - 'auto' or 'sqrt'
- Maximum depth of a tree - values between 2-110
- Minimum number of samples required at the split - The minimum number of samples required to split an internal node
- Minimum number of samples required at the leaf - The minimum number of samples at a leaf

Support vector machine

- Regularisation parameter - values between 1-10
- Kernel type - 'rbf' or 'sigmoid'
- Kernel coefficient - values between .001-1

6.2. Results

In this section, results of tests and performance measures for the fusion module are described. First in Section 6.2.1, the validation results for the brand recognition system is handled. Then, the feature analysis on the features of the fusion module has been done in Section 6.2.2. Lastly, the performance of the fusion module is described in Section 6.2.3.

6.2.1. Brand recognition system validation

In order to use the brand recognition system, the performance of this mechanism should be evaluated. When looking at the explanation of Phishpedia in Section 2.5, three distinct parts contribute to the final performance score of the mechanism. It is important to validate the performance of each part on individual performance, as well as the overall performance.

1. Logo detection
2. Brand identification
3. Combined performance of logo detection and brand identification

Logo detection The logo detection system can be validated separately from the brand recognition system. This validation happens by verifying how good the model is in detecting logos in images. For such a test, a dataset has been gathered of images containing logos. These images are screenshots of websites. For every image, it is known what the coordinates are to draw a box around the logo in the image. The coordinates are the labelling of the image, and thus a labelled image is defined as an image where the coordinates for the box around a logo are known.

For the test, the model was given 1600 labelled images, but the coordinates were not given to the model. The model is thus given an image containing a logo but does not know where the logo is in the picture. The task for the model was to identify a possible logo inside the image and return the coordinates of a box which contains the logo. The performance of the model has been evaluated by means of Average Precision (AP), see Section 2.3. In order to calculate the AP, first, the precision and recall must be calculated. These calculations have been explained in Section 2.3. To create the precision-recall curve, the adjusted threshold is the confidence score produced by the Fast-RCNN network Section 2.5.1. This confidence score is a value for how certain the model is that the predicted box actually contains a logo. If this confidence is close to zero, the model is not certain whether it is correct about its prediction. For a confidence score close to one, the model is certain that there is a logo in the box it predicts. One can set a threshold for this confidence score when accepting a prediction from the model. Thus any prediction for which the model has a confidence score below the threshold could simply be ignored, and any prediction with a confidence score above this threshold can be accepted as a prediction. With a higher threshold, less prediction will be accepted from the Logo detection model. Since for a higher threshold, the model is more certain about its prediction, it can be expected that the number of incorrect predictions will be lower. An incorrect prediction means that the model predicts that there is a logo inside a certain area, but there actually is not any logo in that predicted box.

For the first calculation, the threshold has been set to 0 confidence up to 0.95 with an interval of .05. The precision-recall curve has been created for different intersection over union thresholds, ranging from .5 to .95 with an interval of .05. The varying of the confidence threshold has been done for different intersection over union (IoU) thresholds. As has been explained in Section 2.3.2, setting the IoU threshold means that a prediction from the model should at least have an IoU above that threshold, to classify the sample correctly. By varying these thresholds, different performances could be identified. Do keep in mind that the IoU threshold is not a variable which can be set during the actual use of the logo detection, but is only used for validation purposes to see how well the model performs.

As can be seen in Figure 6.7, the model performs better with a lower IoU threshold. This means that often, the model does include some part of the logo to be detected, but not all. In an ideal situation, varying the IoU threshold should have a limited impact on the performance of the model. Though, the performance of the model is good, since a reasonable precision and recall can be established. Depending on the needs for the performance of the model, one can take a confidence threshold with either a high precision, high recall or an optimal precision/recall ratio. For the fusion module, an optimal precision/recall ratio is desired to misclassify as few samples as possible. Therefore, a point on the Precision-recall curve will be taken where the tangent to the curve is more or less diagonal. Taking such a point is where **the confidence score is** .6. This will be used as a threshold further on in this research.

Table 6.3: Logo detection performance results for different IoU thresholds

IoU	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95
Averge Precision	0.78	0.75	0.71	0.66	0.58	0.45	0.3	0.12	0.02	0

The result can be seen in Table 6.3 and the precision-recall curves can be seen in Figure 6.7.

Brand identification The task of the brand identification model will be to identify the name of a brand when it is given the logo of a brand. On this test, 681 brand logos were given to the model to predict. Each logo is the logo from a brand on which the model has been trained on. Though, the exact logo images from the test set were not used during training. The training set consists of 2720 logos. Per logo, the brand identification system will name a brand. Remember from Section 6.1.1, that the model can either define a brand to a logo or predict 'no brand'. The validation makes use of the following terms:

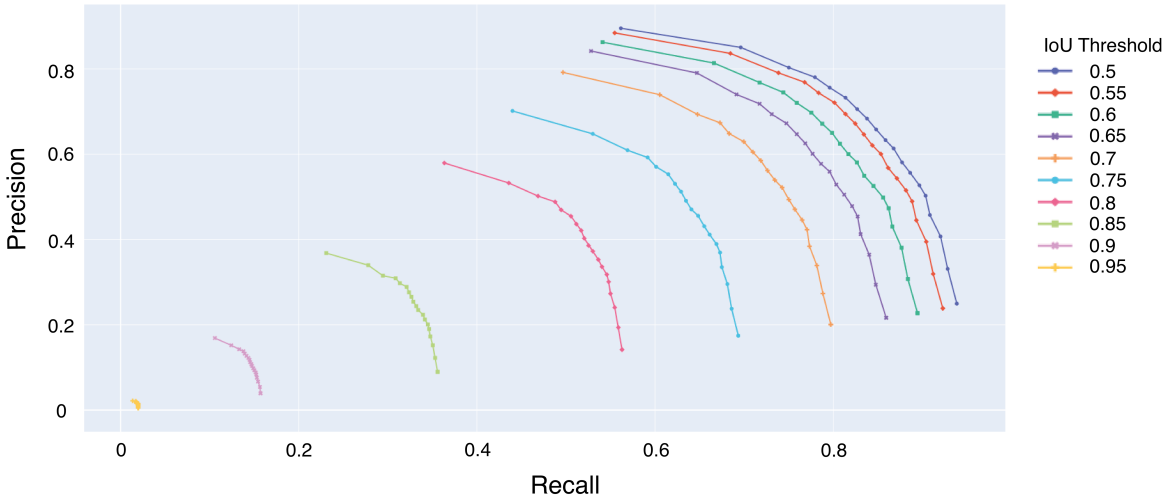


Figure 6.7: precision-recall curve for the logo detection model

- **True Positive** - The model identifies the correct brand of the logo, and the confidence is above the confidence threshold.
- **False Positive** - The model identifies the wrong brand to a logo, but the confidence is above the confidence threshold
- **False Negative** - The model predicts 'no brand' for a logo which it should be able to detect, or the model does a prediction with a confidence below the confidence threshold.

Note that a true negative would be the case where the model predicts 'no brand' correctly. Though, in this test, this is not possible, since the samples in the test dataset are all of brands on which the logo has been trained on. The model has thus not been given 'negative' (no brand) samples.

The performance of the brand identification system is highly correlated with the threshold set to the similarity score produced by the model. Remember from Section 6.1.1, that the model produces a similarity between the logo it is investigating and logos from brands it has seen before. When accepting a brand name to a logo when the similarity is too low, this will produce a lot of false positives. But setting the similarity score too high, the model will have a hard time identifying brands in a logo with enough confidence. Therefore, varying the similarity threshold has been investigated.

Table 6.4: Brand identification performance results for different similarity thresholds

Similarity threshold	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95
TP	555	555	555	555	555	555	539	441	263	78
FP	37	37	37	37	37	36	16	6	3	1
FN	89	89	89	89	89	90	126	234	415	602
Total	681	681	681	681	681	681	681	681	681	681
Precision	0.94	0.94	0.94	0.94	0.94	0.94	0.97	0.99	0.99	0.99
Recall	0.86	0.86	0.86	0.86	0.86	0.86	0.81	0.65	0.39	0.11

The results from the brand identification test can be seen in Table 6.4, and the precision-recall curve, along with the average precision (area under precision-recall curve), can be seen in Figure 6.8. The results from the brand identification test do not differentiate up to the confidence threshold of 0.7. When taking into account the decrease in confidence recall, and increase in precision, the optimal threshold probably is somewhere between .75 and .8. Though, when a false positive is expensive, one

Table 6.5: Brand identification performance results for different similarity thresholds, zoomed in on range 0.8-0.845

Similarity threshold	0.8	0.805	0.81	0.815	0.82	0.825	0.83	0.835	0.84	0.845
TP	539	534	522	518	508	496	489	471	455	447
FP	16	11	9	8	8	8	8	7	7	6
FN	126	136	150	155	165	177	184	203	219	228
Total	681	681	681	681	681	681	681	681	681	681
Precision	0.97	0.98	0.98	0.98	0.98	0.98	0.98	0.99	0.98	0.99
Recall	0.81	0.8	0.78	0.77	0.75	0.74	0.73	0.7	0.68	0.66

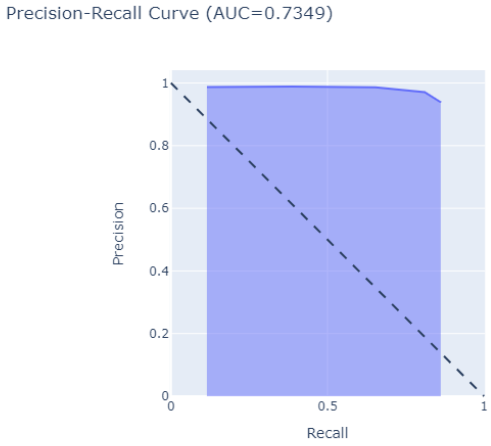


Figure 6.8: precision-recall curve for the brand identification model

might choose to use a bit of a higher threshold to reduce false positives. In that case, a threshold between .8 and .85 is desirable. To further zoom in on that range, another test has been performed, as shown in Table 6.5. When looking at how the brand identification system will actually be used, predicting a wrong brand to a logo might have large consequences in labelling an email is phishing or not. A false positive is thus very expensive for this model, and a high precision is desired. Though, the accuracy should be reasonable. Therefore, a similarity threshold of .835 has been picked.

Combined performance A test has been conducted to validate the combined performance of the first two parts. The system was given emails and needed to identify whether it could recognise any brands in that email. A test was conducted to evaluate the performance of the system for brands which are in the predefined dataset of brands as discussed in Section 2.5. This can thus be seen as a general evaluation of the system.

The dataset for the combined performance test contains emails. Whether the emails are actually phishing or benign does not matter at this point, since the purpose of this test is not to validate whether the system is able to detect phishing, but solely whether the system is able to recognise brands in the email.
Emails can be defined to be of two classes:

1. **No brand** - No brand is being impersonated
2. **Brand** - The model has been trained on the brand impersonated in the email

Do note that there is a difference between the label given to the sample and the class it belongs to. For e.g. an email trying to impersonate 'DHL' will have label: 'DHL', and be part of the 'Brand' class. All

emails with a brand labelled to them are part of the 'Brand' class. The emails with no brand do not hold any visual objects of any brand. The emails from the brand category, hold visual objects (logos) of one particular brand. Whenever an email appears to impersonate a brand by means of using its logo, that brand was labelled as the target brand. Impersonation here means the following: the email uses a logo of a certain brand in order to trick the receiver of the email into believing the email was sent by that brand. Text alone is not enough to perform a try of impersonation, though obviously fake logos still are valid.

The result of the combined performance test can be found in Table 6.6. Interestingly, the model turns out to have the following behaviour: all wrongly classified samples of category 2, are classified as 'no brand', i.e. whenever a sample should be labelled as a brand, the system either labels that brand correctly or classifies the sample as 'no brand', never does the system classify a wrong brand. All wrongly classified samples of category 1 are given a certain, and thus wrong, brand. Therefore, we can distinguish all classified samples into two categories, a sample is either given a brand (positive), or not (negative). This results in four possible classification results:

- **TP**: Sample of category 2 has been labelled with the correct brand
- **FP**: Category 1 has been labelled with a brand
- **TN**: Category 1 has been labelled as 'no brand'
- **FN**: Category 2 has been labelled as 'no brand'

Table 6.6: Per sample category for combined performance test, number of samples and the number of correct predictions done by the Brand recognition system

Category nr.	Category description	Samples	Correct
1	No brand	181	177
2	Known brand	108	87

Now we can compute the confusion matrix:

Table 6.7: Confusion matrix for the test on the combined performance of the brand recognition system

	Positive	Negative
Positive	87	4
Negative	21	177

Performance metrics are now:

- **Precision**: 0.956
- **Recall**: 0.806
- **Accuracy**: 0.914
- **FPR**: 0.022

The number of false negatives is rather high, as can be seen in Table 6.7. On several occasions, this was due to the poor quality of the phishing email. An example can be seen in Figure 6.9. Here it is clear that the logo in the email is not the real logo of DHL. Though the email clearly tries to impersonate DHL and the brand identification fails to recognise.

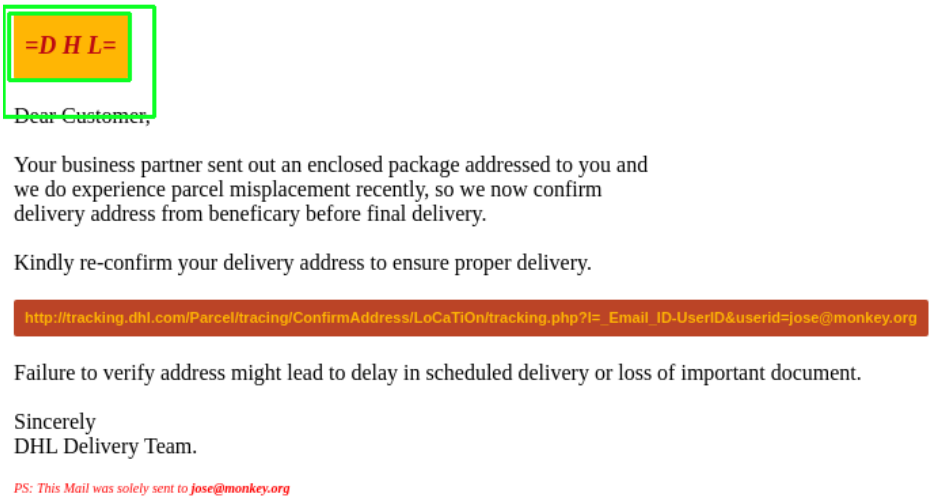


Figure 6.9: Poor quality phishing example

Combined performance for brands outside the dataset As a next step in testing the combined performance, the system is also tested on logos of brands which are not in the predefined dataset of brands. The system will never be able to correctly recognise the brand, but the behaviour of the system is nonetheless interesting to investigate.

The dataset for this second test includes all samples from the first combined performance test, as well as emails which clearly try to impersonate a brand, but that brand is not included in the dataset of the brand recognition module. Therefore, the module will never be able to predict the right brand for those emails, it simply doesn't have the brand as an option.

The samples for the second test hold three categories of emails.

1. **No brand** - No brand is being impersonated
2. **Brand known to system** - The model has been trained on the brand impersonated in the email
3. **Brand unknown to system** - The model has not been trained on the brand impersonated in the email.

The second test is designed to gain insight into the possibility to distinguish the behaviour of the system between an email with no brand, and an email with a brand it simply doesn't know. I.e. see the difference between categories 1 and 3 of the dataset. As mentioned before, when the system is able to distinguish between "no brand in email" and "Brand not in predefined set in email". Right now, one of the limitations to the Phishpedia and brand identification system, is that it is only able to detect brands on which it has been given logos. Whenever the system will be able to identify that an email is trying to impersonate a brand, but the system does not know which brand, this could be used for classifying email as phishing.

In order to investigate the different behaviour, three metrics during the test are saved.

- Number of boxes found by logo detection model
- Average similarity scores
- Unique brands

The first metric is the number of boxes found by the RCN model Section 2.5 to contain possible logos. For every box, a top-four of logos with the highest similarity score is produced, since that is the way the system is designed. All of those logos are theoretically candidate logos. The average similarity score is the average over the similarity scores of the logos of all boxes (e.g. 3 boxes will result in 12 logos). Each of these logos is of a certain brand. The third metric will calculate the number of unique brands over all the candidate logos. In an ideal scenario, those metrics would be different for email samples of categories 1 and 3, since then we could use this to differentiate between those categories.

Table 6.8: Per sample category for test 2, number of samples and the number of correct predictions done by the Brand recognition system

Category nr.	Category description	Samples	Correct
1	No brand	181	177
2	Brand from predefined set	108	86
3	Brand not from predefined set	39	0

Table 6.9: Analysed statistics for test 2

Category	average similarity	nr of boxes	unique brands
1	0.770	2.205	5.675
2	0.821	2.2	2.44
3	0.772	2.474	5.868

Unfortunately, no real distinguishable behaviour can be observed. The average similarity scores of category 1 and category 3 are very similar. If a difference in average similarity score here is observed, one could argue for a sample with a low similarity score that it is more likely to be of either category 1 or 3. The number of boxes detected by the logo detection system is a little bit higher for category 3, but this might be due to a dataset bias since the dataset used for category 3 exist of more recent emails than the datasets used for categories 1 and 2. The number of unique brands proposed by the model is similar for samples belonging to categories 1 and 3. Therefore, also here no distinguishable behaviour can be observed.

Identifying brand outside the predefined training set It would be very convenient if the brand recognition system could handle brands on which it has not been trained on. Though, due to the inner workings of the brand recognition system, the system will never be able to classify the correct brand name when it encounters the logo of a brand it is not familiar with. On the other hand, a part of the brand recognition model can be used to compare two logos of any brand on similarity. As has been described in Section 6.1.1, the Siamese model can turn a logo into a feature vector representation, where similar logos have similar feature vectors. This could be used to compare logos found in an email and on a website. When the logo detection method identifies a logo on a website and a logo in an email, the Siamese model can translate both logos into a feature vector map. By calculating the cosine similarity between these two logos, a prediction can be made whether the logos are of the same brand. This functionality is used during the reverse image search.

The reverse image search module will try to identify brands outside the regular dataset on which the brand identification system has been trained. To test the performance of this system, two datasets have been created Section 5.2.4. Next to this, the logo detection service offered by Google Cloud Vision API is tested. For both the reverse image search test and the Google logo detection test, the two datasets consist of two types of images:

1. Logos of brand
2. No logo

Each logo has been labelled with the correct brand. The performance of the models can be evaluated using the following definitions:

1. **True Positive** - The model gets a picture with a logo of a brand, and predicts the brand correctly
2. **True Negative** - The model gets a random picture, and does not label the picture with a brand. A random picture can not be a computer-generated picture but is a picture taken with any sort of camera.
3. **False Positive** - The model gets a random picture, but predicts a label to the picture. Or, the model gets a picture of a logo but predicts the wrong brand to the logo.

4. **False Negative** - the model gets a logo of a brand, but does not give any brand to the logo.

As discussed in Section 6.1.1, the brand identification system compares the most similar image as presented by Google with the image under consideration. The similarity is calculated by taking the cosine similarity of the feature vector representation found by the brand identification system (Section 6.1.1). This similarity score needs a threshold on when the two images are deemed to be similar enough. The optimal value for this threshold has been determined by varying the threshold and comparing the outcome. In Figure 6.10 and Figure 6.11, the precision-recall curves are depicted for running on the two datasets described in Section 5.2.4. It was found that the optimal threshold for cosine similarity was similar to the threshold found in Section 6.2.1. This is expected behaviour since the reverse image search uses the system proposed there. Therefore, the threshold will be set to .835.

The Google logo detection service presents a confidence score with its prediction. A minimum threshold can be set to determine what the confidence score must be to accept the prediction as the truth. This threshold has been varied, and again for every threshold, the precision and recall have been calculated. These scores are plotted in Figure 6.10 and Figure 6.11.

