Developing a Phishing Learning and Detection tool

Stephen Waddell

MInf Project (Part 1) Report

Master of Informatics
School of Informatics
University of Edinburgh

2019

Abstract

In this project, I outline the design of a Phishing Learning and Detection tool (CatchPhish) focused on detecting malicious URLs. This incorporates existing approaches to tackle phishing and employs embedded training to protect and inform users about this increasing security concern. As part of the first year of this project, the intention of this work was to the design the CatchPhish system with a focus on user interaction. Research was conducted through user interviews to understand how much knowledge users have of malicious phishing, and find the ideal point of user interaction. As part of the design of the tool, an algorithm to analyse URLs was developed and subsequently evaluated by an expert in the field. The completed system was evaluated and found to be usable. Users across each of the conducted studies highlighted a consistent demand for such a tool. This lays ground for further work to complete the implementation of the decision-making aspects of the tool, and establish the efficiency and accuracy of the system.

Acknowledgements

First of all, I would like to thank my supervisor Kami Vaniea for all her support and encouragement throughout this project.

I would also like to thank Kholoud Althobaiti for her continual feedback and thank her, along with Sara Albakry, for their participation in the expert evaluations.

A particular thank you to Rusab Asher who helped with the interview analysis, and for his encouragement throughout the project.

To all the participants who participated in one of the many user studies conducted throughout the project - this would not have been possible without you - thank you.

Contents

1 Introduction **9**
- 1.1 Overview . . . 9
- 1.2 Project specification . . . 10
  - 1.2.1 Results and Accomplishments . . . 10
- 1.3 Report structure . . . 11

2 Phishing **13**
- 2.1 Phishing Variations . . . 13
  - 2.1.1 Spear Phishing . . . 14
- 2.2 Phishing Delivery Vehicles . . . 14
  - 2.2.1 Attachments . . . 15
  - 2.2.2 URLs . . . 15
  - 2.2.3 Example of a URL attack . . . 16
- 2.3 Common indicators of phishing attacks . . . 16
  - 2.3.1 Email Specific indicators . . . 17
  - 2.3.2 Common URL Manipulation Tricks . . . 18
  - 2.3.3 Domain Indicators . . . 18
  - 2.3.4 Page Indicators . . . 19
- 2.4 Other Phishing difficulties . . . 19
- 2.5 Summary . . . 20

3 Related Work **21**
- 3.1 Usable Security . . . 21
- 3.2 Theoretical Phishing research . . . 22
  - 3.2.1 Automated Phishing Detection . . . 22
  - 3.2.2 Automated Security Indicators . . . 23
  - 3.2.3 Training users . . . 25
- 3.3 Existing Phishing tools . . . 26
  - 3.3.1 Further Anti-Phishing Tools . . . 27
- 3.4 Summary . . . 27

4 Requirements Gathering **29**
- 4.1 Initial Concept Design . . . 29
  - 4.1.1 Browser Choice . . . 30
  - 4.1.2 Identifying the Userbase . . . 30
- 4.2 Interviews . . . 31

4.2.1 Method Selection and Reasoning . . . 31
4.2.2 Methodology . . . 32
4.2.3 Results . . . 34
4.2.4 Likelihood to use the tool . . . 36
4.2.5 Recommendation . . . 37
4.3 Requirements . . . 38
4.4 Summary . . . 39

5 Design **41**
5.1 Prototyping the User Interface . . . 41
5.2 Back-end Design Proposal . . . 43
5.2.1 Selecting the Algorithm Type . . . 43
5.2.2 Decision-Making Algorithm . . . 44
5.2.3 Malicious Heuristics . . . 45
5.2.4 Safety Thresholds . . . 46
5.2.5 Data Sources . . . 47
5.3 Expert Design Evaluation . . . 47
5.3.1 User Interface . . . 47
5.3.2 URL Decision making . . . 48
5.3.3 Evaluation of Design Proposal . . . 48
5.4 Final Design . . . 49
5.5 Summary . . . 49