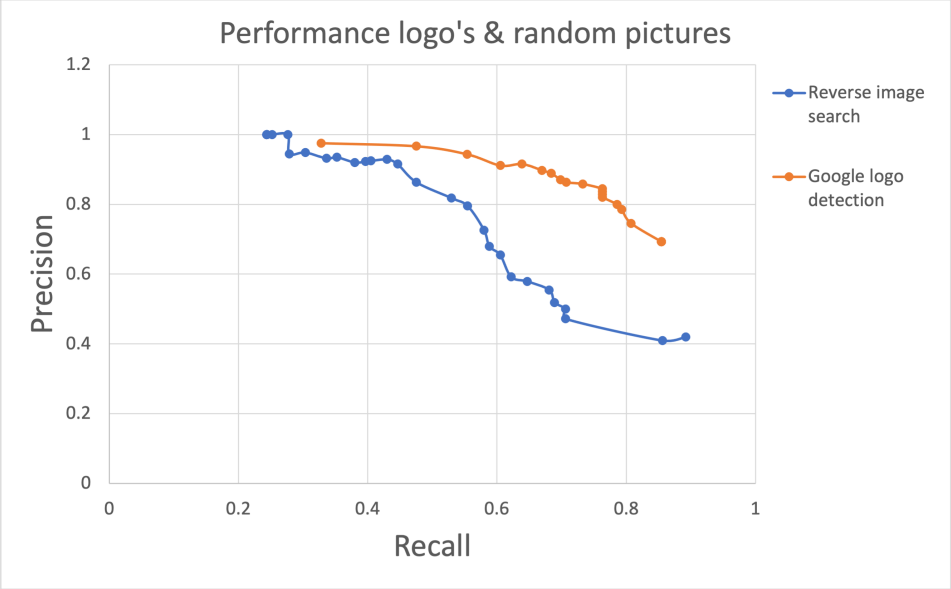


Figure 6.10: Performance of reverse image search on the dataset with logos and random pictures

The first dataset with pictures of logos and random pictures shows that the reverse image search validation method performs worse than the Google logo detection service. Though the results for the second dataset create a different view. For the second dataset, the reverse image search performs better. At this point, it is important to point out that the second dataset is more representative of the actual task where the reverse image search and/or Google logo detection system will be used for. The functionality where the reverse image will be used for is to be able to identify brands in boxes detected by the logo detection model. The first dataset uses very clear, fully centralised logo pictures for the ”positive” class and as ”negative” samples random pictures. The second dataset consists of actual boxes predicted by the logo detection model for both the positive and negative classes. For the rest of this research, the reverse image search technique will be used to identify brands outside the predefined dataset. This is chosen because of two decisions:

- The performance of the reverse image search technique is better than the Google logo detection service for the most representative test.
- The Reverse image search technique relates closer to the techniques used in this thesis than the Google logo detection service.

6.2.2. Feature analysis Fusion module

Each data sample has been translated into a feature vector, for a value for each feature of the Fusion module. The features have been described in Section 6.1.2. To understand the features from the Fusion

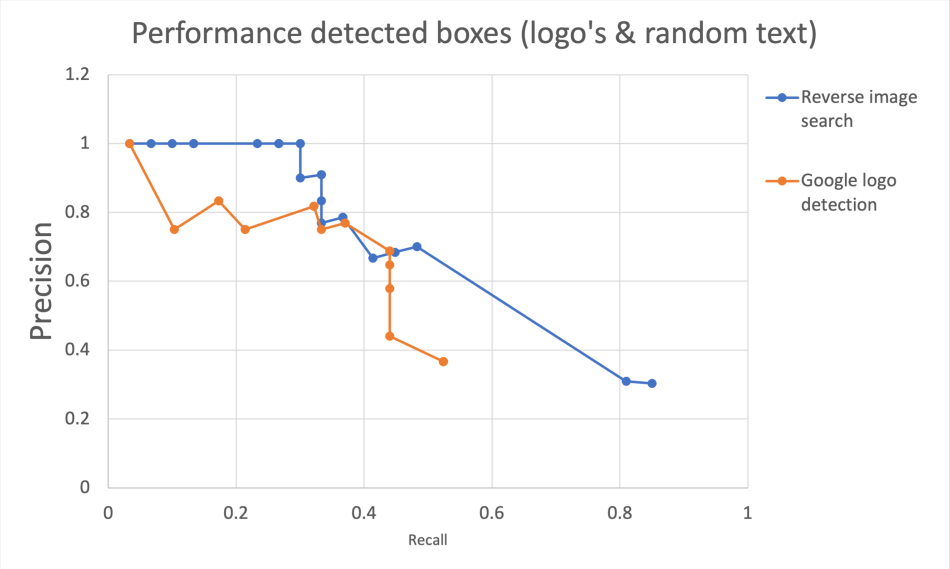


Figure 6.11: Performance of reverse image search on the dataset with logos and predicted boxes produced by the logo detection model

module better, some statistics on each feature are gathered, as discussed in Section 2.6.

General Statistics For convenience, the outcome of the general statistic analysis has been depicted in Chapter D. One of the inspected statistics is the number of a zero occurring for the feature on any samples. A binary or categorical feature can have a lot of ”0” occurrences, but for a continuous feature, many zeros might imply a malfunctioning feature. Luckily, there are no features which produce a zero for all samples. There are though, three continuous features with a lot of zero occurrences in both phish- and benign samples. These have been highlighted in Table 6.10. The ”nr of brands recognised in email” indicated whether a brand has been recognised in the email. Remember from Chapter 5 that the entire phishing dataset consisted of 1733 emails, and the benign dataset of 2412 emails. Therefore, for 11% of the phishing dataset and 15%, brands were detected in the emails. Since these percentages are quite similar, there is little threat that the dataset is biased. Furthermore, this feature on itself might not be an indicator for phishing, but it might give additional information to other features.

Interestingly enough, the ”nr of times logo in website is same as logo in email” has a higher zero occurrence rate than the ”nr of times brand in website is same as brand in email”. The former takes the feature vectors from any logos predicted by the logo prediction module and compares the logos between the website and the email. The latter, also takes the feature vectors from the logo detection module, but tries to identify a brand to the prediction box. Then, it compares the brands detected on the website and the email. Both features work with the same prediction boxes, and are not biased, or run after each other. That there are fewer zeros for the ”nr of times brand in website is same as brand in email”, than for the comparison of the logos directly, implies that is easier for the module to detect a brand directly than to find two logos to be similar. This is most likely due to the existence of the reverse image search. The brand identification does use the reverse image search, whereas the logo comparison does not use this.

Feature ranking The features of the Fusion module have been ranked according to Gini importance, as described in Section 2.6.2. The entire results of this ranking can be found in Table D.2. The actual Gini importance per feature has been depicted in Figure 6.12. Feature number 16 (at index 15), ”minimum days between registration of mail domain and website feature”, is the most important feature for the Fusion module. An interesting insight is obtained when removing this feature and recalculating the most important feature again. The results of this can be seen in Figure 6.13. The importance of every feature does not grow equally. This implies a correlation between the feature removed and the features left.

Table 6.10: Zero occurrences for three continuous features from the Fusion module.

feature	zero occurrence in phish	zero occurrence in benign
nr of brands recognised in email	1381	1816
nr of times brand in website is same as brand in email	1437	2031
nr of times logo in website is same as logo in email	1474	1398

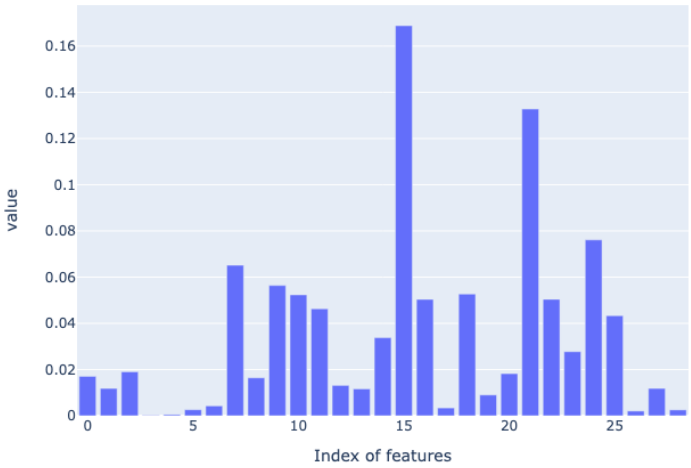


Figure 6.12: Feature importance for all features Fusion module

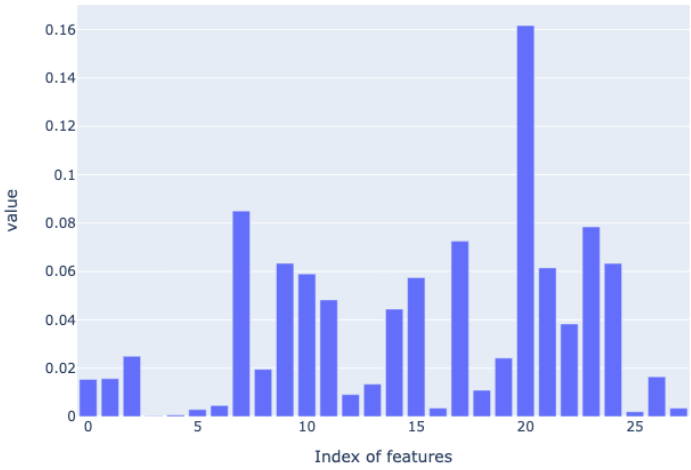


Figure 6.13: Feature importance with most important features removed for the Fusion module

The top-5 of the features are highlighted in Table 6.11. The most indicative feature is the ”minimum days between registration of mail domain and website feature”. When looking at the general statistics in Chapter D, it can be seen that this feature has a value of ”0” 182 times in the phishing dataset, and 1758 times in the benign dataset. This is a 10% and 73% respectively. This feature is ”0”, when the domain from where the phishing email is being sent is also listed as a URL in the email. For benign emails, this

Table 6.11: Top 5 best performing features from the Fusion module

Column names	Feature ranking
minimum days between registration of mail domain and website	28
nr of URLs in email	27
URL in email performs redirect	26
certificate type of domains of links in email	25
nr of times brand in website but different from email	24

can be often the case where a company sends an email, containing a link to their website (in e.g. the footer of the email). Apparently, phishing emails do not often do this. In general, phishing URLs tend to have a very short lifespan and are very new. Based on the experience of cyber security experts, it is known that attackers sometimes hijack the email server of legitimate companies. An attacker will then send an email from a domain which has been registered a decent time ago. Though, the phishing email will try and trick the user to visit a website which is very new. The number of days between registration of the email domain and the domain of the phishing page is then possibly very large, but at least not 0. Though do note, that this feature can be easily circumvented by an attacker. The attacker could add a link to the domain from which the attacker sends its attack to the phishing email. This would result in a zero value for this feature, which is apparently more legitimate behaviour.

A second very distinctive feature is the number of URLs in an email. Phishing emails contain, on average, 3.45 URLs. Benign emails average 12.57 (with one anomaly containing an astonishing 696 URLs). A hypothesis on this difference is that benign marketing emails contain a lot of content, whereas phishing emails tend to be more simplistic. Though, this hypothesis can not be proven with the data at hand.

”URL in email performs redirect feature” is a binary feature. This feature is 1 if any URL in the email performs a redirect. Do note that this feature might very well be a little biased since there are more URLs in benign emails, to begin with.

”certificate type of domains of links in email feature”, shows that the set of certificate types used in benign emails differs from that of phishing emails. The count per set type have been shown in Table 6.12. The phishing emails have way more samples where no certificate is found. Also, the benign emails have a lot more organisation validation certificates present.

The 5th interesting feature is the ”nr of times brand in website but different from email feature”. This feature has an average of 1.6 for the phish class, but 6.8 for the benign class. Though, when correcting for the total number of URLs in both classes, this feature has 0.25 ”brands in website but different from email” for the phish class, and 0.54 for the benign class. This means that the benign email contains, on average more emails which do not match the brand in the email. When no brand is recognised in the email, but a brand has been recognised in the website, this still counts as a difference. One more analysis has been done on the sub-category of the features. In Table 6.13, the average ranking per sub-category has been shown. Note that a high ranking is better than a low ranking. Features using domain registration dates stand out firmly above all other sub-categories. With an average of 21.6 over 5 features, this sub-category is very well-performing. The email brand sub-category performs the worst with an average of 6.22 for 9 features.

Impact on performance for removing features As can be seen from the performance graphs in Figure 6.14, Figure 6.15 and Figure 6.16, the accuracy of the module decreases after the removal of 5 features. The precision and recall decrease slightly less, but also start decreasing after the removal of the 5th feature. It is important to see that there is not one feature which distinguishes the benign from the phish samples, but several features working together in the prediction. This makes the model more resilient against changes in emails, as well as adversarial attacks.

Table 6.12: Certificate set types present in the different classes of the dataset

Certificate type set	Benign	Phish
None	74	781
EV	135	68
OV	793	221
OV + EV	20	53
DV	686	551
DV + EV	19	47
DV + OV	629	66
DV + OV + EV	195	37

Table 6.13: Average feature ranking per sub-category

Sub-categories:	Count	Average rank
Email brand	9	6.22
Website brand	6	11.5
Email logo	1	15
Website logo	1	15
Website layout	3	8.67
Website URL redirect	2	20
Certificate	6	11.83
website domain	7	16.71
Email 'from:' domain	11	15.91
email content	2	16.5
domain registration date	5	21.6
DMARC	1	19
spf	1	17

6.2.3. Performance Fusion module

The performance of the Fusion module has been evaluated on classification results, and time performance. As discussed in Section 6.1.3, three models have been picked to be trained on the features extracted from the emails. These models are Random Forest, XGBoost and Support Vector Machines.

Classification results First, the classification results of the Fusion module will be analysed.

All features used The classification results of the Fusion module have been depicted in Table 6.14. As can be seen, the XGBoost algorithm shows significantly better results compared to the other two models. The Support Vector Machine has an exceptionally high false negative rate. Overall, the model shows fairly good results. Especially when taking into consideration that the results are similar to the results obtained by the classification based on the structure features. These features were

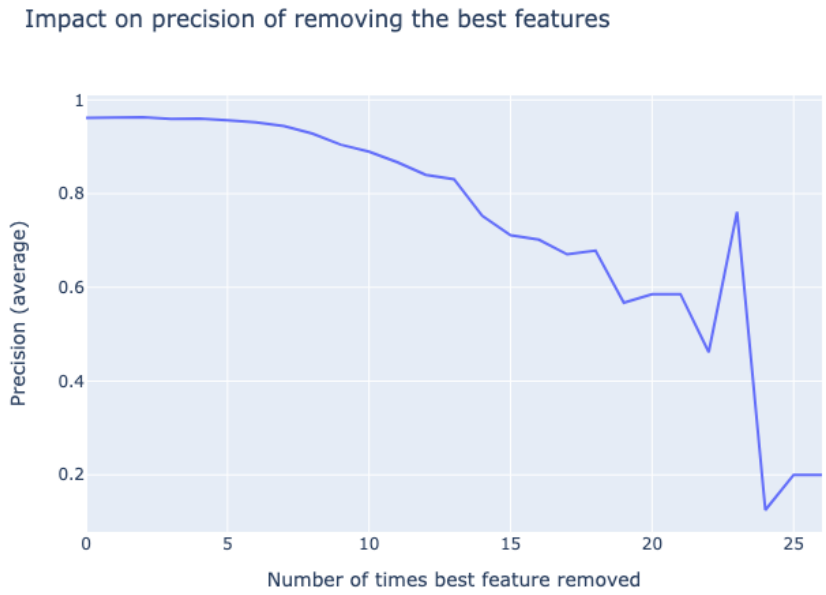


Figure 6.14: Precision development when removing features for the Fusion module. Experiment has been done 5 times, the results are averaged.

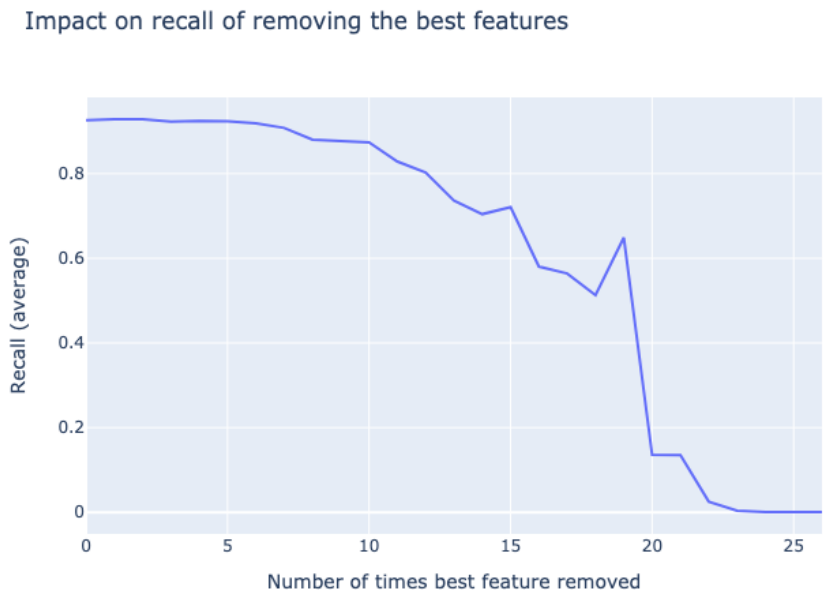


Figure 6.15: Recall development when removing features for the Fusion module. Experiment has been done 5 times, the results are averaged.

selected from state-of-the-art papers. The performance of the structure module will be discussed in Section 7.2.1.

In Table 6.15, the number of misclassified samples per dataset have been shown. For both the Random Forest and XGBoost algorithm, the number of misclassified samples is equally divided. Though by ratio, the eye phish is fairly often misclassified as being benign. This is an important dataset since the Eye phish and Eye benign datasets are the only two datasets from one organisation including both

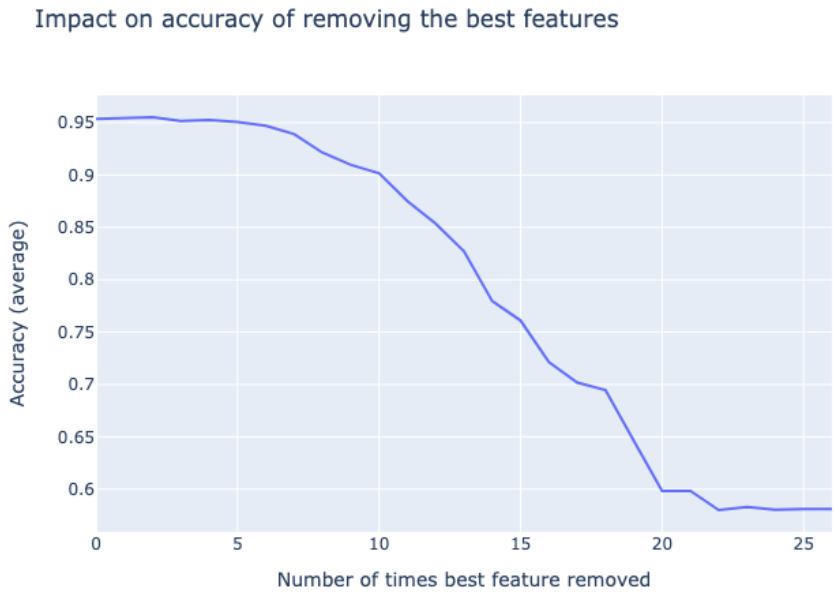


Figure 6.16: Accuracy development when removing features for the Fusion module. Experiment has been done 5 times, the results are averaged.

phish and benign emails. The emails from Eye phish are being misclassified by XGBoost in almost 16% of the samples. For Eye benign, this is 3%, Personal 2% and Nazario 4%. The misclassification of Eye phish is thus exceptionally high.

Table 6.14: Performance per machine learning model for the Fusion module

Metric	XGBoost classifier	Random Forest Classifier	SVM classifier
True positives	1642	1619	693
True negatives	2356	2349	2329
False positives	56	63	83
False negatives	91	114	1040
Precision	0.967	0.963	0.893
Recall	0.947	0.934	0.4
Accuracy	0.965	0.957	0.729
False positive ratio	0.023	0.026	0.034
Average area under curve ROC	0.962	0.954	0.683

Brand recognition features When looking at the generated features in Table 6.2, a clear distinction between the features can be made. Those making use of the brand recognition system, and those who don't. The features using the brand recognition system are the features with a low-level technique stating it uses Logo detection or brand recognition in Table 6.2. For analysis purposes, it is interesting to see what the classification performance of the model does when either of the two categories is left out. These results have been shown in Table 6.16 and Table 6.17. Using only features

Table 6.15: Misclassified samples per machine learning model for the Fusion module

Dataset	XGBoost misclassified	Random Forest misclassified	SVM misclassified	total used
Eye phish	28	39	146	178
Eye benign	17	19	4	591
Personal	39	43	79	1821
Nazario	63	83	894	1555

without brand recognition (Table 6.16), shows superior results to a classifier trained only on brand recognition features. Though, the brand recognition features perform fairly well. Combining the brand recognition features with the non-brand recognition features, result in a reduction of 25 misclassified samples compared to the non-brand recognition features.

Table 6.16: Performance of the Fusion module without using the brand recognition features

Metric	XGBoost classifier	Random Forest Classifier	SVM classifier
True positives	1634	1625	693
True negatives	2339	2350	2329
False positives	73	62	83
False negatives	99	108	1040
Precision	0.957	0.963	0.893
Recall	0.943	0.938	0.4
Accuracy	0.959	0.959	0.729
False positive ratio	0.03	0.026	0.034
Average area under curve ROC	0.956	0.956	0.683

Hyper parameter optimisation By trying different parameter settings for each machine learning model, the performance of the Fusion module can be increased. The classification results for optimal settings found after 100 iterations of random search are depicted in Table 6.19. The optimal settings per module were:

XGBoost

- Learning rate - .69
- Maximum depth of a tree used in the XGBoost - 68
- Number of trees used in the XGBoost - 300
- Minimum sum of weight needed in a child node - 1
- Minimum loss reduction needed before making a split in a tree (gamma) - 0

Random Forest

- Number of trees - 70

Table 6.17: Performance of the Fusion module while using only brand recognition features

Metric	XGBoost classifier	Random Forest Classifier	SVM classifier
True positives	1448	1470	1401
True negatives	2124	2106	1837
False positives	288	306	575
False negatives	285	263	332
Precision	0.834	0.828	0.709
Recall	0.836	0.848	0.808
Accuracy	0.862	0.863	0.781
False positive ratio	0.119	0.127	0.238
Average area under curve ROC	0.858	0.861	0.785

Table 6.18: Overview of the performance when splitting the features into different machine learning models.

Metric	All features	No brand recognition	Only brand recognition
True positives	1642	1634	1448
True negatives	2356	2339	2124
False positives	56	73	288
False negatives	91	99	285
Precision	**0.967**	**0.957**	**0.834**
Recall	**0.947**	**0.943**	**0.836**
Accuracy	0.965	0.959	0.862
False positive ratio	0.023	0.03	0.119
Average area under curve ROC	0.962	0.956	0.858
Execution time per sample (s)	95.75	25.57	70.18

- The number of features to consider when looking for the best split - 'sqrt'
- Maximum depth of a tree - 79
- Minimum number of samples required at split - 5
- Minimum number of samples required at leaf - 1

Support vector machine

- Regularization parameter - 18.5
- Kernel type - 'rbf'
- Kernel coefficient - .039

Though, the optimal settings do not improve on the results found before in Table 6.14.

Table 6.19: Performance of the machine learning models for the Fusion module with the best performing parameter settings

statistic	XGBoost classifier	Random Forest Classifier	SVM classifier
True positives	1648	1637	529
True negatives	2347	2349	2400
False positives	65	63	12
False negatives	85	96	1204
Precision	0.962	0.963	0.978
Recall	0.951	0.945	0.305
Accuracy	0.964	0.962	0.707
False positive ratio	0.027	0.026	0.005
Average area under curve ROC	0.962	0.959	0.65

Adjusting weights to classes As introduced in Section 2.2.5, experiments have been run to see results for adjusting weights for the classification classes. In this research, the aim is to produce fewer false positives. Therefore, the weights applied were focused on the positive class. The negative class kept a weight of 1, while the weight of the positive class (phishing), was varied from 1 to .01. The experiment was run for an XGBoost machine learning algorithm with default settings. For every weight, the precision-recall has been calculated. This has been plotted in Figure 6.17. Setting the weight (cost of misclassification) for the positive class to .09 resulted in a precision of .984 and a recall of .921. With a false positive of 26 and a false negative of 137, the conclusion can be drawn that the false positives have indeed decreased compared to a non-weighted classifier. When the weight would be further reduced, the number of false positives can decrease even more, but with a rising number of false negatives.

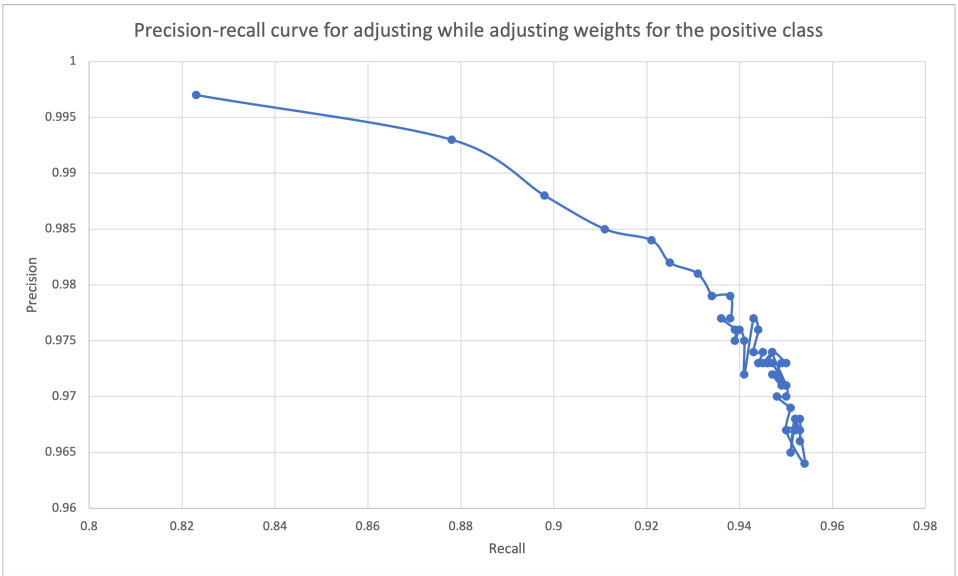


Figure 6.17: Precision recall curve for Fusion module while adjusting the weights for the positive classification class

Time performance Fusion module The experiments were run on a Dell XPS 15 7590 with Intel Core i7-9750H CPU @ 2.60 GHz and 32.0GB RAM. The timing performance of the Fusion module is far from perfect. The average time (Figure 6.18), shows that on average, the Fusion module takes more

than 90 seconds to extract all the features needed to perform the prediction. The phishing detection system proposed here performs the phishing detection after an email has been successfully received by an email server. If the server would first have to perform a prediction before handing the email over to the receiver, this would introduce a delay between the time when the email is received by the server and the time the email becomes available to the intended receiver. For most businesses and individuals, a delay in the receiving of an email of 90 seconds might not even be noticed. Though, computation time is generally expensive. Therefore, it is desired to keep the execution time of the Fusion module as low as possible. To gain a better feeling of the time performance from the module, the samples can be divided into several ranges (Figure 6.19). For 21% of the samples, the Fusion module takes less than 30 seconds to execute. 31% of the samples fall within the 30-60 second range. Another 18% takes between 60 and 90 seconds, and the other 30% takes longer than 90% seconds.

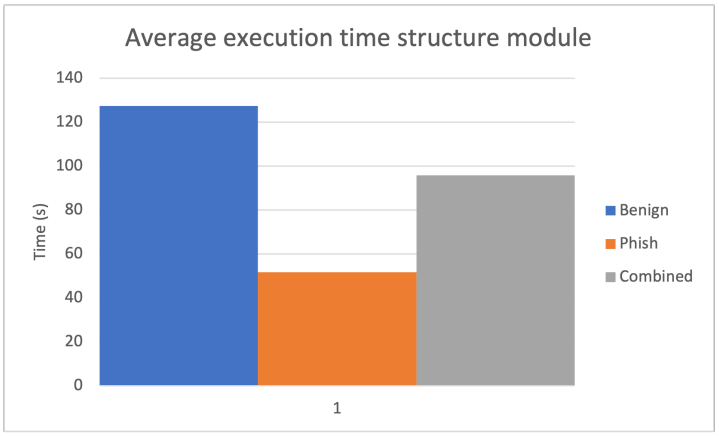


Figure 6.18: Average execution time of the Fusion module for Phish, benign and combined

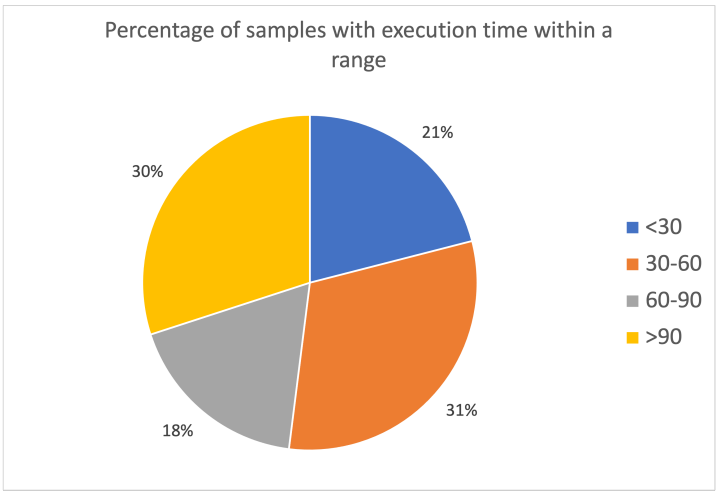


Figure 6.19: Distribution of samples with an execution time within a time interval for the Fusion module.

Time per sub-part of the Fusion module What takes the most time for the execution of the Fusion module? To answer this question, the time has been split to ratio in Figure 6.20. A large part of the execution time goes to taking screenshots of the email and each website in it (32.3%). Taking a screenshot of a website takes a lot of time. This is mostly due to the waiting time which is introduced to make sure the screenshot does not fail. Also, many times a website takes a long time to load in full or doesn't load at all. Though, the system will still wait until the timeout is triggered. By fine-tuning the waiting times introduced, as well as finding a more optimal timeout, this execution time can be reduced. Another improvement would be to use several browser instances and take the screenshots of

several websites at once. This will reduce the time needed to extract all features from one email, but it does not decrease the computation time needed. Another big chunk of the execution time (18.2%), is the detection of logos and identification of brands by the system as introduced in Section 6.1.1. Again, this can be reduced by building the module to be able to work threaded. The creation of screenshots, the logo detection, brand identification and the reverse image search query are all needed only for the logo detection and brand identification system. Though, these parts take up 73.3% of the execution time of the Fusion module.

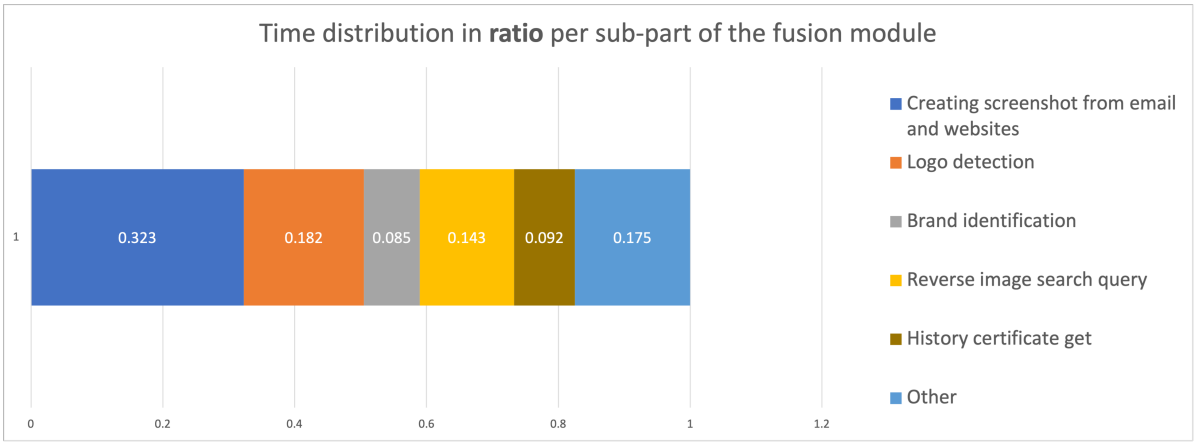


Figure 6.20: The ratio of total execution time for each sub-part in the Fusion module.

Table 6.20: Feature ranking of features which use either the brand recognition or the logo detection system

Nr	Low-level	Name	Ranking
19	Selenium, brand recognition system	nr of times brand in website but different from email	24
23	Selenium, brand recognition system	nr of websites with brand recognised	22
21	Selenium, logo detection system	nr of times logo in website is same as logo in email	15
3	Selenium, logo detection system	average nr logos detected by logo detection per website	13
14	Selenium, brand recognition system, sockets/crt.sh	from email address is owned by brand in email	10
2	Selenium, logo detection system	at least one website with no logos	9
28	Selenium, brand recognition system, sockets/crt.sh	URLs of unknown brands exist and brand recognised	8
20	Selenium, brand recognition system	nr of times brand in website is same as brand in email	7
7	Selenium, brand recognition system	brand of website is mentioned in email	6
18	Selenium, brand recognition system	nr of brands recognised in email	5
29	Selenium, brand recognition system, logo detection system	website without logo but brand in email	4
6	Selenium, brand recognition system	brand is recognised in email	3
27	Selenium, brand recognition system, sockets/crt.sh	URLs of unknown brands exist	2
5	Selenium, brand recognition system, list_of_social_media_brands	brand in email is social media	1

Running on a website Since the creation of screenshots for websites takes a lot of time, it is expected that emails with more URLs take more time. Indeed this turns out to be true. When looking at Figure 6.21, it is clear that the execution time grows linearly with the number of URLs present in an email. This results in bad time performance when the number of URLs is excessively high. For example, in the benign database, there is one email containing 696 URLs. The extraction of features from this email took 4625 seconds, which is over 77 minutes. Even though this is a very extreme example, this still shows that if there were to be a trend in the design of emails where the number of URLs would increase, this system must speed up. This also opens an attack vector where the mailbox of a company could become victim to a DDoS attack where attackers would send emails with many URLs.

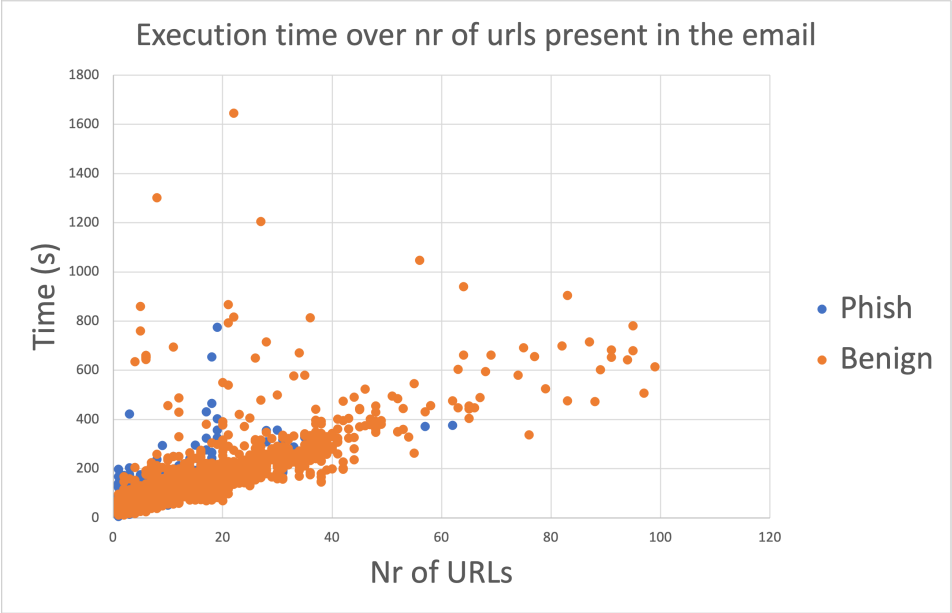


Figure 6.21: The number of URLs present in an email shows a correlation with the execution time of the Fusion module on the email. Do note that several anomalies have been removed for the visibility of the graph.

7

Holistic detection system

The phishing detection landscape has seen many detection tools come and go. As has been discussed in Chapter 4, there are multiple detection mechanisms focusing on different parts of a phishing attack. How great would it be, that these systems would not operate individually, but work together and enforce each other? The holistic detection system provides just that. In this chapter, a very clean and simple way of making several phishing detection systems work together to perform a single classification task is introduced.

7.1. Methodology

In this section, the methodology used to create a holistic detection system has been described. First, the general architecture is explained. Then, the methodology to build the modules for the detection of the structure of the email is made clear. Lastly, the methodology for the module performing phishing detection on websites is explained.

7.1.1. General architecture of the holistic detection system

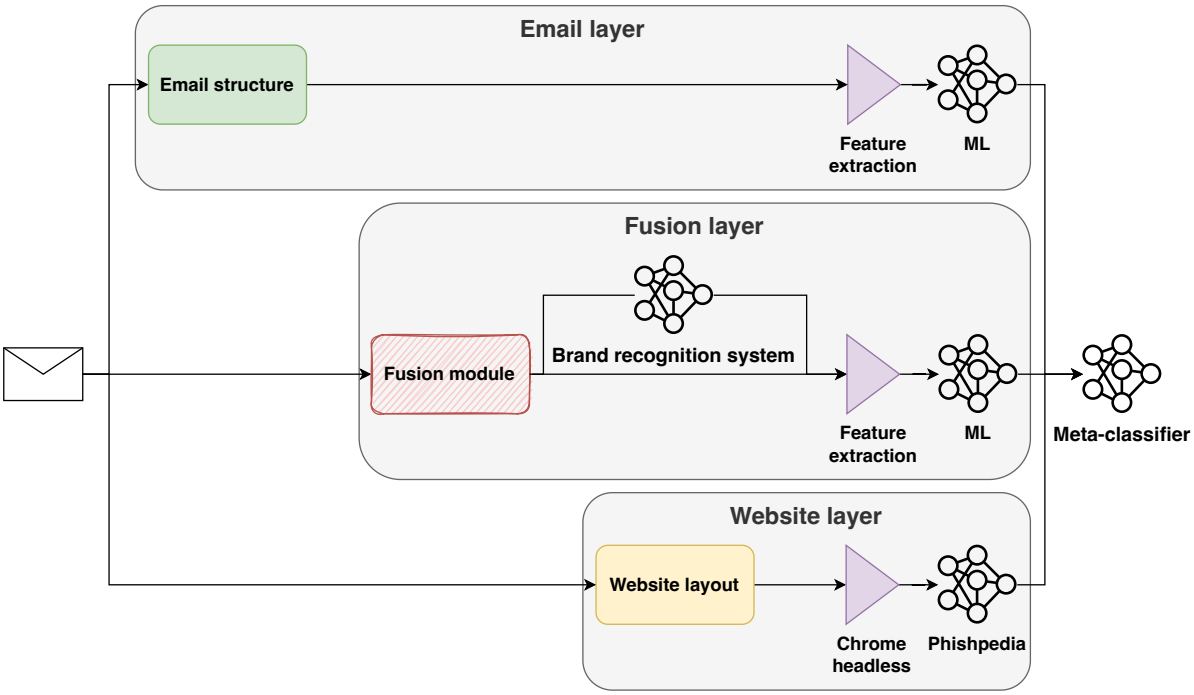


Figure 7.1: The architectural overview for the holistic system

As discussed in Chapter 4, this research is the first to create a phishing detection model based on the combination of the email and website stages. In Chapter 6, the design for the fusion module has been explained. This module already combines stages using respectively email and websites for detection. Though, from Figure 4.3 it is clear that there are phishing detection mechanisms which can perform phishing detection on the separate stages. When it would be possible to combine several phishing detection mechanisms, a phishing attack might be better predicted. The possibility to combine several stages has been investigated in this thesis. Such a system could combine phishing detection mechanisms which perform the detection on different stages in a phishing attack. This would create a **holistic phishing detection system**, which performs phishing detection on multiple stages in the phishing attack chain. For this, the best system which performs phishing detection solely on an email, and the best system which performs phishing detection on a website have been chosen. In combination with the new fusion module designed, this would create a whole new type of phishing detection system. When building a system which can use several different phishing detection mechanisms, the system would also be resilient over time. The email and phishing landscape is continuously evolving. Since the phishing landscape is evolving so fast, phishing detection possibilities also keep evolving. In this dynamic field, a phishing detection system becomes redundant very fast. By being able to combine different detection systems into a single classification system, old systems can be replaced with new, state of the art techniques.

After an extensive literature study, many systems which qualify were found. By comparing performance on accuracy between the systems, two systems were selected. The Email structure module (Section 7.1.2) and the website module (Section 7.1.3).

Different modules The final classifier has been built out of three different phishing detection models. The first model performs phishing detection solely on the email structure. The design of this model has been explained in Section 7.1.2. The second model is the fusion model, as discussed in Chapter 6. The third model detects phishing websites and is discussed in Section 7.1.3. Those three models will each be built into modules in the holistic detection system.

Meta-classifier After the selection and fabrication of phishing detection models for different stages, the different modules should be combined to build a single classifier. As discussed in Section 2.2.3, meta-learning is a technique where a machine learning classifier is built on top of other classifiers. For any sample **X**, the different detection modules will each produce a regression score on how likely the module deems the sample to be phishing or not. This regression score will then be used as a feature for the meta classifier. The meta-classifier has been built from Random Forest, XGBoost, and Support Vector Machine. The results have been compared to pick the classifier with the best performance [12][40].

First, the regression score for every sample in the dataset should be obtained for every module. For every module, the best performing machine learning algorithm with the correct settings was chosen, as found during previous performance tests. Then, k-fold cross-validation is run on every machine learning model (Section 2.3.4). First, the training set was used to train the module. Then, the test set was run on the module. Instead of using the test set in every iteration of the k-fold to test the performance of the model, the regression scores produced when running the test set were saved for each sample. After the k-fold has fully run, this will result in a predicted value for each sample for the structure, website and fusion module.

Per email sample, we have thus obtained three features for the holistic detection system. I.e. the regression prediction for the structure, fusion and website modules. With these regression scores, the meta classifier has been trained and tested. During this training, the machine learning model has been trained on the training part of the dataset. The classification of the meta classifier will be tested with the original label of the sample. E.g., Sample 1 is a phishing email. The structure module gives a regression score of .78, the website module .65 and the fusion module .95. The meta classifier labels the sample with a binary classification and labels the email as ”1”, i.e. phishing. In this case, the meta classifier is thus correct. The validation of the meta-classifier has been done by using k-fold validation on the dataset.