6 Implementation **51**
6.1 Project Timeline . . . 51
6.2 System Overview . . . 52
6.2.1 Components and Interaction . . . 52
6.2.2 Choice of Technologies . . . 52
6.2.3 Understanding Chrome Extensions . . . 55
6.3 Extension Infrastructure . . . 56
6.3.1 Content Script . . . 56
6.3.2 Background Script . . . 57
6.4 Front-end and User Interaction . . . 58
6.4.1 Research Interface Implementation . . . 59
6.4.2 Choice of Extension UI Elements and Intervention . . . 62
6.4.3 User support and understanding . . . 65
6.5 Decision-Making Server . . . 67
6.5.1 Implemented Heuristics . . . 69
6.5.2 Unshorten Links . . . 70
6.5.3 Chrome Extension API . . . 70
6.6 Additional Considerations . . . 71
6.6.1 Efficiency and Security . . . 71
6.7 Summary . . . 72

7 Evaluation **73**
7.1 Study Preparation . . . 73
7.2 Demonstration with System Usability Scale . . . 74

7.2.1 Methodology . . . 74
7.2.2 Results . . . 75
7.3 Think Alouds . . . 76
7.3.1 Preparation . . . 76
7.3.2 Methodology . . . 76
7.3.3 Results . . . 77
7.4 Expert Evaluation . . . 78
7.4.1 User Interface . . . 79
7.4.2 Implemented Backend . . . 79
7.4.3 Further Advice . . . 80
7.5 Discussion . . . 80
7.5.1 Requirements of the tool . . . 81
7.6 Summary . . . 81

8 Further Work **83**
8.1 Decision-Making Components . . . 83
8.2 User Interface Improvements . . . 84
8.3 Longitudinal Study . . . 84
8.4 Summary . . . 85

9 Conclusion **87**
9.1 Overview . . . 87

Bibliography **89**

A Interview Materials **95**

B Interview Results **105**

C Additional Paper Designs **113**

D Phishing Heuristics Proposal **115**

E System Usability Scale Survey **135**

F Think Aloud Materials **137**

G Think Aloud Data Feature Codes **143**

H Think Aloud Qualitative Data **145**

Chapter 1

Introduction

1.1 Overview

Phishing emails are a routine occurrence for anyone with email in the Internet age. Masking themselves as reputable companies such as Paypal, using devious means such as the use of company logos and standards, malicious actors, again and again, attempt to lure in individuals from a wide range of technical shades. Phishing ranges extensively in sophistication: from mass-produced misspelt requests for overly specific details to sophisticated spear phishing attacks focused on the details of the individual. The ability of phishing attacks to innocuously harvest your private credentials can leave you mercilessly exposed in our data-intensive world. All it takes is for a user to make the critical mistake of clicking on a single malicious link. Through a simple mistake, a user exposes themselves and their data from anything from drive-by downloads, cross-site scripting attacks to the harvesting of their details in an innocuous web form.

Phishing has two main delivery vehicles: emails and websites. Emails being the foremost of these, are most classically associated with phishing. These often include malicious URLs to direct users towards maliciously crafted content. By obfuscating the real destination of a URL through a few simple manipulations, users can quickly find themselves on unknown and insecure ground. Therefore it is vital to tackle this massive worldwide problem. In the United Kingdom (UK) alone, phishing is expected to cost the UK economy as much as £280 million per year [42]. This is encouraging companies such as Google [27] to look into the future of Uniform Resource Locators (URLs) themselves [84].

To tackle the problem of phishing, my project has been focused on tackling the malicious URLs included in them as "more than 75% of phishing mails include malicious URLs to phishing sites" [16]. Existing techniques to handle URLs involve automated phishing detecting (mainly employing machine learning techniques), user training (the best results of which are gained from embedded training) and automated security indicators (providing information to help the users decide). I aim to create a system which incorporates aspects of these techniques, to inform and protect users from malicious

phishing.

1.2 Project specification

To solve this problem, I have been building a user-focused tool which seeks to catch problematic URLs before they reach their full malicious potential. The tool that I have developed is a Phishing Learning and Detection tool, built for the Chrome platform in the form of an extension called Catch-Phish. It classifies URLs into one of three safety states using a combination of natural language processing and knowledge acquisition, before providing users with the necessary information to understand how this classification was derived. It helps to train them in URL safety by presenting this information at critical points of intervention.

The primary motivation behind this project is the lack of tools which purposefully prevent users from visiting malicious URLs and more crucially inform them of why they have been prevented. This is important due to the significant average availability time of phishing links, which according to Canova et al. [11], was 32 hours and 32 minutes in the first half of 2014. Machine learning techniques are very successful in matching established patterns of URLs but not as successful at identifying new URL variations, which means malicious URLs can go sometime before being detected. The lack of user knowledge of phishing also encourages users to ignore warnings when presented to them. In the UK for example, only 72% of technology users had heard of phishing as a term despite 95% of organisations saying that they train end users [60]. For these reasons, it is important to train users to detected phishing themselves.