7.1.2. Email structure module

The email structure module is one of the three modules of the holistic phishing detection system. This module is able to give a prediction score whether an email is phishing email, based on structural elements of that email. The structure module uses a selection of handcrafted features extracted from email samples.

Feature selection The structure part of the email is so complex, that the training of a self-learning machine learning model to interpret the structure of an email, would probably require a very vast number of training samples. Unfortunately, such a vast number is very hard to come by. Next to the training data problem, the structure of different email service providers differs. Changes in this structure might bias the network. In order to overcome these difficulties, the email structure can be preprocessed to create a regular machine learning model. One way of preprocessing is to define a list of features for the email, and translate an email into a feature vector. A feature holds information on certain parts of the structure of the email. Information types can be, amongst others, the existence of some element, the length of certain parts of the email structure or the number of appearances of elements. The features used in this research were based on several papers. Lee et al. [34] crafted a list of features from a wide variety of previously conducted research. The phishing prediction model built from those features had a recall of .9933. The validation of the performance of those features has been described in their paper. Next to the proper validation, the performance of the system built is excellent. Therefore, the features were included in the list of features for this research. The next-best performing module which performs phishing detection on the structure of an email is proposed by Smadi et al. [49]. The phishing detection performance of the system created by Smadi et al. has an accuracy of 98.63% and a recall of 99.07%. Though, [34] did not name their paper as a source for crafting the features selected. Therefore, this research will use the best performing features from [49], which are not already included in the list from [34].

Every email can now be translated into a feature vector with a value for every feature. Those feature vectors can serve as input for a machine learning model. First though, since the code was not publicly available, the extraction of every feature has been rebuild.

Machine learning models To optimise the performance of the structure module, several machine learning models have been created, and the performance compared. After literature study, three machine learning models were selected. (see also Section 2.2.4).

- Random Forest
- XGBoost
- Support Vector Machine

Each of the three models have been trained and tested on the same dataset as mentioned in Chapter 5. The validation has been done by using k-fold cross validation. (Section 2.3.4).

7.1.3. Website module

To perform phishing detection based on the website(s) which are in an email, it is desired to build a website prediction module. For the website module, the best performing algorithm was the Phishpedia model [35]. The working of this model has been explained in Section 2.5. The Phishpedia model as proposed by Lin et al. only works on a screenshot. What the website module thus does, is visiting every URL in an email and take a screenshot. The screenshot and corresponding URL have been fed to the Phishpedia model. The model tries to recognise the brand in the screenshot and verify whether the domain is owned by the company. The Phishpedia model simply holds a list of valid domains for every brand. Whenever a brand has been detected and the domain is not in the list of domains for that brand, the sample will be labelled phishing. Luckily, the code for the Phishpedia model has been publicly released. Though, many adjustments were needed to get the model working, and modify it in such a way it could cooperate with the different modules in the holistic detection system.

Determining the prediction score The Phishpedia model as proposed by Lin et al.[35] performs a prediction on one website. Though, the website module should produce a prediction score for an email, which might contain several websites at once. Also, the Phishpedia model is a binary classification model. It predicts the similarity between a logo under consideration and any other logo. If that similarity is above a certain threshold, the model predicts the website as being phishing or not. The website module, on the other hand, would preferably be a regression model. Lastly, the Phishpedia model only considers one possible logo to be a brand and discards all others. To adjust the Phishpedia model into a regression model, the following steps are taken.

1. Determine a phishing regressions score for a logo on a website
2. Determine a regression score for a website with multiple logos, each producing a regressed prediction.
3. Combine the regression scores produced for every website in an email, to a single regression score for the email.

Regression score for a logo The regression scores used in this thesis have a value between 0 and 1. A regression score for an email close to 0 means that the email is predicted to be a benign email, whereas a value close to 1 means that the model predicts the email to be phishing. The regression formula takes into account all variables discussed above and produces a regression score between 0 and 1. The desired functionality of the formula is that a box with a high similarity and phishing prediction -1, is most likely not phishing, and thus the regression score produces a value between 0-0.5. How close the value is to zero, should depend on how confident the model is in whether the brand it detected is correct. When the brand identification system produces a high similarity score with a logo from a brand, the model is very certain that the prediction box is a logo from that brand. Thus, in that case, the regression score should be close to zero. When the model is very certain that it has recognised a brand, but the domain is not associated with that brand, the chance of phishing is higher, and thus the regression score should be close to 1. When the model is less certain, the value should be around 0.5. See Table 7.1 for the high level desired outcome of the regression formula.

Table 7.1: Desired outcome of the regression formula

Similarity Score	Phishing prediction	Regression score
HIGH	-1	~0
HIGH	1	~1
LOW	1	~0.5

Based on the requirements for the regression score as set above, the following variables have been used in the regression formula.

- **Similarity score prediction box and brand** - The website module produces a regression score for every possible logo as detected by the logo detection system from Section 6.1.1. Such a possible logo is named a prediction box. The brand identification model compares each of these possible logos to any of the logos it has in its database. The similarity is the cosine similarity produced by the identification model. This is a score between 0-1 on how similar the two images are.
- **Phishing prediction** - When the Phishpedia model detects a brand on a website, it checks whether the domain where the logo has been detected is from that brand. If it is not, the domain is deemed to be phishing. If the domain is owned by that brand, the domain is thus not phishing. The check performed to verify whether the domain is owned by a brand, is done by whitelisting. Remember from Section 2.5, that Phishpedia holds a fixed set of brands that it can handle. For each brand, one or multiple domains are listed to perform a check on. This validation is used in the regression score. For the brand with the highest similarity score to a prediction box, the domain(s) from the brand is/are checked with the domain on which the prediction box has been found. If they are equal, the box is given a -1 phishing prediction, indicating no phishing. If the domain is not a domain associated with the brand found in the prediction box, the phishing prediction is 1.

- **Maximum similarity threshold before rounding off** - The similarity score given to every prediction box will be given a threshold to determine when it is accepted as similar enough. To the same extent, it might be the case that after a certain similarity, we are almost completely certain that two images are similar. For convenience, the similarity can be rounded to one. This might have an effect on the prediction given by the system.

To start simple, the regression formula should look something like this:

$$p_b(\alpha) = 0.5 + \varphi * f(\alpha) \tag{7.1}$$

where $p(b)$ is the regression score for prediction box b with similarity score α, φ the phishing prediction and $f(\alpha)$ a function of the similarity score α. $f(\alpha)$ should be close to 0.5 for a high similarity score. For example, $\frac{1}{1-\alpha}$ will produce a higher value when α is larger.

$$g(\alpha) = \frac{1}{1-\alpha} \tag{7.2}$$

The outcome of $f(\alpha)$ should be between 0 and 0.5. Therefore, a scalar is used to normalise the outcome of $g(\alpha)$. This scalar is dependent on the largest possible value for $g(\alpha)$. This maximum score is reached when α is close to 1. As mentioned before, after a certain similarity the model might be certain "enough". We are therefore not interested in the similarity between 0.95 - 1. Though, that range sees a huge rise in value for $g(\alpha)$. Therefore, a cap is introduced to cap the maximum similarity before normalisation. When setting this cap (τ) to 0.9, this means that any sample with a similarity score above 0.9 will result in a regression score of either 0 or 1. The normalisation value is thus dependent on the threshold τ:

$$n(\tau) = \frac{0.5}{g(\tau)} \tag{7.3}$$

Resulting in the final formula:

$$p_b(\alpha) = 0.5 + \varphi * \frac{0.5}{g(\tau)} * g(\alpha) \tag{7.4}$$

When $g(\alpha)$ is squared or replaced with g^3, the effect of a higher similarity score for a regression closer to 0 or 1 is dramatised. The following three regression formulas will thus be used and tested on performance:

$$p(\alpha) = 0.5 + \varphi * \frac{0.5}{\frac{1}{1-\tau}} * \frac{1}{1-\alpha} \tag{7.5}$$

$$p(\alpha) = 0.5 + \varphi * \frac{0.5}{(\frac{1}{1-\tau})^2} * (\frac{1}{1-\alpha})^2 \tag{7.6}$$

$$p(\alpha) = 0.5 + \varphi * \frac{0.5}{(\frac{1}{1-\tau})^3} * (\frac{1}{1-\alpha})^3 \tag{7.7}$$

Regression score for a website The model can now produce a regression score for every prediction box in a website. The regression score for the website will be based on these scores. On a single website, there can be logos from multiple brands. For example, a website might contain logos from social media or references from other companies. Whenever any logo is matched to the domain by the phishing detection sufficiently, the website should be labelled as non-phishing. For example, the brand recognition system recognises Amazon, Facebook and Twitter on the legitimate Amazon website. The predicted boxes for the Amazon logo will produce a low regression score, but the prediction boxes for the Facebook and Twitter logos will produce a high regression score. It might even be the case, that the regression scores for Facebook and Twitter are closer to 1 than the regression score for Amazon is to 0. Thus, whenever a regression score below 0.5 is seen on a website, the regression score for the website will be the minimum of all regression scores. But, when no regression score for a prediction box below 0.5 is detected, the regression score for the website is set to the maximum regression score. The maximum regression score is taken since the model is as certain as its highest similarity when it has been established that the page is not-benign.

Regression score for an email An email can, and often does, contain multiple domains. One trick

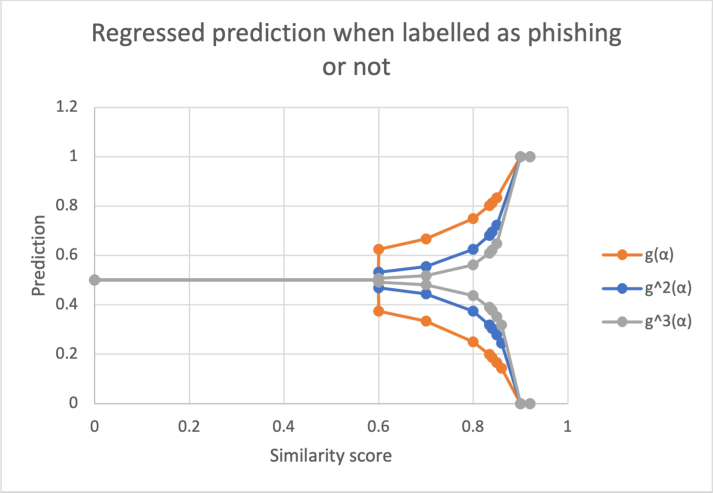


Figure 7.2: Regression score for different formulas for $g(\alpha)$

used by phishers to obfuscate their phishing email is to encapsulate several links to benign websites in their phishing mail, as well as a link to their phishing page. This makes it harder for phishing detection systems to flag the email as phishing, since it also includes benign websites. Though, there has to be only one malicious link in the email, to detect the email as a phishing email. Using this, we can state than whenever one website in the entire email has a high likelihood of being phishing, the email can be flagged as such. Therefore, the regression score of the email is the maximum of the regression scores of all websites.

7.2. Results

In this results section, the results of the tests performed on the individual models are first described. Then, the results of the holistic model are given

7.2.1. Structure module validation

The validation of the Structure module has been done by first performing a feature analysis. After this has been done, the classification results are shown and discussed. Then, the time performance of the structure module is analysed.

Feature analysis Structure module The feature analysis for the Structure module has been split into general statistics, feature ranking and impact on performance for removing features, as described in Section 2.6.

General statistics To understand the features from the Structure module better, some statistics on each feature is gathered, as discussed in Section 2.6. The outcome of this general statistic analysis can be seen in Chapter A. Remember from Chapter 5, that the dataset used for validation consists of 1733 phishing emails and 2412 benign emails. First, features are investigated which show a high zero occurrence rate per class. ”Boundary id is decimal”, and ”Boundary id starts with equal symbol” both show a high zero occurrence for benign, but that occurrence is lower for phishing emails. This indicates that the boundary id is different for benign emails compared to phishing ones. ”number of ’in-reply-to’ entities”, ”re in subject” and ”id part of message id is decimal” and several others have a high zero occurrence for both phishing and benign. Comparing these 5 features, the first two features have a higher feature ranking (out of 67) of 57 and 54 respectively, whereas the other three rank 29, 26 and 19. This is as expected, since a feature with many zero occurrences for either class, is less indicative of phishing than features which show a clear distinction between either class.

Feature ranking The ranking of the features for a Random Forest classifier has been inspected. The feature ranking is listed per feature in Table A.2. Also, the results of the calculation can be seen in

Figure 7.3. Feature number 27 (at index 26), ”index of charset in first section”, is the most important feature for the Structure module. An interesting insight is obtained when removing this feature and recalculating the most important feature again. The results of this can be seen in Figure 7.4.

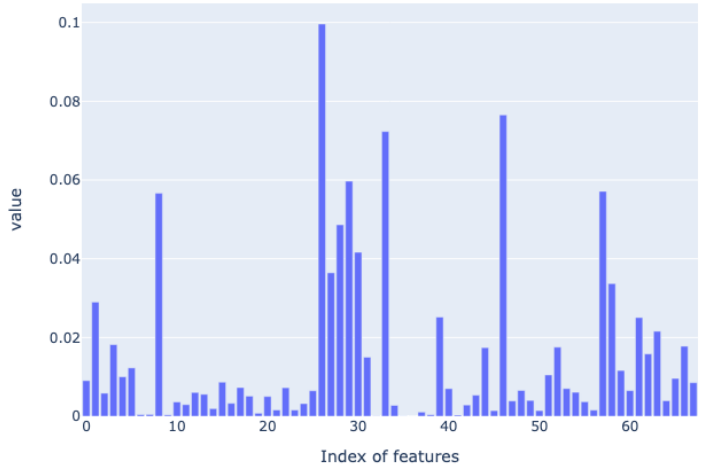


Figure 7.3: Feature importance for all features for the Structure module

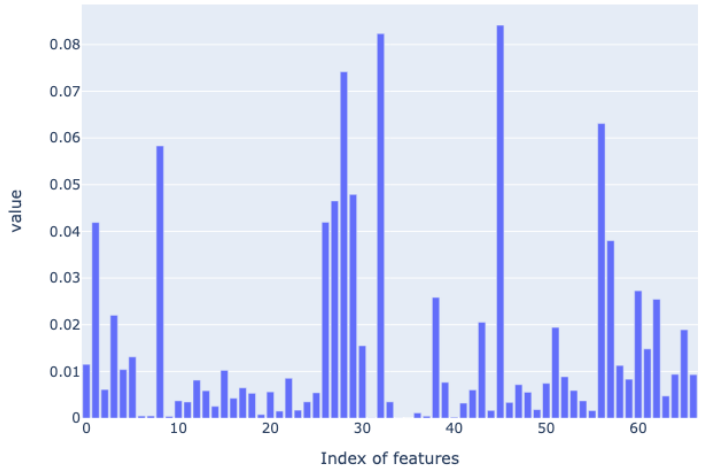


Figure 7.4: Feature importance with most important feature removed for the Structure module

Impact on performance for removing features In Figure 7.5, Figure 7.6 and Figure 7.7, the precision, recall and accuracy (respectively) per number of features removed have been visualised. Remember from Section 2.6, that the best performing feature in terms of Gini importance is determent, and removed. The results show that the performance in terms of the three metrics stays fairly stable, even when many features are removed. An important conclusion here is that there are multiple features fairly good in classification. This implies that when a dataset would grow and more anomalies are present in the dataset, the module will have multiple features to detect the anomaly. The precision has a strange increase after feature 47, but this comes at a very steep fall in recall.

The top two best performing features show some distinguished behaviour in Table A.1. Their zero occurrence is low for both phish and benign emails. Also, the frequency of the most occurring option for that feature in the phish and benign datasets have clear different values. Though, the third most

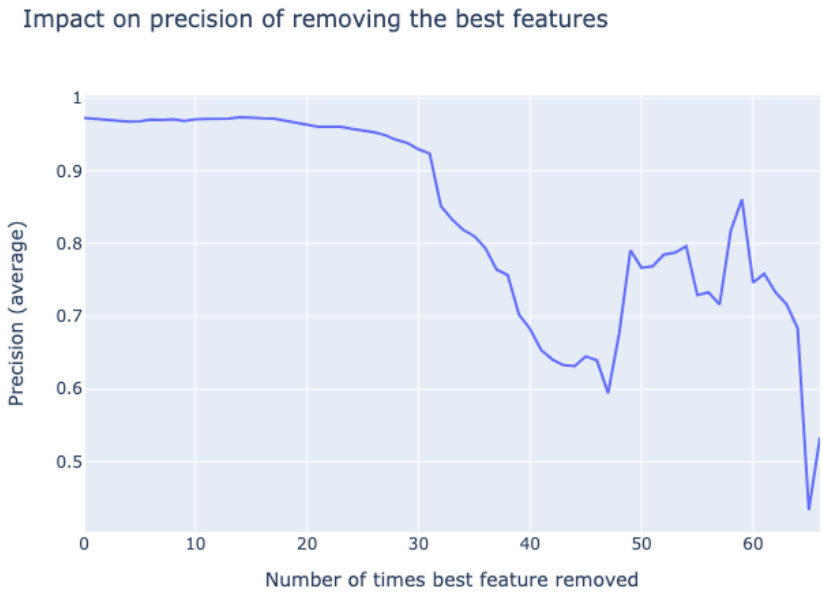


Figure 7.5: Precision development when removing features for the Structure module. The experiment has been done 5 times, the results are averaged.

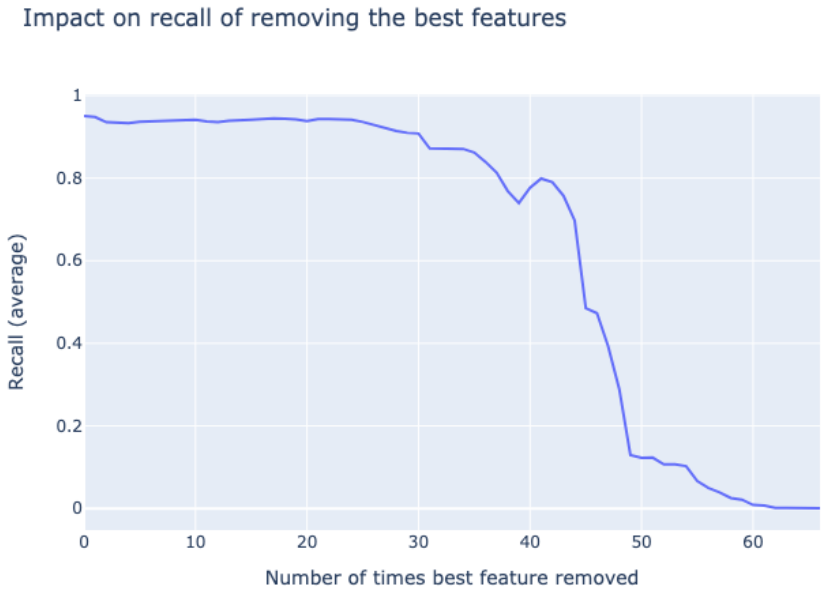


Figure 7.6: Recall development when removing features for the Structure module. The Experiment has been done 5 times, the results are averaged.

important feature shows that the general statistics are clearly not enough to analyse the features. The frequency of the most occurring option per feature of the ”number of standard headers” is very low for both phish and benign emails, indicating that this number varies much. This feature would not stand out as important when looking at the general statistics only.

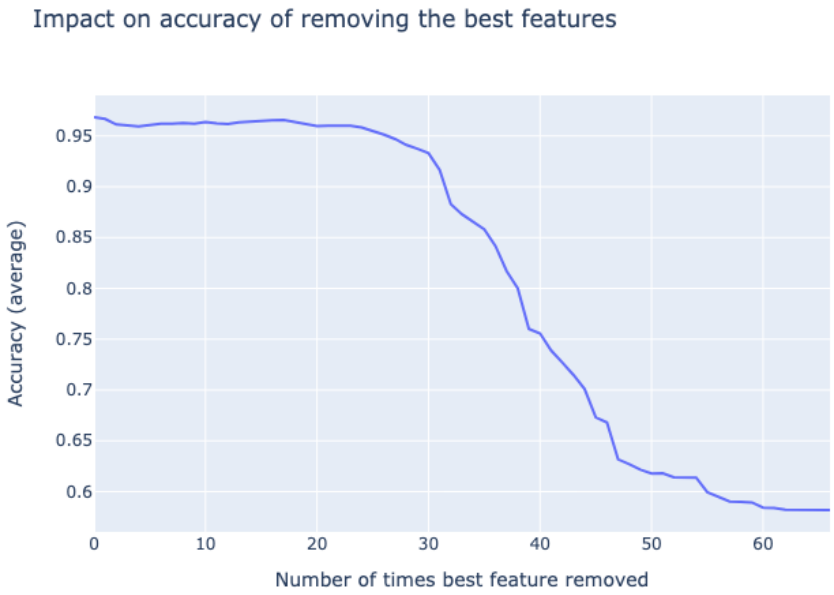


Figure 7.7: Accuracy development when removing features for the Structure module. The Experiment has been done 5 times, the results are averaged.

Classification results Structure module The performance of the three machine learning models is evaluated using several metrics. The results are depicted in Table 7.2. Unfortunately, the recall which was found by [34] and [49] could not be matched. Though, this might very well be due to the different datasets used in this research compared to theirs.

Table 7.2: Performance per machine learning model for the Structure module

Metric	XGBoost classifier	Random Forest Classifier	SVM classifier
True positives	1657	1648	1281
True negatives	2373	2366	1904
False positives	39	46	508
False negatives	76	85	452
Precision	0.977	0.973	0.716
Recall	0.956	0.951	0.739
Accuracy	0.972	0.968	0.768
False positive ratio	0.016	0.019	0.211
Average area under curve ROC	0.97	0.966	0.764

The XGBoost classifier is slightly better than the Random Forest classifier, but the Support Vector Machine comes nowhere near the other two. Since the XGBoost algorithm performs better on all metrics, this classifier will be picked to build the meta classifier with.

The misclassified samples shown in Table 7.3, give very interesting insights into the behaviour of the Structure module, compared to the Fusion module. The Structure module performs better in precision and recall values compared to the Fusion module. Though, the number of misclassified samples from

Table 7.3: Misclassified samples per machine learning model for each dataset for the Structure module

Dataset	XGBoost	Random Forest	SVM	Total used
Eye phish	46	53	75	178
Eye benign	8	9	109	591
Personal	31	37	399	1821
Nazario	30	32	377	1555

the 'Eye phish' is higher for the Structure module than for the Fusion module (See Table 6.15). One hypothesis here is that the structure of the emails between the 'Eye phish' and the 'Eye benign' datasets is more similar than the behavioural difference analysed in the Fusion module. This hypothesis though, can not be confirmed.

Hyper parameter optimisation As done for the Fusion module, hyper parameter optimisation has been performed. Using this technique, the classification performance found in Table 7.4 has been obtained. The optimal settings used for this table were:

XGBoost

- Learning rate - .58
- Maximum depth of a tree used in the XGBoost - 77
- Number of trees used in the XGBoost - 105
- Minimum sum of weight needed in a child node - 1
- Minimum loss reduction needed before making a split in a tree (gamma) - 0.222

Random Forest

- Number of trees - 227
- The number of features to consider when looking for the best split - 'auto'
- Maximum depth of a tree - 110
- Minimum number of samples required at split - 2
- Minimum number of samples required at leaf - 1

Support vector machine

- Regularization parameter - .05
- Kernel type - 'rbf'
- Kernel coefficient - 6.0

Interestingly enough, the hyper parameter optimisation does not always result in overall improved performance. The XGBoost and Random Forest have increased performance compared to Table 7.2, but the support vector machine is not necessarily better. It has a precision of 1, but the recall is worse.

Adjusting weights to classes As introduced in Section 2.2.5, experiments have been run to see results for adjusting weights for the classification classes. In this research, the aim is to produce fewer false positives. Therefore, the weights applied were focused on the positive class. The negative class kept a weight of 1, while the weight of the positive class (phishing), was varied from 1 to .01. The experiment was run for an XGBoost machine learning algorithm with default settings. For every weight, the precision recall has been calculated. This has been plotted in Figure 7.8. Setting the weight (cost of misclassification) for the positive class to .11 resulted in a precision of .982 and a recall of .925. With 29 false positives and 130 false negatives, the conclusion can be drawn that the false positives have indeed decreased compared to a non-weighted classifier. When the weight would be further reduced, the number of false positives can decrease even more, but with a rising cost of false negatives.

Table 7.4: Performance of the machine learning models for the Structure module with the best performing parameter settings

Statistic	XGBoost classifier	Random Forest Classifier	SVM classifier
True positives	1658	1660	285
True negatives	2373	2370	2412
False positives	39	42	0
False negatives	75	73	1448
Precision	0.977	0.975	1
Recall	0.957	0.958	0.164
Accuracy	0.972	0.972	0.651
False positive ratio	0.016	0.017	0
Average area under curve ROC	0.97	0.97	0.582

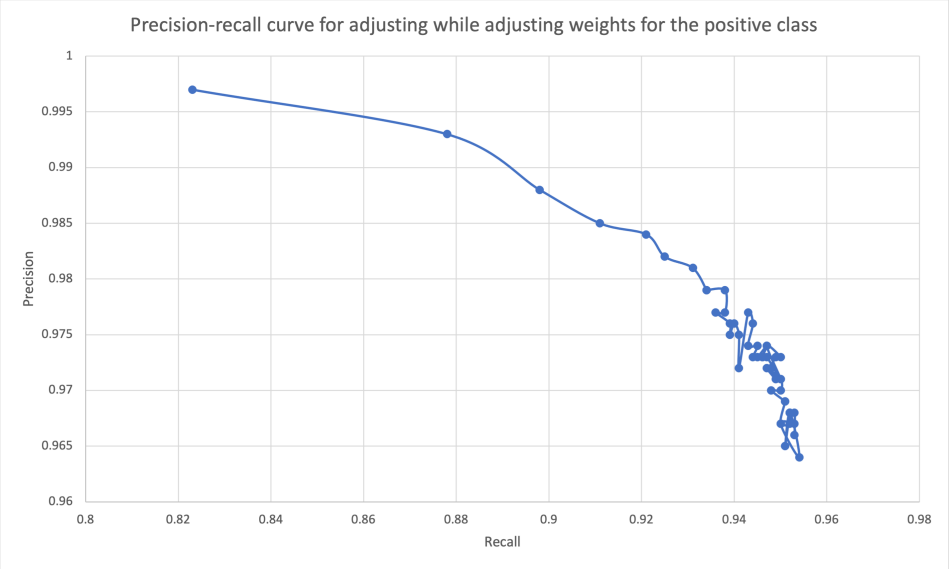


Figure 7.8: Precision recall curve for Structure module while adjusting the weights for the positive classification class

Time Performance Structure module The experiments were run on a Dell XPS 15 7590 with Intel Core i7-9750H CPU @ 2.60 GHz and 32.0GB RAM. The average time for the Structure module can be split into benign and phishing emails. This has been depicted in Figure 7.9. The Structure module works very fast. The Structure module was able to extract the features of 76% in less than .5 seconds, Figure 7.10. Only 6% of the samples took more than 1 second to complete.

7.2.2. Website module validation

For the validation of the Website module, the regular dataset has been used from Chapter 5. Using this dataset, the Website module produces a regression score for every email sample. To evaluate the performance, every email with a regression score above 0.5 was labelled to be phishing, and every sample with 0.5 or lower is labelled as a benign email. Three variables were adjusted and the impact on the performance was evaluated.

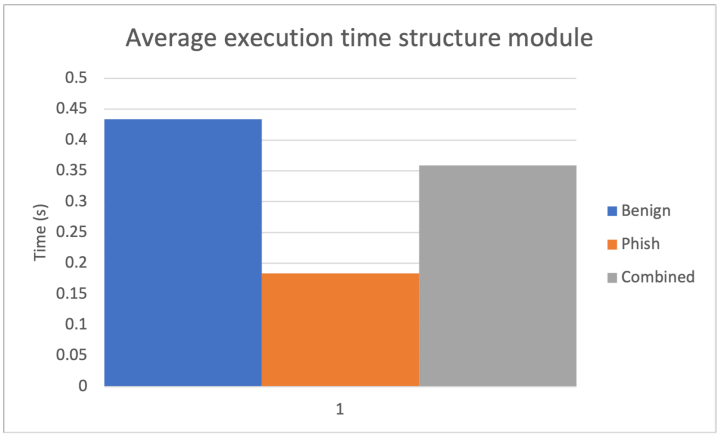


Figure 7.9: Average time per email class for the Structure module to create its features

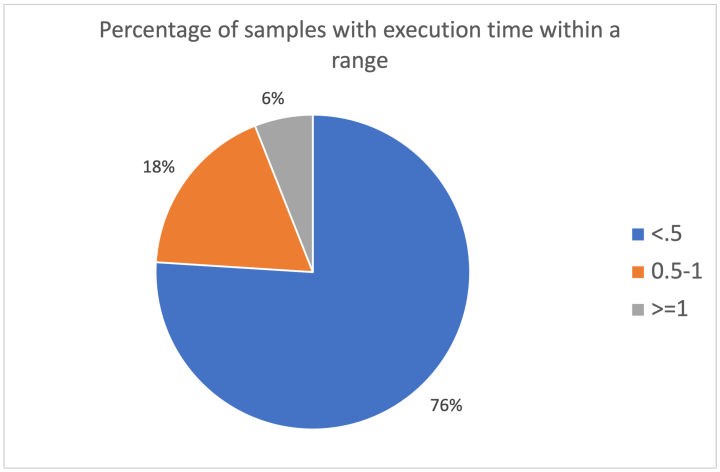


Figure 7.10: Distribution of samples with an execution time within a time interval for the Structure module.

Highest possible similarity As mentioned in Section 7.1.3, the similarity score is capped to a certain value when calculating the regression score. In order to investigate the impact on the performance of this metric, the threshold is adjusted and the results analysed. In Table 7.5, it is shown that varying the threshold actually has no impact on the labelling performance of the model. This can be expected since the threshold only impacts how close the regression score is to one or zero, but does not impact on which side of 0.5 the regression score actually is. This is due to the design of the threshold, discussed in Section 7.1.3. Though, the Website module will be used as a regression model in the meta classifier. The regression score given by the model might actually matter for the meta classifier. Therefore, the impact of the thresholds on the prediction loss is also investigated. The prediction loss is defined as the difference between the actual label of the sample (1 for phishing, 0 for non-phishing), and the regression score. A total lower prediction loss for all samples is therefore better. As can be seen in Table 7.5, the total prediction loss is lowest for a threshold below .8. Though, previous tests on the brand identification system have shown that the optimal threshold for minimum similarity is .835. Therefore, picking a cap value lower than the minimum threshold could create inaccurate results. A threshold of .85 for the maximum similarity threshold is set for further experiments. This means that when a similarity of .85 occurs, the regression score floors to either 1 or 0, depending on the phishing prediction variable.

Minimum similarity threshold Next to the maximum similarity threshold cap before rounding, a minimum similarity threshold will also be set. The minimum similarity threshold will result, in theory, is a model which is more certain when predicting. With a higher threshold, more samples will be classified with a regression score of 0.5, indicating that the model is not confident in labelling the sample. As can be seen in Table 7.6, a varying minimum similarity threshold has a significant impact on

Table 7.5: Performance for different maximum similarity threshold

Similarity cap	0.75	0.8	0.85	0.9
True positives	1343	1343	1343	1343
True negatives	1723	1723	1723	1723
False positives	689	689	689	689
False negatives	390	390	390	390
Precision	0.661	0.661	0.661	0.661
Recall	0.775	0.775	0.775	0.775
Accuracy	0.74	0.74	0.74	0.74
False Positive ratio	0.286	0.286	0.286	0.286
Average area under curve ROC	0.745	0.745	0.745	0.745
Total prediction loss	1079	1079	1096.103	1294.647

the performance of the module. Judging from the precision-recall values from the table, a good choice for a similarity threshold is, again, .835. The prediction loss depicted in Figure 7.11, shows the same result.

Table 7.6: Performance for varying similarity thresholds

Minimum similarity threshold	0.7	0.75	0.8	0.81	0.82	0.83	0.84	0.85
true positives	1708	1703	1529	1479	1413	1369	1252	1174
true negatives	134	136	1008	1289	1501	1694	1753	1761
False positives	2278	2276	1404	1123	911	718	659	651
false negatives	25	30	204	254	320	364	481	559
Prediction loss	2158.158	2160.233	1567.912	1363.557	1232.019	1090.684	1140.932	1210
precision	0.43	0.43	0.52	0.57	0.61	0.66	0.66	0.64
recall	0.99	0.98	0.88	0.85	0.81	0.79	0.72	0.68

Different regression score formulas Having set the similarity threshold and cap, the different regression formulas can be investigated. Though, the labelling performance of the different formulas is equal. This is due to the design of the regression function. The effect of $g(\alpha)$ from eq. (7.4) on the regression score is minimised to determine the extent to how fast the regression scores go to either 0 or 1, but will never switch the prediction from phish to non-phish or vice versa. Though, there is a difference in prediction loss. Judging from the total prediction loss, taking the linear $g(\alpha)$ will minimise the prediction loss.

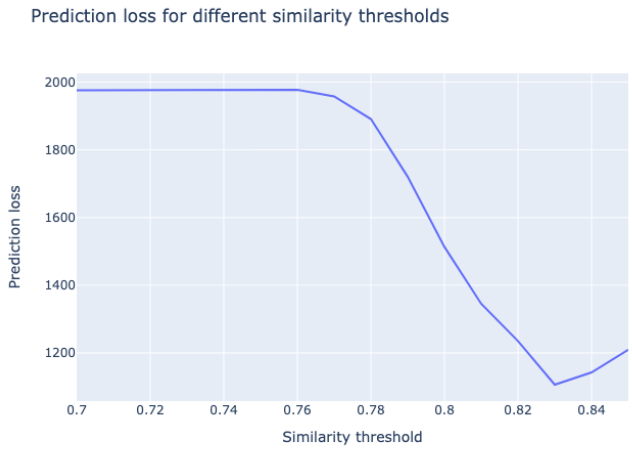


Figure 7.11: Prediction loss for different similarity thresholds. Run with a similarity cap of .85.

Table 7.7: Results for using different regression score formulas.

Statistic	Linear	Quadratic	Square
True positives	1343	1343	1343
True negatives	1723	1723	1723
False positives	689	689	689
False negatives	390	390	390
Precision	0.661	0.661	0.661
Recall	0.775	0.775	0.775
Accuracy	0.74	0.74	0.74
False positive ratio	0.286	0.286	0.286
Average area under curve ROC	0.745	0.745	0.745
Total prediction loss	1087.850	1096.103	1103.800

Table 7.8: Misclassified samples for the Website module per dataset.

Dataset	Misclassified	Total used
Eye phish	65	178
Eye benign	116	591
Personal	573	1821
Nazario	325	1555

Classification results Website module The final performance in the individual validation of the Website module of the model is depicted in the 'Linear' column of Table 7.7. This Website module has the following settings:

- **Highest possible similarity before rounding** - .85
- **Minimum similarity threshold** - .835
- **Regression formula** - linear

Table 7.9: Top 10 brands producing false positives.

Brand	False positives
facebook	219
microsoft	138
linkedin corporation	70
twitter	64
youtube	45
instagram	43
google	27
dhl	17
wetransfer	13
spotify	9
ziggo	7

This module, however, has fairly bad performance when compared to the other modules from Section 7.2.1 and Section 6.2.3. A hypothesis for the bad behaviour of this model is that it is not always illegitimate for the logo of brand **x** to appear on the website of brand **y**. For example, assume the website of a brand which links to their Instagram on their homepage. In that case, the Website module will detect an Instagram logo on a website which is not owned by Instagram and will label the site as phishing.

To inspect this behaviour, the top 10 brands producing false positives have been listed in Table 7.9. Here the reader can see that a lot of social media brands produce false positives, this could be in line with the previous suggestion. Interestingly, the Microsoft logo also produces a lot of false positives. The logos associated with Microsoft also include logos for Microsoft products such as OneDrive, Teams and Office365. It could be that these logos appear on different websites. Another option for false positives is that the domains sent were actually from Microsoft, but were not listed in the whitelisted pages for the Microsoft brand. This exposes the vulnerability of using such a list.

The number of false negatives is also high. This means that phishing emails were either not targeted on a brand known to the Website module, or the brand was just not recognised.

Time performance Website module The experiments were run on a Dell XPS 15 7590 with Intel Core i7-9750H CPU @ 2.60 GHz and 32.0GB RAM.

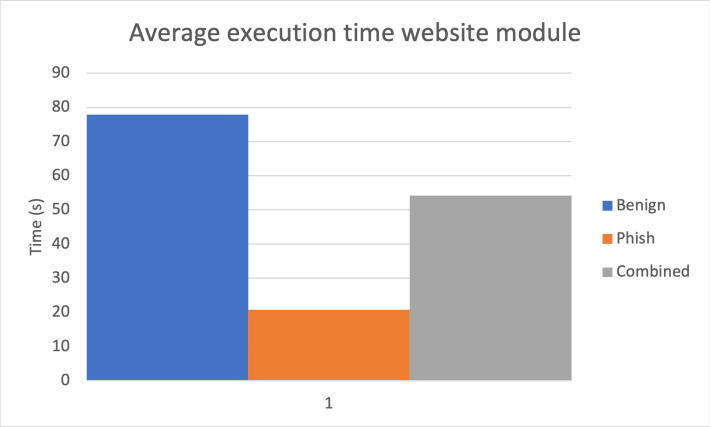


Figure 7.12: Average time per email class for the Website module to make a prediction

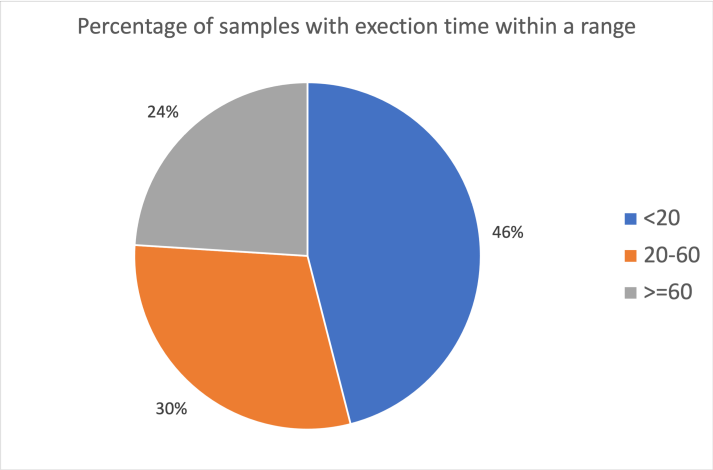


Figure 7.13: Distribution of samples with an execution time within a time interval for the Website module

The Website module, as is the fusion module, is way faster for phishing emails than it is for benign emails. As discussed in Section 6.2.3 as well, this is due to the difference in the number of URLs which each class holds on average. The Website module needs to make a prediction for every URL in the email. Since the benign emails hold more URLs, this takes longer. It is, however, interesting to see that almost 50% of the samples take under 20 seconds to be evaluated. When this system would be used on every email coming into an organisation, this is a fair delay.

7.2.3. Holistic system validation

To evaluate and validate the holistic system, first, a feature analysis has been conducted. Secondly, the classification results are shown for three different machine learning models. Then, the time performance is analysed. A qualitative analysis has been performed which is described lastly.

Feature analysis The regression scores from each of the detection modules serve as a feature of the holistic system. When investigating the feature importance using Gini importance for a Random Forest, the structure module is slightly more important than the fusion module. The Website module is clearly the least important feature, as can be seen in Figure 7.14. This could also be expected since the individual classification performance of the website module was inferior to the performance of the other two modules.

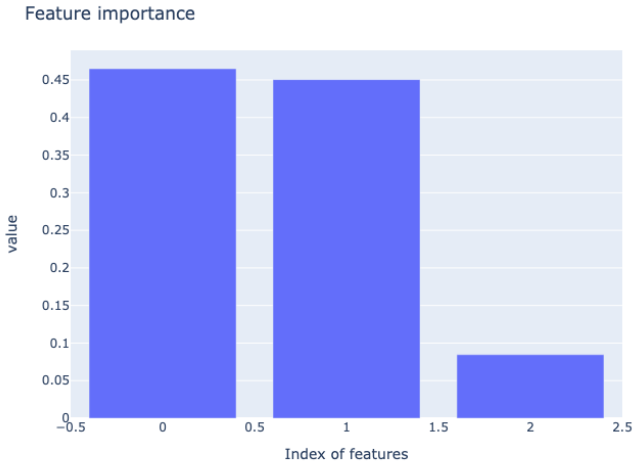


Figure 7.14: The feature importance for the meta classifier, where each feature is the regression score of one of the modules. From left to right: structure module, fusion module and website module

Classification results By combining three different detection modules, a single meta classifier has been built with a **better classification performance than any of the three modules individually**. This meta classifier is called the holistic system. For the individual models, the XGBoost algorithm outperformed the others. The performance of the Support Vector Machine came nowhere near the other two models. Though, for the meta classifier, Support Vector Machine shows the best results, at least when looking at precision. For recall, the XGBoost algorithm performs better. When looking at the misclassified samples per dataset, each algorithm shows its own behaviour. Remember from Section 7.1.2, that the features used for the Structure module were selected from state-of-the-art systems. If the features were implemented correctly, this would thus imply that the Holistic system could have better classification capabilities than the current state of the art techniques.