As the first year of my Minf project, the focus this year was working on both designing the tool and implementing and evaluating the means for how the user would interact with the tool. The project has involved a welcome and generous amount of advice from a PhD student who is an expert in phishing, computer security and URLs and is working on a similar project. This student has produced a lot of the research referenced in this report. However, I make a clear distinction between the work done by myself and this student throughout the report.

1.2.1 Results and Accomplishments

As part of this project, there were several significant pieces of work which are outlined below:

- Completed a literature review of the subject
- Conducted extensive user interviews to find the ideal point of user intervention for the tool
- Developed an algorithm to classify each URL with one of three designed status values

- Developed the infrastructure of the tool itself including implementing the custom extension user interactions
- Implemented a detailed URL analysis user interface based on current research, using modern web technologies
- Extended the user's visited webpages with the dynamic insertion of visual elements to represent the status of each link in the page
- Implemented the web redirection features with the real-time analysis of user web traffic
- Used secure programming techniques to incorporate the ability for the users to have personal URL safety lists securely
- Evaluated the overall usability of the finished tool using a triangulated approach

Throughout each of the user studies I conducted within this project, there has been a strong demand for the features outlined in the tool, before and after their implementation. The completed tool was regarded in the final evaluations as being easy to use, with the implementation of URL analysis user interface being particularly commended.

1.3 Report structure

This report covers the progress I made this year. The remaining 9 chapters are structured as follows:

Chapter 2: presents previous approaches to phishing, their advantages and how they differ compared to my approach.

Chapter 3: presents an overview of phishing and explains common indicators of phishing and malicious URLs and how these are useful to detect them.

Chapter 4: discusses the gathering of requirements for the system including the interviews conducted for this purpose.

Chapter 5: describes the iterative design process used to design the tool based on the required requirements.

Chapter 6: details how I built the system, including the features included in the tool and the final design.

Chapter 7: critically evaluates the overall usability of the system using a triangulated evaluation approach and discusses the results in relation to the design requirements established in Chapter 4.

Chapter 8: discussed the further work required for this project.

Chapter 9: conclusion and overview of the project so far.

Chapter 2

Phishing

As mentioned in the introduction to this report, phishing is a means of trying to acquire private and confidential details from people, by luring them through deceptive means. This can often occur as a result of a malicious actor pretending to be an established company or organisation that the users might be familiar with. Phishing attacks are deployed through a number of different means including emails and phishing websites. Links are the most common means of directing users to malicious resources, but attachments in emails can also be used to target users computers more directly.

2.1 Phishing Variations

There are several variants of phishing, [69] which range in sophistication, but are each used to exploit potential victims. Some of these phishing variants include:

- Deceptive Phishing - is the most common type of phishing scam, by which fraudsters impersonate a legitimate company and attempt to steal peoples' personal information or login credentials. This often hinges on how closely the attack email resembles a legitimate companys official correspondence, and is largely deployed on-mass to multiple recipients in a blanket approach.
- Spear Phishing - is a more individually focused phishing attack, personalised to each recipient with details such as their name, position and company. The primary aim is to trick the recipient into believing the attacker has a personal connection with them.
- Whaling - a spear phishing attack focused on attacking, or "harpooning", a CEO to capture their private details in a similar way - by mimicking a sender whom the victim knows. The content of the email is relevant to the victim, intended to not trigger any suspicion from the victim.
- Pharming - exploits issues in existing Internet infrastructure, rather than baiting victims like other phishing attacks. It achieves this by exploiting the Domain Name System (DNS) used within the Internet to convert alphabetical website

names to their related Internet Protocol addresses (which are used to specify the location of computing devices on the Internet). By poisoning a popular DNS cache, attackers can direct users trying to connect to genuine sites to their own malicious ones, which can be very difficult for users to detect.

- Online Drive phishing - some attacks are specialised according to an individual company or service, such as the Dropbox [19] and Google Drive [28] services. These are highly popular file storage backup services with millions of users respectively. Phishing attacks aim to exploit users of these systems by luring them into entering their credentials into similar looking sites. This can give them access to all of a users files, documents, spreadsheets for instance.

Spear phishing for instance, is one variant of phishing that even those with extensive training, struggle to identify.