Table 7.10: Performance per machine learning model for the holistic system

Statistic	XGBoost classifier	Random Forest Classifier	SVM classifier
True positives	1694	1692	1688
True negatives	2375	2385	2386
False positives	37	27	26
False negatives	39	41	45
Precision	0.979	0.984	0.985
Recall	0.977	0.976	0.974
Accuracy	0.982	0.984	0.983
False positive ratio	0.015	0.011	0.011
Average area under curve ROC	0.981	0.983	0.982

Table 7.11: Misclassified samples per machine learning model for each dataset for the holistic system

Dataset	XGBoost misclassified	Random Forest misclassified	SVM misclassified	total used
Eye phish	27	31	33	178
Eye benign	9	7	8	591
Personal	28	20	18	1821
Nazario	12	10	12	1555

Hyper parameter optimisation As done for the fusion module and structure module, hyper parameter optimisation has been performed. This resulted in optimal settings for the three different machine learning models:

XGBoost

- Learning rate - .15
- Maximum depth of a tree used in the XGBoost - 30
- Number of trees used in the XGBoost - 161
- Minimum sum of weight needed in a child node - 1
- Minimum loss reduction needed before making a split in a tree (gamma) - 1

Random Forest

- Number of trees - 227
- The number of features to consider when looking for the best split - 'auto'
- Maximum depth of a tree - 110
- Minimum number of samples required at split - 2
- Minimum number of samples required at leaf - 1

Support vector machine

- Regularization parameter - .05
- Kernel type - 'rbf'
- Kernel coefficient - 6.0

Even though the model aimed to reach a high accuracy, the accuracy found during the hyper parameter optimisation was lower than the accuracy obtained for default settings (Table 7.10). Though, precision scores were high for the newly found configurations, with fewer false positives. This could be useful when fewer false positives are desired.

Table 7.12: Performance of the machine learning models for the holistic system with the best performing parameter settings

Statistic	XGBoost classifier	Random Forest Classifier	SVM classifier
True positives	1670	1671	1668
True negatives	2390	2389	2392
False positives	22	23	20
False negatives	63	62	65
Precision	0.987	0.986	0.988
Recall	0.964	0.964	0.962
Accuracy	0.979	0.979	0.979
False positive ratio	0.009	0.01	0.008
Average area under curve ROC	0.977	0.977	0.977

Adjusting weights to classes To reduce false positives for the model, weighting of classes has been applied. Remember from Section 7.1.1, that the holistic module consists of a meta classifier, build on the regression score of three different machine learning algorithms. The weighting of the classes can be done in 3 different scenarios.

1. Adjust the weights on the individual machine learning models of the structure and fusion modules. (See Figure 7.8 and Figure 6.17 respectively.) The meta classifier is trained on the regression scores provided by the structure and fusion module. In this scenario, the regression scores provided by the two modules will be different compared to non-weighting.

2. Adjust the weights on the meta classifier. The regression scores as provided by the structure and fusion modules will stay the same. Though, the meta classifier, which is built from an XGBoost machine learning model, will bias the negative class, reducing the number of false positives.

3. Adjust the weights on both the individual learning models, as well as the meta classifier. In this case, the structure, fusion and meta classifier machine learning models will bias to the negative class.

The adjusting of the weights has been done for a Support Vector Machine algorithm with default settings. In this research, the aim is to produce fewer false positives. Therefore, the weights applied were focused on the positive class. The negative class kept a weight of 1, while the weight of the positive class (phishing), was varied from 1 to .01. Support Vector Machine model has been chosen based on performances from Table 7.10.

Table 7.13: Performance when weighting classes. The columns represent for which machine learning models the weighting has been applied.

Statistic	Structure & fusion	Holistic	Structure, fusion & holistic
True positives	1677	1610	1661
True negatives	2393	2407	2400
False positives	19	5	12
False negatives	56	123	72
Precision	0.989	0.997	0.993
Recall	0.968	0.929	0.958
Accuracy	0.982	0.969	0.98
False Positive ratio	0.008	0.002	0.005
Average area under curve ROC	0.98	0.963	0.977

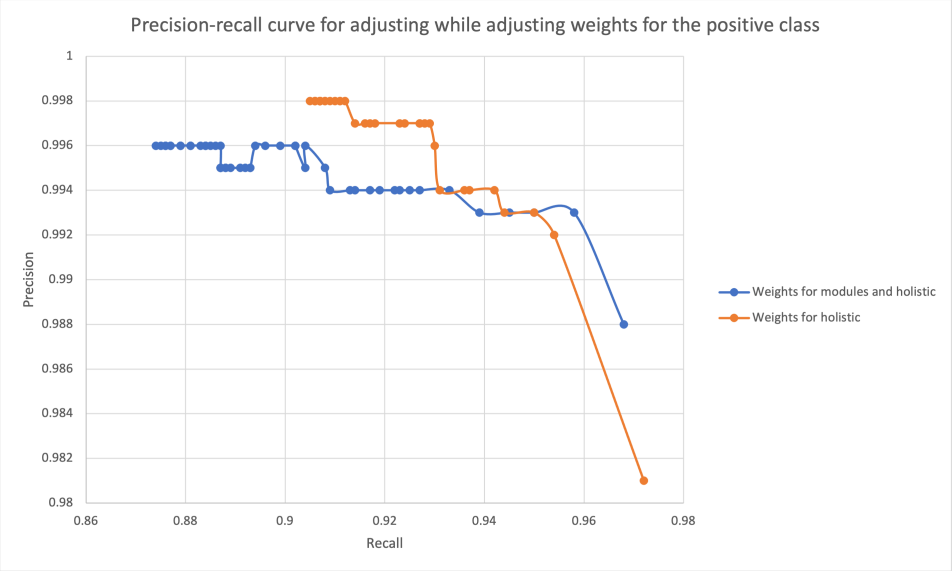


Figure 7.15: Precision recall curve for holistic module while adjusting the weights for the positive classification class.

In Figure 7.15, the precision and recall curve for adjusting the weights in the meta classifier have been depicted. For two of the three scenarios, such a graph can be shown. The third scenario, adjusting only the weights of the structure module and fusion module, does not involve the adjustment of the weights for the meta classifier. From Table 7.13, it can be deducted that each of the three scenarios

produce a lower number of false positives than the non-weighted scenario (Table 7.10). Though, this comes at a cost of a lower accuracy.

Random Forest regression predictor For previous classifications for the holistic system, a classification predictor was used. As has been discussed before in Section 2.2, these predictors differ from regression predictors. The classification predictor provides a classification label for each sample. For the holistic system, this meant that an email was labelled as phishing or benign. The regression score provides a score of how likely an email is phishing. A value close to 1 means the predictor predicts phishing, and a value close to 0 means the predictor predicts the sample to be benign. This regression score can still result in the same labelling as a classification algorithm, but the threshold for which value corresponds to which label can be adjusted. For example, one can label any sample with a regression score below .9 as benign and a regression score of .9 and above as phishing. By adjusting this threshold, the performance of the model will differ. A Random Forest regression predictor has been trained for the holistic system, and the thresholds were adjusted from .5 up to .95. In Table 7.14, the performance of this model has been depicted. The number of false positives reduces when the threshold varies. Though, this results in more false negatives. When fewer false positives are desired, a regression predictor can be used instead of a classifier.

Table 7.14: Performance of a Random Forest regression predictor for varying threshold

Threshold	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95
True positives	1693	1690	1677	1676	1669	1661	1651	1645	1634	1602
True negatives	2377	2381	2382	2388	2391	2393	2396	2399	2400	2405
False positives	35	31	30	24	21	19	16	13	12	7
False negatives	40	43	56	57	64	72	82	88	99	131
Total	4145	4145	4145	4145	4145	4145	4145	4145	4145	4145
Precision	0.98	0.982	0.982	0.986	0.988	0.989	0.99	0.992	0.993	0.996
Recall	0.977	0.975	0.968	0.967	0.963	0.958	0.953	0.949	0.943	0.924
Accuracy	0.982	0.982	0.979	0.98	0.979	0.978	0.976	0.976	0.973	0.967

Time performance Holistic system The experiments were run on a Dell XPS 15 7590 with Intel Core i7-9750H CPU @ 2.60 GHz and 32.0GB RAM. The time it takes for the holistic system to make a prediction on the email, is dependent on the time it takes to extract features from the email by the structure-, website and fusion module. After these modules have created a regression score for the email, the prediction time for the holistic system is negligible. From Section 7.2.1, Section 7.2.2 and Section 6.2.3, the execution times for these models are known. The website module uses the Phishpedia model to create a prediction. Since Phishpedia is the basis for the brand recognition model, the Phishpedia model does not have to be run twice for both the website- and fusion module. All time-consuming tasks such as taking screenshots, detecting logos and identifying brands can be done once and the information can be used in either module. Compared to the time-consuming website module and fusion module, the execution time of the structure module is negligible. Since the fusion module uses the same resources the website module uses along with additional computations, the execution time of the fusion module will also be the execution time of the meta classifier.

Qualitative analysis After the performance of the model has been measured, it is interesting to conduct a qualitative analysis of the model. During this analysis, a benign email will be adjusted and turned into a phishing email, while the sample should be kept undetectable by the system as a phishing email.

The analysis of the performance of the different modules will be used to guide the adjustments which will be made. This analysis will provide information on indicative features in the model. Those features

will have a high impact on the prediction of the model and should therefore be left minimally changed in respect to a benign email. The procedure to create such an adversarial example will be as followed:

- Take an email which is benign, and also flagged by the system as such
- Adjust several elements of the email and see how the system behaves.
- In some cases, add a phishing link.

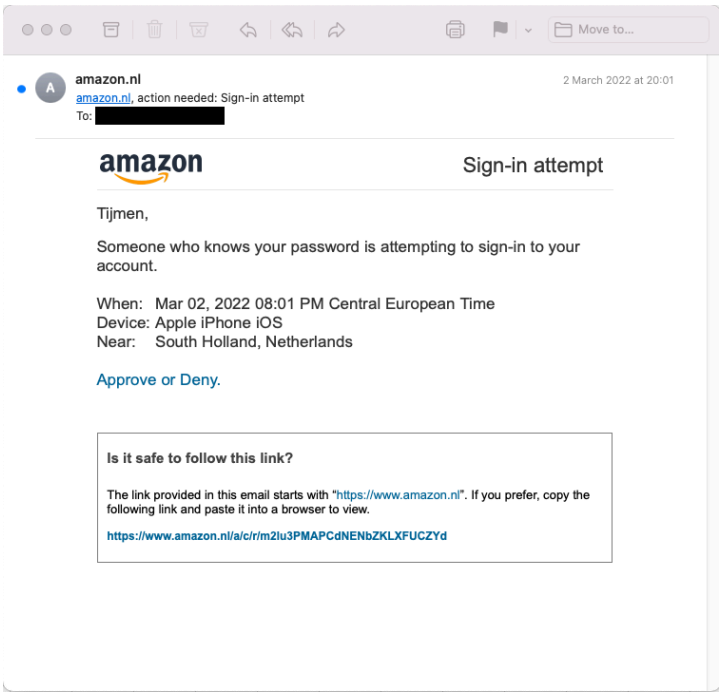


Figure 7.16: Email used for qualitative analysis. Email from every scenario has an identical layout

Several scenarios are investigated:

1. Original Amazon email. Used to see if it is flagged as benign
2. The original Amazon email, but sent from a different but benign amazon domain. This is sort of legitimate since the email originates from a different domain than originally, but the actual domain from which the email is sent is legitimate Amazon.
3. Amazon email from a phishing address. This is straight up suspicious, even without a phishing link. Though, this should be less suspicious than the same email with a phishing link.
4. Amazon email from a phishing address with a phishing link. The same email as the one before, but with a phishing link.
5. Amazon correct email address with phishing link. This might be indicative of a spoofed message.
6. Amazon email from a phishing address and a phishing link, where the phishing link and email domain are equal. The email originates from the same domain where the phishing website is hosted. This is investigated due to the best-performing feature of the fusion module (Table 6.11).
7. Amazon email from a phishing address and a phishing link, where the phishing link and email domain are equal, but with several benign URLs. Same email as the one before this, but with multiple benign domains added. This is done since the number of URLs in benign emails was higher than for phishing emails. Thus, it is interesting to see whether the system will be less confident of this being phishing due to the feature: number of URLs in email. (Table 6.11). The URLs are not visible and do not adjust the layout of the email.

Table 7.15: Predictions for the modified emails for the qualitative analysis

Email	Phishing link?	Structure	Fusion	Website	Holistic
Amazon original	0	0.69	-0.04	0	**-0.09**
Amazon different but legit email address	0	0.69	0.48	0	**0.72**
Changed email address	0	0.69	1.01	0	**0.99**
Changed email address and insert phishing link	1	0.77	0.98	1	**1**
Amazon correct email address with phishing link	1	0.72	1	1	**1**
Amazon phishing link and email domain equal	1	0.86	0.96	1	**1**
Amazon phishing link and email domain equal and extra URLs	1	0.3	0.86	1	**1.02**

Benign emails The prediction results of each module, as well as the holistic prediction (regressed), are presented in Table 7.15. The original email is, with a lot of confidence, being flagged as benign, which is correct. The second scenario is being flagged as phishing, but with a lot less confidence than the other scenarios. This behaviour is expected and desired, although it would be better if the model would be uncertain, so a regression is even closer to 0.5. The only difference in regression scores comes from the fusion module. This module now states it is uncertain about the legitimacy of the email. Three important features which differ between the first and second scenarios:

- Certificate type of email domains
- Minimum days between registration of mail domain and website
- URL with same root domain as from address feature

Apparently, the combination of these differences makes the email a bit more likely to be phishing. Another interesting insight is that a rise of .52 regression score of the fusion module, results, in this case, in a rise of .81 regression score for the holistic system. Furthermore, if the structure module would be used as the sole source, both benign emails would be flagged as phishing.

Phish emails Only the sending of the email from a malicious domain immediately makes the email flagged as being phishing. This domain is a domain used for impersonating Amazon, obtained from Phishtank. By using a 'fresh' domain, the model is prevented from bias. Only changing to this illegitimate domain results in the model being very certain of this email being phishing. Even though the structure and website modules do not differ, the fusion module gives a prediction of 1, resulting in a regression score of 1 from the holistic system. Inserting a phishing link even lowers the regression score for the fusion module a bit, although the website module and structure module do have an increase in regression.

In the last scenario, the performance of the "nr of URLs in email" feature is abused into tricking the fusion module into predicting the email as benign. This attack does work somewhat, the regression score is lowered by .10. Though, the combination of the three modules results in the highest regression score for the holistic system.

Overall good performance The overall performance of both the holistic system as well as the fusion module shows good performance for the email used in the scenarios. The original email is recognised as benign. Changing the domain of which the email has been sent raises suspicion but only when

phishing domains are used or phishing links added, the model is certain in predicting the email to be phishing.

8

Discussion and conclusion

The increase in the number of phishing attacks as well as the increase in sophistication of phishing attacks challenges the academic community to improve phishing detection capabilities.

8.1. Answering the research questions

Three research questions have been answered in this research in order to reach the research goal.

What different phases occur in a credential phishing attack?

To answer this research question, previously conducted research on general phishing attacks was investigated to see their applicability for credential phishing attacks as considered in this research. It was found in Section 3.1 that a credential phishing attack can be defined to have four phases. First, an attacker will plan its attack in the planning phase. While planning, the attacker will obtain a target, craft an email and create a website. Next, in the preparation phase, the attacker will host the phishing website and send the phishing email. Then, the attack conduction phase is entered. This phase has been separated into four different stages. The attack conduction starts when the email is delivered. Then the email is received, the website visited and lastly, credentials are stolen. The final phase is the valuables acquisition phase. During this study, no research was found that performed phishing detection on the combination of stages within the attack conduction phase. Combining the 'email received' and 'website visited' stages into a single phishing detection system had not been explored, as well as combining different stages into a meta classifier.

How to create a phishing detection mechanism which uses information from different stages of a phishing attack?

In Chapter 6, this question has been answered. By means of a brainstorm with cyber security professionals, several interesting features were defined. These features extract information which becomes available when looking at an email and the websites it includes as one entity, rather than two isolated entities. Several machine learning models have been trained on these features, creating a phishing detection system called the Fusion module. The best performing machine learning model was found to be an XGBoost algorithm with a precision of .967 and a recall of .947. This was just under the performance of the Structure module from Section 7.1.2, making its phishing detection performance near state-of-the-art. Some of the features use the recognition of brands in both emails and websites. For these features, a brand recognition system was designed. This brand recognition system uses multiple pre-trained neural networks and Google reverse image search to recognise brands in emails and websites. Each sub-part of this brand recognition system was tested and fine-tuned for usage in the features of the Fusion module. Though, this brand recognition system turned out to take 73.3% of the execution time of the feature extraction for the Fusion module. This was mainly caused by the analysis of the visuals of the websites and emails. Even so, the time needed for extraction of the brand recognition features grew linearly with the number of URLs present in an email. Due to the brand-recognition system, the feature extraction of the Fusion module took on average 96 seconds. Still, the brand recognition features did add classification capabilities to the model. When leaving the brand recognition features out of the model, the feature extraction time dropped to 25.57 seconds and the performance dropped to a precision of .957 and a recall of .943. The best performing features were features using the domain

registration of either email domains, website domains or both. For the dataset used, using a combination of brand recognition and non-brand recognition features as created in Section 6.1.2 forms a good basis for a phishing detection mechanism which uses information from different stages of a phishing attack.

How to combine phishing detection mechanisms from different stages into a holistic detection mechanism?
To answer this question, a phishing detection system performing phishing detection on the 'email received' stage, and one on the 'website visited' stage were designed. These systems were combined with the Fusion module introduced in this thesis. Together, these three detection systems comprise two stages in the attack conduction phase in the phishing attack chain. The Structure module performs phishing detection in the 'email received' stage. For this module, 68 features were extracted that focus on structural aspects of an email. From those, several machine learning models were built and evaluated on performance. The best performing algorithm was an XGBoost classifier, with a precision of .977 and a recall of .957. For the 'website visited' stage, the website module was introduced. This website module was created from a detection model, predicting whether a website was likely to be phishing or not. This module was based on the recognition of brands in a website. The module produces a precision of .661 and a recall of .775. To obtain a regression result for the website module, a regression formula was introduced. Both the Structure module and the website module produce a regression score for an email with their prediction of how likely this email is phishing. Together with the Fusion module proposed in Chapter 6, three regression scores for the likelihood of an email being phishing are produced. A meta classifier has been built to use these three scores as a feature in a machine learning model. This meta classifier uses the output of every individual module and gives a final prediction on whether an email is phishing or not. The machine learning model used was a Support Vector Machine classifier with a precision of .985 and a recall of .974 on the dataset used in this thesis. This meta classifier produces thus a result better than any of the individual detection systems. As was discussed in Section 7.1.2, the structure module was created by using state-of-the-art detection systems. Unfortunately, the performance of the system designed in this thesis was not able to match the same performance as found in the original papers. This might very well be due to some implementation flaws but probably by the use of different, smaller, datasets used in this thesis. Still, by building a meta classifier from different detection stages, a holistic detection mechanism has been created showing improved phishing detection capabilities over each individual model.

8.2. Conclusion

The goal of this research was to investigate phishing detection capabilities of taking a holistic approach to a credential phishing attack. This holistic approach meant taking different stages in the credential phishing attack chain into consideration when performing phishing detection. The holistic approach was investigated by creating two new detection systems. First, a phishing detection system was created that combines two different stages from the credential phishing attack chain in a singular detection system. Second, three detection systems were combined into a single phishing detection system. Together, these three detection systems spanned two stages in a credential phishing attack. This 'holistic' system showed improved phishing detection capabilities over each individual system. Thus, within the scope of this thesis, two phishing detection systems taking a holistic approach were designed and evaluated. It would be interesting to see the performance of the systems proposed in this thesis when run on the same datasets used for the evaluation of the state-of-the-art systems. Still, both introduced systems show detection capabilities competitive with state-of-the-art systems. It is concluded that taking a holistic approach to a credential phishing attack has very promising phishing detection capabilities.

8.3. Discussion

It is good to note that the research as performed has several limitations. Among other, smaller limitations, this section will address the limitation with respect to the phishing email dataset. Also, the vulnerability of the brand recognition system to adversarial attacks and the computation time of the Fusion module are discussed.

8.3.1. Phishing email dataset

The dataset used to validate the overall performance of both the Fusion module and holistic system consists of a combination of both open and private datasets (see Chapter 5). It is important to note that these datasets are a very limited scope of the entire email landscape. Even though the datasets are selected with care, the content, structure and layout of both the emails and the websites in the emails will most likely differ very much from many other emails.

Next to this, some of the phishing web pages used during the analyses were not the original web pages from the emails, as was explained in Section 5.1.5. To keep as close to the original pages as possible, phishing pages used for analysis were pages from the same brands as any brands identified in the email. Though, it was not possible to identify a brand in all emails in the dataset. Furthermore, the selection of which of the domains in the email was the phishing domain was done in an automatic way. This might result in the wrong web page being replaced with a phishing page. Also, a phishing page for the wrong brand might be inserted. The impact of this enrichment is probably limited, however. Remember from Section 5.1.5, that only the layout of the no longer available original phishing website was replaced with the screenshot of a phishing web page. The actual URL, registration history and other information were kept intact. Section 6.2.3 showed that the classification performance of the features which use the brand recognition system has a limited impact on the total performance of the Fusion module. Since the enrichment of the dataset was only used by the brand recognition system, it can be expected that the total impact on the performance of the proposed system was minimal. On the other hand, it might even be so that the enrichment of the dataset has a very negative impact on the performance of the features using the brand recognition system. In that case, using fresh phishing emails could even improve the detection capabilities of the system.

8.3.2. Adversarial attack on the brand recognition system

A known vulnerability for neural networks is an adversarial attack [61]. In such an attack, two visually similar logos can result in a totally different classification. Since the brand recognition system uses neural network-based techniques, the brand recognition system is vulnerable to this type of attack. This might trick the brand recognition system into classifying a wrong brand to a screenshot, or not classifying a brand at all when there clearly is one present. This behaviour probably has a limited impact on the Fusion module due to the limited impact of the brand recognition system on this module, as has been shown in Section 6.2.3. Though, an adversarial attack has a high impact on the website module, as this leans heavily on the brand recognition system. When taking into account the importance of the website module to the holistic system as depicted in Figure 7.14, it can be argued that an adversarial attack has minimal impact on the classification capabilities of the holistic system.

8.3.3. Whitelisting domains for the Website module

The Website module is able to recognise a fixed set of brands, as was discussed in Section 7.1.3. For each of these brands, the Website module holds domains which are owned and operated by that brand. If the logo of a brand appears on a domain which is owned by that brand, the website will not be labelled as phishing. One could suggest, that the website module can thus simply take the list of all domains of all brands. If the domain of the website under investigation is in this list, the Website module can simply classify the website as benign, without even trying to recognise any brand. This would, however, make the system as strong as the security of the weakest domain in the whitelist. If any of the domains owned by any of the brands is hacked, an attacker can host a phishing site on the hacked domain. It can then try to phish for credentials of any brand. This phishing URL will then not be labelled as phishing, since the domain is in the whitelist. This also holds for any other scenario in which a domain stops being under the control of the brand by which it is associated.

8.3.4. Computation time of the Fusion module

A lot of time is needed to extract the features for the Fusion module (Section 6.2.3). In particular, the features using the brand recognition system take a lot of time. The average time for the system to extract all features is 96 seconds per email. For an average email user, this delay in the receiving of the email is not too much trouble. Also, some of this time can be reduced by introducing more threading and fine-tuning waiting times introduced in several places in the system. Even with these improvements though, the brand recognition system will still take some computation time. This can become expensive. Especially for large organisations where many emails are received on a daily basis.

By removing the brand recognition features from the system, the computation time would decrease by 73%. This will reduce the classification performance of the module, but computation time is greatly decreased. The residual computation time can be further improved by optimising code, introducing threading and more available CPU.

8.4. Future work

The primary limitation of this research is the dataset. As mentioned in Section 8.3, phishing URLs are not always online and the variety of the dataset is limited in scope. It might very well be so that due to the limitations, there are biases in the dataset. These biases could result in an over-or underperforming prediction module. Features from the Fusion module which performed good on the dataset used in this research, might not be performing so well on other datasets. Also, when the features described in this thesis were to be extracted immediately after receiving an email, the visuals of any potential phishing websites could be analysed better. This way, results for features using the brand recognition system would be more reliable.

The problem with datasets containing emails is confidentiality. Even with the system proposed in Chapter 5, external companies were not too keen on sharing data. A very interesting approach would be to use federating learning to train the system. After the training of the system on a local dataset, the holistic phishing detection system could be offered as an Outlook add-in. Living in the inbox of clients, the system would be able to predict an email for being phishing or not. If some sort of feedback system could be introduced, users could give the system feedback on whether the prediction on the email was actually correct. Using this in combination with federated learning techniques[53][36], data would be kept locally on the client-side, and no data sharing is needed. In order to enable the holistic system to be used in a federated learning system, the machine learning model should be built from a neural network instead of a general machine learning model. The Structure module has been built from a deep learning model as a tryout. This can be seen in Chapter E.

In this research, two different stages from the attack conduction phase in a credential phishing attack were chosen to be included in the holistic approach. Though, there are more stages in the attack conduction phase. Expanding the holistic approach by adding phishing detection mechanisms on more stages of the attack conduction phase might create an even better detection mechanism. Even more so, the stages chosen in the holistic approach were both from the attack conduction phase. Combining attack indicators from the valuables acquisition phase as defined in Section 3.1, could improve the system proposed in this research.

Appendices

A

General feature analysis structure module

Table A.1: General statistics structure module features.

column names	zero occurren in phish	zero occurre in benign	only zero occurrer in phish	only zero occurrer in benign	frequency of the option for this variable occurring the most	frequenc of most occurrin element benign	most occurring feature same for phish and benign
index of charset in first section	128	6	FALSE	FALSE	0.56	0.859	FALSE
number of a tags with href and datasaferedirecturl	79	30	FALSE	FALSE	0.464	0.09	FALSE
number of standard headers	0	0	FALSE	FALSE	0.091	0.082	FALSE
email message size in bytes	0	0	FALSE	FALSE	0.017	0.005	FALSE
number of urls in text html section	61	27	FALSE	FALSE	0.257	0.07	FALSE
length of style bodies and inline style bodies	162	46	FALSE	FALSE	0.093	0.028	FALSE
length of message id	282	0	FALSE	FALSE	0.163	0.138	FALSE
length of boundary id	587	471	FALSE	FALSE	0.339	0.195	TRUE
number of words in body	122	39	FALSE	FALSE	0.07	0.016	TRUE
length of texts in text html sections	295	269	FALSE	FALSE	0.17	0.112	TRUE

boundary id is decimal	1162	2378	FALSE	FALSE	0.671	0.986	TRUE
number of dom leaf nodes	60	26	FALSE	FALSE	0.04	0.022	FALSE
ratio of unique domains to domains of all urls any tags	65	27	FALSE	FALSE	0.474	0.095	TRUE
boundary id starts with equal symbol	1142	2321	FALSE	FALSE	0.659	0.962	TRUE
ratio of words to characters	122	39	FALSE	FALSE	0.194	0.087	FALSE
number of received entities	14	61	FALSE	FALSE	0.391	0.254	TRUE
number of unique dom leaf node types	60	26	FALSE	FALSE	0.168	0.162	FALSE
standard deviation of dom leaf nodes depth	79	45	FALSE	FALSE	0.046	0.019	TRUE
ratio of unique to domains to unique domains in all email addresses	156	9	FALSE	FALSE	0.515	0.519	FALSE
mean of dom leaf node depths	60	26	FALSE	FALSE	0.035	0.011	TRUE
number of unique charsets in all sections	125	6	FALSE	FALSE	0.887	0.947	TRUE
return path addresses are equal to reply to	784	1852	FALSE	FALSE	0.548	0.768	FALSE
depth of dom object tree	60	26	FALSE	FALSE	0.053	0.058	FALSE
ratio between words and characters in subject	16	10	FALSE	FALSE	0.159	0.126	FALSE
ascii number of message id boundary character	282	0	FALSE	FALSE	0.72	0.621	TRUE
case variance in mime version	1708	2012	FALSE	FALSE	0.986	0.834	TRUE
boundary id is hexadecimal	1066	1786	FALSE	FALSE	0.615	0.74	TRUE
ratio of text plain to any text sections	580	528	FALSE	FALSE	0.544	0.588	TRUE
number of email addresses in text html section	1020	1423	FALSE	FALSE	0.589	0.59	TRUE

to entity is bracketed	1112	1156	FALSE	FALSE	0.642	0.521	FALSE
number of unique domains from hyperlinks	70	34	FALSE	FALSE	0.682	0.457	TRUE
existence of other special characters in boundary id	1479	1431	FALSE	FALSE	0.853	0.593	TRUE
existence of special characters in id part of message id	578	281	FALSE	FALSE	0.666	0.883	TRUE
existence of equal symbol in middle of boundary id	903	1664	FALSE	FALSE	0.521	0.69	TRUE
existence of underscores in boundary id	1234	1190	FALSE	FALSE	0.712	0.507	FALSE
number of unique domains in all email addresses	10	0	FALSE	FALSE	0.564	0.529	FALSE
from domain equal to return path domain	444	1379	FALSE	FALSE	0.744	0.572	FALSE
existence of dots in id part of message id	883	1464	FALSE	FALSE	0.51	0.607	TRUE
number of in reply to entities	1662	2058	FALSE	FALSE	0.959	0.853	TRUE
number of text plain sections	580	528	FALSE	FALSE	0.6	0.642	TRUE
id part of message id is hexadecimal	709	1053	FALSE	FALSE	0.591	0.563	TRUE
re in subject	1702	2187	FALSE	FALSE	0.982	0.907	TRUE
number of to domains equal to from domains	1333	2121	FALSE	FALSE	0.769	0.879	TRUE
existence of x mailer	1483	2056	FALSE	FALSE	0.856	0.852	TRUE
number of unique reply to addresses in mail thread	1441	1534	FALSE	FALSE	0.832	0.636	TRUE
number of text html sections	58	26	FALSE	FALSE	0.852	0.833	TRUE
existence of special characters in domain part of message id	383	131	FALSE	FALSE	0.779	0.946	TRUE

number of image sections	1579	2051	FALSE	FALSE	0.911	0.85	TRUE
id part of message id is decimal	1628	2104	FALSE	FALSE	0.939	0.872	TRUE
existence of dots in boundary id	1533	2064	FALSE	FALSE	0.885	0.856	TRUE
existence of dots in domain part	465	337	FALSE	FALSE	0.732	0.86	TRUE
number of applications sections	1579	2051	FALSE	FALSE	0.911	0.85	TRUE
number of sender addresses	33	0	FALSE	FALSE	0.964	0.98	TRUE
existence of inline javascript	1648	2275	FALSE	FALSE	0.951	0.943	TRUE
from domain equal to reply to domain	137	342	FALSE	FALSE	0.921	0.858	TRUE
number of to header tags in mail thread	92	8	FALSE	FALSE	0.946	0.995	TRUE
from entity is bracketed	63	75	FALSE	FALSE	0.964	0.969	TRUE
number of unique to addresses in mail thread	92	8	FALSE	FALSE	0.946	0.995	TRUE
existence of user agent	1712	2407	FALSE	FALSE	0.988	0.998	TRUE
number of cc addresses	1728	2258	FALSE	FALSE	0.997	0.936	TRUE
number of cc domains identical to from domains	1730	2316	FALSE	FALSE	0.998	0.96	TRUE
domain part of message id is hexadecimal	1712	2319	FALSE	FALSE	0.988	0.961	TRUE
existence of direction rtl style	1732	2344	FALSE	FALSE	0.999	0.972	TRUE
domain part of message id is decimal	1712	2319	FALSE	FALSE	0.988	0.961	TRUE
number of from header tags in mail thread	10	0	FALSE	FALSE	0.992	0.998	TRUE
number of bcc addresses	1731	2411	FALSE	FALSE	0.999	1	TRUE
mime version of first section	0	0	FALSE	FALSE	0.999	1	TRUE

number of bcc domains equal to from domains	1733	2412	TRUE	TRUE	1	1	TRUE

Table A.2: Feature ranking structure module.

column names	feature ranking (high is best)
index of charset in first section	67
number of a tags with href and datasaferedirecturl	66
number of standard headers	65
email message size in bytes	64
number of urls in text html section	63
length of style bodies and inline style bodies	62
length of message id	61
length of boundary id	60
number of words in body	59
length of texts in text html sections	58
boundary id is decimal	57
number of dom leaf nodes	56
ratio of unique domains to domains of all urls any tags	55
boundary id starts with equal symbol	54
ratio of words to characters	53
number of received entities	52
number of unique dom leaf node types	51
standard deviation of dom leaf nodes depth	50
ratio of unique to domains to unique domains in all email addresses	49
mean of dom leaf node depths	48
number of unique charsets in all sections	47
return path addresses are equal to reply to	46
depth of dom object tree	45
ratio between words and characters in subject	44

ascii number of message id boundary character	43
case variance in mime version	42
boundary id is hexadecimal	41
ratio of text plain to any text sections	40
number of email addresses in text html section	39
to entity is bracketed	38
number of unique domains from hyperlinks	37
existence of other special characters in boundary id	36
existence of special characters in id part of message id	35
existence of equal symbol in middle of boundary id	34
existence of underscores in boundary id	33
number of unique domains in all email addresses	32
from domain equal to return path domain	31
existence of dots in id part of message id	30
number of in reply to entities	29
number of text plain sections	28
id part of message id is hexadecimal	27
re in subject	26
number of to domains equal to from domains	25
existence of x mailer	24
number of unique reply to addresses in mail thread	23
number of text html sections	22
existence of special characters in domain part of message id	21
number of image sections	20
id part of message id is decimal	19
existence of dots in boundary id	18
existence of dots in domain part	17
number of applications sections	16

number of sender addresses	15
existence of inline javascript	14
from domain equal to reply to domain	13
number of to header tags in mail thread	12
from entity is bracketed	11
number of unique to addresses in mail thread	10
existence of user agent	9
number of cc addresses	8
number of cc domains identical to from domains	7
domain part of message id is hexadecimal	6
existence of direction rtl style	5
domain part of message id is decimal	4
number of from header tags in mail thread	3
number of bcc addresses	2
mime version of first section	1
number of bcc domains equal to from domains	0

B

Common email service providers used in Fusion module

- gmail.com
- googlemail.com
- outlook.com
- hotmail.com
- yahoo.com
- aol.com
- protonmail.com
- zoho.com
- icloud.com
- gmx.com

C

Social media brands used in Fusion module

- Facebook
- Twitter
- Linkedin
- Instagram
- Youtube
- Pinterest
- Tumblr
- Reddit
- Whatsapp
- Tiktok
- WeChat

D

General feature analysis Fusion module

Table D.1: General statistics Fusion module features.

column names	zero occurrence in phish	zero occurrence in benign	only zero occurrence in phish	only zero occurrence in benign	frequency of the option for this variable occurring the most	frequency of most occurring element benign	most occurring feature same for phish and benign
any links redirects to main domain of large company feature	1413	1095	FALSE	FALSE	0.815	0.519	FALSE
at least one web-site with no logos feature	1444	1107	FALSE	FALSE	0.833	0.541	FALSE
average nr logos detected by logo detection per website feature	1	5	FALSE	FALSE	0.642	0.286	TRUE
brand but from com-mon email provider feature	1730	2407	FALSE	FALSE	0.998	0.998	TRUE

brand in email is social media feature	1732	2406	FALSE	FALSE	0.999	0.998	TRUE
brand is recognised in email feature	1381	1816	FALSE	FALSE	0.797	0.753	TRUE
brand of website is mentioned in email feature	155	264	FALSE	FALSE	0.797	0.75	TRUE
certificate type of domains of links in email feature	729	46	FALSE	FALSE	0.421	0.309	FALSE
certificate type of email domains feature	474	237	FALSE	FALSE	0.416	0.468	TRUE
days since newest mail domain registration feature	1	0	FALSE	FALSE	0.179	0.061	FALSE
days since oldest mail domain registration feature	338	53	FALSE	FALSE	0.195	0.061	FALSE
dmarc check success feature	1058	438	FALSE	FALSE	0.611	0.818	FALSE
domain is of common email provider feature	1698	2201	FALSE	FALSE	0.98	0.913	TRUE
from email address is owned by brand in email feature	298	399	FALSE	FALSE	0.536	0.452	TRUE

maximum days between registration of mail domain and website feature	729	923	FALSE	FALSE	0.421	0.383	TRUE
minimum days between registration of mail domain and website feature	182	1758	FALSE	FALSE	0.36	0.729	FALSE
minimum days since registration of any link feature	0	0	FALSE	FALSE	0.244	0.042	FALSE
nr of brands recognised in email feature	1381	1816	FALSE	FALSE	0.797	0.753	TRUE
nr of times brand in website but different from email feature	343	381	FALSE	FALSE	0.582	0.158	FALSE
nr of times brand in website is same as brand in email feature	1437	2031	FALSE	FALSE	0.829	0.842	TRUE
nr of times logo in website is same as logo in email feature	1474	1398	FALSE	FALSE	0.851	0.58	TRUE
nr of urls in email feature	0	0	FALSE	FALSE	0.555	0.092	FALSE

nr of websites with brand recognised feature	157	300	FALSE	FALSE	0.636	0.125	FALSE
spf check success feature	717	184	FALSE	FALSE	0.586	0.924	TRUE
url in email performs redirect feature	1056	203	FALSE	FALSE	0.609	0.898	FALSE
url with same root domain as from address feature	1531	1112	FALSE	FALSE	0.883	0.5	FALSE
urls of unknown brands exist and brand recognised feature	1635	2096	FALSE	FALSE	0.943	0.869	TRUE
urls of unknown brands exist feature	1161	1022	FALSE	FALSE	0.67	0.576	FALSE
website without logo but brand in email feature	1444	2136	FALSE	FALSE	0.833	0.886	TRUE

Table D.2: Feature ranking for Fusion module.

column names	feature ranking (high is best)
minimum days between registration of mail domain and website feature	28
nr of urls in email feature	27
url in email performs redirect feature	26
certificate type of domains of links in email feature	25
nr of times brand in website but different from email feature	24

days since newest mail domain registration feature	23
nr of websites with brand recognised feature	22
days since oldest mail domain registration feature	21
minimum days since registration of any link feature	20
dmarc check success feature	19
url with same root domain as from address feature	18
spf check success feature	17
maximum days between registration of mail domain and website feature	16
nr of times logo in website is same as logo in email feature	15
any links redirects to main domain of large company feature	14
average nr logos detected by logo detection per website feature	13
certificate type of email domains feature	12
domain is of common email provider feature	11
from email address is owned by brand in email feature	10
at least one website with no logos feature	9
urls of unknown brands exist feature	8
nr of times brand in website is same as brand in email feature	7
brand of website is mentioned in email feature	6
nr of brands recognised in email feature	5
website without logo but brand in email feature	4
brand is recognised in email feature	3
urls of unknown brands exist and brand recognised feature	2
brand in email is social media feature	1
brand but from common email provider feature	0

E

Deep neural network for structure module

The structure module as discussed in Section 7.1.2, has been trained on several machine learning models. In order to investigate the behavioural difference between the a 'regular' machine learning model and a neural network, a neural network has been trained and optimised to perform the prediction task by means of a neural network. For the sake of documentation, this appendix includes the approach taken and the results from the model.

Neural network

The neural network is comprised of several components. The optimal design for some components is determined by a literature study, whereas certain parameters in other components should be determined by optimisation.

As discussed in , a neural network consists of 3 layer types.

- Input layer
- Hidden layers
- Output layer

Other artefacts of a neural network are:

- Rectifier
- Loss function
- Learning algorithm
- initialisation [26] [9]

Adjustable parameters

- Number of layers
- Node configuration
- Learning rate
- epochs
- batch size

Loss function For a classification algorithm, the Cross-entropy loss function is a preferred choice [31][8].

Determining optimal architecture There are infinitely many options for the adjustable parameters. Therefore, it is infeasible to obtain the optimal composition of parameters by experiment. Though, since it is also impossible to calculate the optimal parameters, a strategy was used to obtain a good composition of parameters, in a tight time schedule.

First, the best performing number of layers will be determined. Next, a proper configuration of nodes in the layers will be estimated. Then, to prove theoretical equality, a linear network model is compared to a convolutional model. After that, with the configuration of the layers and nodes established, the final hyperparameters will be determined.

Layer performance To estimate the optimal layer configuration, a combination of theoretical and practical analysis is performed. First, a theoretical approach is taken to determine a general strategy for the layer configuration.

After the theoretical analysis, practical analysis is done. This is done by testing many semi-random configurations. First, a list of configurations is created. That list of configurations holds configuration settings for the number of hidden layers to be used in a model and the number of input and output nodes used in those layers. The number of nodes in the input layer is equal to the number of features (68) and the number of nodes in the output layer is equal to the number of classes which can be predicted (2). The list configuration then chooses a random number of nodes between 2 and twice the number of features (136). This is done for an equal number of configurations for models with 1 up to 5 hidden layers.
After early experiments, it was clear that good learning rates for any configuration are .1 and .2. Therefore, all configurations will be tried for both learning rates. The batch size is set to 48 and the configurations are run for both learning rates and trained for 20 and 200 epochs with different initialised weights and biases. Thus any configuration is run 4 times. Per number of layer category, a total of 150 unique configurations are created and all run 4 times.

Table E.1: Averaged performance of random layer configurations.

Nr of layers	1	2	3	4	5	6
Precision	0.933	0.902	0.883	0.91	0.865	0.804
Recall	0.806	0.749	0.719	0.749	0.715	0.634
Accuracy	0.92	0.898	0.886	0.899	0.884	0.854
Auc_roc	0.958	0.938	0.929	0.943	0.922	0.883
Auc_pr	0.971	0.961	0.955	0.963	0.949	0.925

Table E.2: Maximum values of performance of random layer configurations.