2.1.1 Spear Phishing

Kang Leng Chiew et al. [13] outline the goal of spear phishing as being the same as deceptive phishing: to lure the victim into clicking on a malicious URL or email attachment, so that they will hand over their personal data. Spear phishing is especially commonplace on social media sites like LinkedIn [40], where attackers can use multiple sources of information to craft a targeted attack email.

Spear phishing has become the popular choice (Nagunwa, 2014)[44] by phishers over the conventional phishing using mass and random email phishing. This popularity is because of the high success rate compared to the more conventional phishing approaches (Krombholz et al., 2015)[39]. The success rate of spear phishing is high because internet users will normally trust emails from the website of an organisation that they have used before or have an account with [13].

Spear phishing can expose victims to further attacks such as an Advanced Persistent Threat (APT): a long-term low-profile attack that infiltrates a specific target of interest and uses vulnerability exploits or malware to achieve a set of objectives such as espionage or sabotage (Symantec, 2011)[66].

53 percent of IT and security professionals who responded to the Wombat survey reported their organisations have experienced these more advanced, targeted spear phishing attacks in 2017 [60]. Therefore spear phishing is an example of the increasing amount of sophisticated phishing attacks that users regularly face. It can require a lot of technical skill to understand how to avoid these attacks.

2.2 Phishing Delivery Vehicles

Emails are often regarded as the main delivery vehicle of phishing attacks. The typical form of these attacks involve a spam email appearing to be from a reputable company that the user might be familiar with. With 54.6% of all email being spam, it can be

hard for users to differentiate the malicious phishing emails from the more innocuous marketing attempts, with the average user receiving around 16 malicious emails per month [65]. Malicious emails can be caught by the users' email clients or spam filter, but this has a varying effectiveness [83, 14, 64] as it is often based on established spam email patterns. This means that some malicious email are still able filter passed this protection.

Phishing attacks can, however, be spread through any online communications means that allow for the use of links or attachments. For instance, social media sites are another example means of phishing attacks. This can occur innocently, such as in the case of a family member spreading what they think is a great deal site for cheap holidays - and infecting the family with malicious software. These social media sites can then be hijacked to help spread these links further, by the harvesting and exploitation of a victim's credentials, which can cause a further loss of personal information to an increasing number of victims.

2.2.1 Attachments

An email attachment is a computer file sent along with an email message. One or more files can be attached to an email message, and be sent along with it to the recipient. This is typically used as a simple method to share documents and images [82]. These can use however we be used as a vehicle for phishing attacks. By downloading one of these files, you could be downloading malicious software attached to it which can be used to further exploit your system, according to the attacker's original intention. In particular, invoices purporting to be from reasonable companies, account for 15.9% of all malicious attachments according to [65].

2.2.2 URLs

A URL is a Uniform Resource Locator, which is often colloquially called a web address or link, and is used as a reference to a web resource. A URL specifies a web resource's location on a computer network and a mechanism for retrieving it. URLs are a very flexible and adaptable technology, even though they most commonly used to reference web pages. They are also used for multiple other purposes such as file transfer (ftp), email (mailto) and database access (JDBC).

They are able to be used for many purposes due to their flexible structure, outlined in figure 2.1.

URL adaptability means that they require expertise to truly understand them, even though their presence is ubiquitous throughout the internet. Internet Users, however, have been shown to be lacking this knowledge, having difficulty with reading URLs [5, 4, 11, 62]. Even after having extensive training in the subject users can still fail to notice visually deceptive manipulations [18]. This leaves internet users susceptible to attack without persistent or accessible URL knowledge.

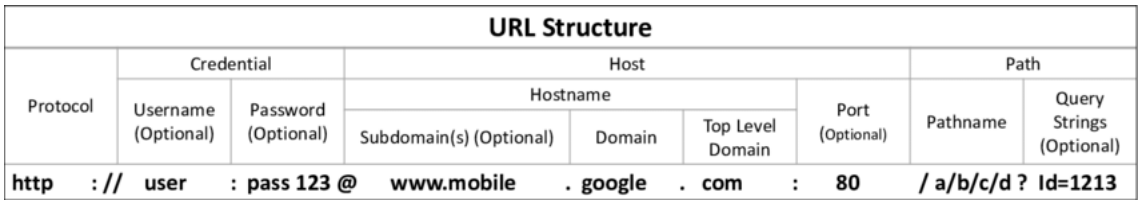

Figure 2.1: URL structure example [5]

2.2.2.1 URL Manipulation Attacks

URLs are used in emails and beyond for the purposes of Phishing attacks. Attacks which involve the manipulation of URLs are known as URL manipulation attacks, and can occur within both phishing and other attacks which employ URLs.