Nr of layers	0	1	2	3	4	5
Precision	1	1	1	1	1	1
Recall	1	1	1	0.998	1	1
Accuracy	0.996	0.996	0.996	0.996	0.996	0.995
Auc_roc	0.999	0.999	1	0.999	1	0.999
Auc_pr	0.999	0.999	0.999	0.999	0.999	0.999

The performance as shown in Table E.1 shows the best-averaged performance for architecture with 0 hidden layers. Though, the highest average performance score is not the best indicator for determining the best number of layers. In Figure E.1 and Figure E.2, a bar chart is plotted. This chart shows the number of times the recall of a configuration falls in a certain recall region. These graphs give two important insights. In Figure E.2, one can see that 0 hidden layers perform good for a recall higher than .97 and lower or equal to .99. Though, it has few configurations in the best-defined segment (.99 - 1). Therefore, on average, the 0 hidden layers might perform well, but other configurations have a higher chance of a configuration in the top segments. In Figure E.1, it is remarkable that an increasing number of layers performs worse than the lower numbers. When investigating the layer configuration

of configurations with a recall of 0, it is notable that many configurations include 2 consecutive layers with the same number of nodes.

Since configurations with 5 hidden layers have produced the best scores as shown in Table E.2, and perform have a high chance of good performance (Figure E.2), while reducing the chance of overfitting with respect to 6 hidden layers, this layer number has been chosen.

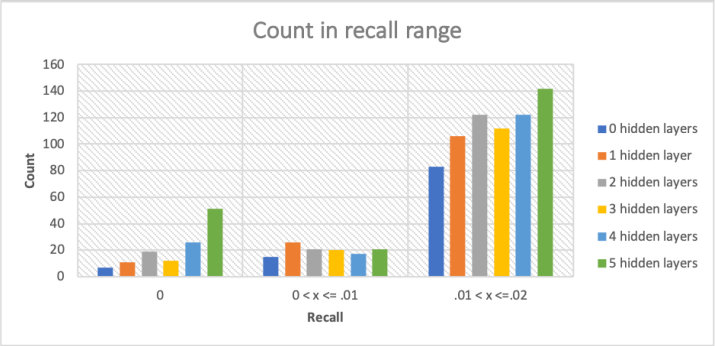


Figure E.1: Count of recall values per configuration for the lower segment of the recall range

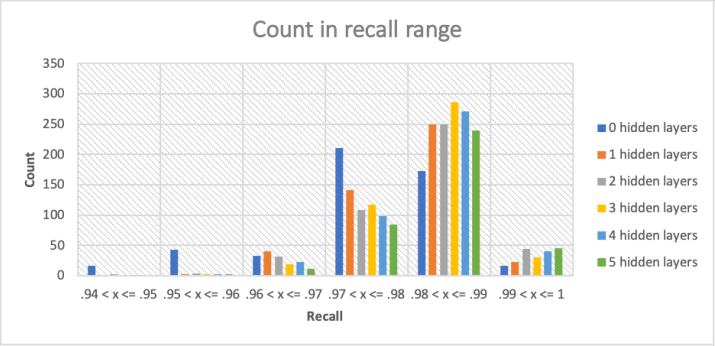


Figure E.2: Count of recall values per configuration for the higher segment of the recall range.

Node configuration The number of layers has now been determined. Now, an experiment is run to determine a well-performing node configuration. The experiment run is very similar to the experiment for the number of layers. The only exception is that the configurations for the models are now fixed on the number of layers. It is important to note that by taking this approach, the optimal configuration can never be determined. During the node configuration experiment, 300 different, semi-random configurations are used to train a model. The best performing model in area under precision-recall curve and recall, was the following:

Input layer: 68 nodes 1st hidden layer: 61 nodes 2nd hidden layer: 5 nodes 3rd hidden layer: 5 nodes 4th hidden layer: 3 nodes 5th hidden layer 2 nodes output layer: 2 nodes

Linear Transformation vs convolutional network

For the layers of nodes, a transformation is used to determine the output. In a convolutional neural network, the convolution part, means that a filter is applied to the input values. In the application of the structure module, this filter would combine the input of neighbouring features for an output value. Though, the feature order is chosen at random. Thus the filtering for neighbouring features would not make sense, and the filter size will be set to 1. This is effectively a linear transformation.

Hyper parameters The number of layers and nodes could be determined while keeping other variables (more or less) equal. Though, the parameters are still left to be determined, have a great impact on one another. For example, the combination of learning rate and the number of epochs has a great impact on the performance and training time needed for a model. Therefore, the previous plan of action of keeping most of the variables stable while investigating the impact on the performance of the other, will not be suitable for the hyperparameters. For the calculation of the best performing configuration of hyperparameters, all possibilities under investigation will be cross-validated. Thus,

for testing 5 learning rates and 5 epoch sizes, 25 different models should be run.
Though, in contrast with the practical determination of the nodes and layers, the initialization of the weights and biases can be done with the same values. Therefore, one source of error is eliminated.

F

Unused features for the Fusion module

During the feature engineering of the Fusion module, several feature ideas were created. Due to time and technology constraints, not all features were selected for implementation.

Table F.1: Unused features for the Fusion module.

Category	Name	Description
email	obfuscates javascript	There is Javascript obfuscation in the email
website	domain rating of urls	Domain rating of websites in email (website authority checker)
website	is website written in php	Is website written in PHP
email	does alt tag of picture in email hold brand name	Does alt-tag of picture hold brand name
email	name of from email address is brand	The display name of the ”from” address is the name of a brand in the list of brands used for the brand identification system
email	compare alt tag of picture with alt tag of website	Compare the alt-tag of pictures from the email with alt-tags on website. If one or more alt-tags overlap 1, otherwise 0
website + email	nr times language between email and website are equal	nr. Of times language between email and websites are equal
website + email	picture in email is used in website	The same picture from the email is used in website
email	Non ascii characters in url but not in text	There are non-ASCII characters in the url, but there are non in the text of the email. Or the general language of the text of the email is of an alphabet of ASCII characters.
website + email	nr overlapping domains email and links	Nr of overlapping domains between email and links
email	nr of overlapping domain certificate authorities	Nr of overlapping domain certificate authorities of all links in email
website	nr of overlapping domain certificate types	Nr of overlapping certificate types between all links in email

Website	free certificate	Indicates whether the certificate of urls is registered at a service which offers free certificates
email	email service provider is cloud	The email is sent by an email cloud service provider
email	email size	The size of all text in the email
email	attachment holds html file but no links in email	Attachment holds html file but no links
email	brand name appears in email	does a brand name appear in email
Website + email	content in email loaded from domain	Any content in email is loaded from a domain. E.g. pictures are loaded from a domain
website + email	same brand name appears in website and email	same brand name in both website and email
website + email	trust of email higher than website	Domain trust of email domain is higher than that of website domain
website	page rank of domain used for content loading	page rank of domain used for content loading
email	malicious ip in header	In some (old) cases, the ip address of the sender of the email appears in the headers of the email. That ip address might be known to be of a ”shady” hosting provider

Bibliography

[1] Giuseppe Aceto et al. “Cloud monitoring: A survey”. In: *Computer Networks* 57 (9 June 2013), pp. 2093–2115. ISSN: 13891286. DOI: 10.1016/j.comnet.2013.04.001. URL: https://doi-org.tudelft.idm.oclc.org/10.1016/j.comnet.2013.04.001.

[2] Zainab Alkhalil et al. “Phishing Attacks: A Recent Comprehensive Study and a New Anatomy”. In: *Frontiers in Computer Science* 3 (Mar. 2021). DOI: 10.3389/fcomp.2021.563060. URL: https://doi.org/10.3389/fcomp.2021.563060.

[3] Barracuda. *Spear Phishing: Top Threats and Trends*. Barracuda, July 2021.

[4] Faisal Bashir and Fatih Porikli. “Performance Evaluation of Object Detection and Tracking Systems”. In: *Proceedings 9th IEEE International Workshop on PETS* (2006), pp. 7–14.

[5] Candice Bentéjac, Anna Csörgő, and Gonzalo Martínez-Muñoz. “A comparative analysis of gradient boosting algorithms”. In: *Artificial Intelligence Review* 54 (3 Mar. 2021), pp. 1937–1967. ISSN: 15737462. DOI: 10.1007/s10462-020-09896-5.

[6] Hugo Bijmans et al. “Inadvertently making cyber criminals rich: A comprehensive study of cryptojacking campaigns at internet scale.” In: *28th USENIX Security Symposium (USENIX Security 19)* (2019), pp. 1627–1644. URL: www.usenix.org/conference/usenixsecurity21/presentation/bijmans.

[7] Ahmet Selman Bozkir and Murat Aydos. “LogoSENSE: A companion HOG based logo detection scheme for phishing web page and E-mail brand recognition”. In: *Computers and Security* 95 (Aug. 2020). ISSN: 01674048. DOI: 10.1016/j.cose.2020.101855.

[8] Jason Brownlee. *How to choose loss functions when training deep neural networks*. Jan. 30, 2019. URL: https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/ (visited on 06/24/2021).

[9] Jason Brownlee. *Weight initialization for Deep Neural Networks*. May 24, 2017. URL: https://machinelearningmastery.com/weight-initialization-for-deep-learning-neural-networks/ (visited on 04/13/2022).

[10] Jason Brownlee. *What Is Meta-Learning in Machine Learning?* URL: https://machinelearningmastery.com/meta-learning-in-machine-learning/#:~:text=Meta%25%5C%25%2Dlearning%25%5C%25%20algorithms%25%5C%25%20typically%25%5C%25%20refer,to%25%5C%25%20as%25%5C%25%20multi%25%5C%2Dtask%25%5C%20learning. (visited on 05/11/2022).

[11] Deanna D. Caputo et al. “Going spear phishing: Exploring embedded training and awareness”. In: *IEEE Security and Privacy* 12 (1 2014), pp. 28–38. ISSN: 15407993. DOI: 10.1109/MSP.2013.106.

[12] Tianqi Chen and Carlos Guestrin. “XGBoost: A Scalable Tree Boosting System”. In: 2016, pp. 785–794. ISBN: 9781450342322. DOI: 10.1145/2939672.2939785. URL: http://dx.doi.org/10.1145/2939672.2939785.

[13] Kang Leng Chiew, Kelvin Sheng Chek Yong, and Choon Lin Tan. “A survey of phishing attacks: Their types, vectors and technical approaches”. In: *Expert Systems with Applications* 106 (Sept. 2018), pp. 1–20. ISSN: 09574174. DOI: 10.1016/j.eswa.2018.03.050.

[14] COCO. *Random picture dataset*. 2017. URL: http://images.cocodataset.org/zips/val2017.zip (visited on 05/11/2022).

[15] Cofense. *History of Phishing*. URL: https://cofense.com/knowledge-center/history-of-phishing/ (visited on 04/11/2022).

[16] Cambridge Dictionary. *Meaning of Phase*. URL: https://dictionary.cambridge.org/dictionary/english/phase (visited on 11/24/2021).

[17] Cambridge Dictionary. *Meaning of Stage*. URL: https://dictionary.cambridge.org/dictionary/english/stage (visited on 11/24/2021).

[18] Yan Ding et al. “A keyword-based combination approach for detecting phishing webpages”. In: *Computers and Security* 84 (July 2019), pp. 256–275. ISSN: 01674048. DOI: 10.1016/j.cose.2019.03.018.

[19] Vincent Drury and Ulrike Meyer. “Certified Phishing: Taking a Look at Public Key Certificates of Phishing Websites”. In: *Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019)* (2019), pp. 211–223. URL: https://www.usenix.org/conference/soups2019/presentation/drury.

[20] Christa Dürscheid and Carmen Frehner. “Email communication”. In: *Pragmatics of Computer-Mediated Communication*. Vol. 9. DE GRUYTER, Jan. 2013, pp. 35–54. DOI: 10 . 1515 / 9783110214468 . 35.

[21] *Eye Security*. 2022. URL: https://www.eye.security/ (visited on 05/11/2022).

[22] Yong Fang et al. “Phishing Email Detection Using Improved RCNN Model With Multilevel Vectors and Attention Mechanism”. In: *IEEE Access* 7 (2019), pp. 56329–56340. ISSN: 21693536. DOI: 10.1109/ACCESS.2019.2913705.

[23] Fang Feng et al. “The application of a novel neural network in the detection of phishing websites”. In: *Journal of Ambient Intelligence and Humanized Computing* (Apr. 2018), pp. 1–15. ISSN: 18685145. DOI: 10.1007/s12652-018-0786-3.

[24] Irene Giacomelli et al. “Privacy-Preserving Collaborative Prediction using Random Forests”. In: *AMIA summits on translational science proceedings* (2019).

[25] *Google Cloud Vision API*. 2022. URL: https://cloud.google.com/vision (visited on 05/12/2022).

[26] Kaiming He et al. “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”. In: *Proceedings of the IEEE international conference on computer vision*. (2015), pp. 1026–1034.

[27] Jonathan Hui. *mAP (mean Average Precision) for Object Detection*. May 7, 2018. URL: https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173 (visited on 11/19/2021).

[28] Federal Bureau of Investigation internet crime complaint center. *Internet Crime Report 2020*. 2021. URL: https://www.ic3.gov/Media/PDF/AnnualReport/2020_IC3Report.pdf.

[29] Federal Bureau of Investigation internet crime complaint center. *Internet Crime Report 2021*. 2022. URL: https://www.ic3.gov/Media/PDF/AnnualReport/2021_IC3Report.pdf.

[30] Ankit Kumar Jain and B. B. Gupta. “Towards detection of phishing websites on client-side using machine learning based approach”. In: *Telecommunication Systems* 68 (4 Aug. 2018), pp. 687–700. ISSN: 15729451. DOI: 10.1007/s11235-017-0414-0.

[31] Namrata Kapoor. *Loss Functions - when to use which one*. Nov. 17, 2020. URL: https://towardsdatascience.com/loss-functions-when-to-use-which-one-718ebad36e0 (visited on 06/24/2021).

[32] KOUSTUBHK. *Logo dataset*. Mar. 3, 2022. URL: https://www.kaggle.com/datasets/kkhandekar/popular-brand-logos-image-dataset (visited on 05/11/2022).

[33] Elmer Eh Lastdrager. “SYSTEMATIC REVIEW Open Access Achieving a consensual definition of phishing based on a systematic review of the literature”. In: *Crime Science* 3 (BioMed Central 2014), pp. 1–10. URL: http://www.crimesciencejournal.com/content/3/1/9.

[34] Jehyun Lee et al. “D-Fence: A Flexible, Efficient, and Comprehensive Phishing Email Detection System”. In: *2021 IEEE European Symposium on Security and Privacy* (2021), pp. 578–597.

[35] Yun Lin et al. “Phishpedia: A Hybrid Deep Learning Based Approach to Visually Identify Phishing Webpages”. In: *30th USENIX Security Symposium (USENIX Security 21)* (2021), pp. 3793–3810.

[36] Dianbo Liu and Tim Miller. “Federated pretraining and fine tuning of BERT using clinical notes from multiple silos”. In: *arXiv preprint arXiv:2002.08562* (2020).

[37] Bjoern H. Menze et al. “A comparison of random forest and its Gini importance with standard chemometric methods for the feature selection and classification of spectral data”. In: *BMC Bioinformatics* 10 (July 2009), pp. 1–16. ISSN: 14712105. DOI: 10.1186/1471-2105-10-213.

[38] Jose Nazario. *Nazario data set*. URL: https://monkey.org/~jose/phishing/ (visited on 03/16/2022).

[39] Stephen J Nightingale. *Email authentication mechanisms: DMARC, SPF and DKIM*. National Institute of Standards and Technology, Feb. 2017. DOI: 10.6028/NIST.TN.1945. URL: https://nvlpubs.nist.gov/nistpubs/TechnicalNotes/NIST.TN.1945.pdf.

[40] Rahul Nijhawan, Balasubramanian Raman, and Josodhir Das. "Meta-Classifier Approach with ANN, SVM, Rotation Forest, and Random Forest for Snow Cover Mapping". In: *Advances in Intelligent Systems and Computing 704*. Ed. by Bidyut B Chaudhuri, Mohan S Kankanhalli, and Balasubramanian Raman. 2017, pp. 279–288. URL: http://www.springer.com/series/11156.

[41] William S Noble. "What is a support vector machine?" In: vol. 24. 2006. ISBN: 1514915154. URL: http://www.nature.com/naturebiotechnology.

[42] Adam Oest et al. "Sunrise to Sunset: Analyzing the End-to-end Life Cycle and Effectiveness of Phishing Attacks at Scale Sunrise to Sunset: Analyzing the End-to-end Life Cycle and EEectiveness of Phishing Attacks at Scale". In: *29th USENIX Security Symposium (USENIX) Security 20)* (2020). URL: https://www.usenix.org/conference/usenixsecurity20/presentation/oest-sunrise.

[43] Kemal Oksuz et al. "Localization Recall Precision (LRP): A New Performance Metric for Object Detection". In: *Proceedings of the European Conference on Computer Vision (ECCV)* (Sept. 2018).

[44] Padilla Rafael, L. Netto Sergio, and da Silva Eduardo A. B. "A Survey on Performance Metrics forObject-Detection Algorithms". In: *Proceedings of the IWSSIP 2020* (2020), p. 237. URL: https://ieeexplore-ieee-org.tudelft.idm.oclc.org/stamp/stamp.jsp?tp=&arnumber=9145130.

[45] Akhlaqur Rahman and Sumaira Tasnim. "Ensemble Classifiers and Their Applications: A Review". In: *International Journal of Computer Trends and Technology* 10 (1 2014). ISSN: 2231-2803. URL: www.internationaljournalssrg.org.

[46] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Advances in neural information processing systems* 28 (2015). URL: https://github.com/.

[47] Ozgur Koray Sahingoz et al. "Machine learning based phishing detection from URLs". In: *Expert Systems with Applications* 117 (Mar. 2019), pp. 345–357. ISSN: 09574174. DOI: 10.1016/j.eswa.2018.09.029.

[48] scikit-learn. *Precision-Recall*. URL: https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html (visited on 04/12/2022).

[49] Sami Smadi, Nauman Aslam, and Li Zhang. "Detection of online phishing email using dynamic evolving neural network based on reinforcement learning". In: *Decision Support Systems* 107 (Mar. 2018), pp. 88–102. ISSN: 01679236. DOI: 10.1016/j.dss.2018.01.001.

[50] Gianluca Stringhini et al. "B@bel: Leveraging Email Delivery for Spam Mitigation". In: *21st USENIX Security Symposium (USENIX Security 12)* (2012), pp. 16–32.

[51] Shan Suthaharan. "Support vector machine". In: *Machine learning models and algorithms for big data classification*. Springer, 2016, pp. 207–235.

[52] Choon Lin Tan et al. "PhishWHO: Phishing webpage detection via identity keywords extraction and target domain name finder". In: *Decision Support Systems* 88 (Aug. 2016), pp. 18–27. ISSN: 01679236. DOI: 10.1016/j.dss.2016.05.005.

[53] Chandra Thapa et al. "Evaluation of Federated Learning in Phishing Email Detection". In: *arXiv preprint arXiv:2007.13300* (July 2020). URL: http://arxiv.org/abs/2007.13300.

[54] Ray Tomlinson. *The First Network Email*. 2009. URL: https://www.raytheon.com/sites/default/files/news/rtnwcm/groups/public/documents/content/rtn12_tomlinson_email.pdf.

[55] Joaquin Vanschoren. "Meta-Learning: A Survey". In: *Automated Machine Learning*. Springer, Cham, Oct. 2019, pp. 35–61. URL: http://arxiv.org/abs/1810.03548.

[56] Wei Wei et al. "Accurate and fast URL phishing detector: A convolutional neural network approach". In: *Computer Networks* 178 (Sept. 2020). ISSN: 13891286. DOI: 10.1016/j.comnet.2020.107275.

[57] Wikipedia. *Incremental decision tree*. URL: https://en.wikipedia.org/wiki/Incremental_decision_tree (visited on 11/22/2021).

[58] Jan Willem. *Blue underwater scene of a phishing hook trying to catch "@" signs, symbolizing email phishing*. Cover image. 2021. URL: https://stock.adobe.com/nl/contributor/208806639/janwillem?load_type=author&prev_url=detail.

[59] Yuxin Wu et al. *Detectron2*. https://github.com/facebookresearch/detectron2. 2019. (Visited on 11/22/2021).

[60] Guang Xiang et al. "CANTINA+: A feature-rich machine learning framework for detecting phishing web sites". In: *ACM Transactions on Information and System Security* 14 (2 Sept. 2011). ISSN: 10949224. DOI: 10.1145/2019599.2019606.

[61] Cihang Xie et al. "Adversarial Examples for Semantic Segmentation and Object Detection". In: *Proceedings of the IEEE international conference on computer vision* (2017), pp. 1369–1378.

[62] Penghui Zhang et al. "CrawlPhish: Large-scale Analysis of Client-side Cloaking Techniques in Phishing". In: *2021 IEEE Symposium on Security and Privacy (SP)* (2021), pp. 1109–1124.

[63] Xin Zhou and Rakesh m Verma. "Phishing Sites Detection from a Web Developer's Perspective Using Machine Learning". In: *Proceedings of the 53rd Hawaii International Conference on System Sciences* (2020).