URLs are more commonly used in phishing attacks than malicious attachments. The latest Quarterly Threat Report by proofpoint highlighted: “the pendulum of malware delivery mechanisms in email continued to swing towards URLs; malicious URLs outnumbered attachments ... by over 370%.”[55]. This is why the primary focus of this project is on URLs due to their greater use within and without phishing attacks, and thus the wider scope of the project to help reduce phishing and URL manipulation attacks as a result.

2.2.3 Example of a URL attack

For example, suppose a victim clicks a malicious phishing link presented to them on a seemingly familiar website such as Facebook, such as the link below:

http://www.facebook.pr0file.com.sq/k23IDCWs

An example attack such as the one discovered by Wang Jing [67] might occur; on clicking a malicious link, a popup window from Facebook could suddenly appear and ask the victim would to authorise the app. If the victim chooses to authorise the app, the popup could gain access to the victim’s personal sensitive information through Facebook: such as the user’s email address, birth date and contacts. Worse still, in case the user has consented to giving the app greater privilege, the attacker may possibly control and operate the users account. Even if the victim does not choose to authorise the app, he or she will still get redirected to a website controlled by the attacker. This could potentially further compromise the victim [67]. One of the causes of the described vulnerability is a lack of validation of the redirect URL.

2.3 Common indicators of phishing attacks

There are multiple indicators of phishing attacks, but these can depend on the type of phishing delivery vehicle used. An indicator is used as a gauge or measure of whether

a phishing attack has a occurred, and examples of these are discussed in the subsequent sections of this chapter.

2.3.1 Email Specific indicators

There are a range of email specific indicators depending on how sophisticated the attack is. For example, looking at the email sender and matching that with known email or details of the company might be useful for lower quality deceptive phishing emails. However, more sophisticated spear phishing attack can employ tactics such as email spoofing [59], so that the message appears to have originated from a non-malicious source.

In addition, a common email specific indicator is to use a strong or threatening tone to make the users concerned about the impact of the email. This is often in combination with a presented time limit to expedite their use of the accompanying link or attachment. For example, *"your account will be suspended within 24 hours unless you verify your account details accessed by clicking here"*. An example of this can be seen in figure 2.2.

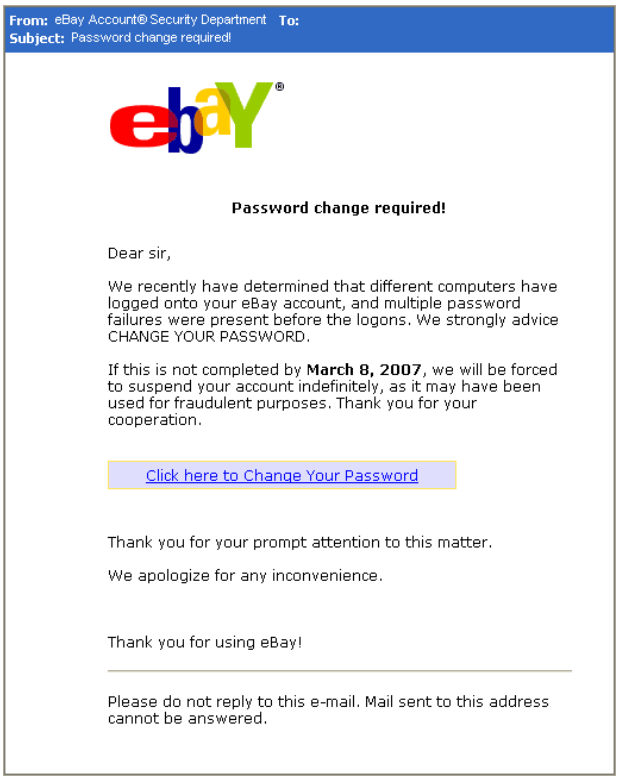

Figure 2.2: Phishing email example [51]

Lower quality emails also include indicators that people are more familiar with. For instance, poor spelling and grammar are common indicators. This is in stark contrast to the emails from very large companies such as PayPal, who typically spend a lot of time vetting their email content. In some cases, other languages can be used in the

email along with poor or low-quality images. This is often a clear indicator to users that they are not receiving a link from the intended company.

2.3.2 Common URL Manipulation Tricks

URLs can be manipulated in multiple ways in order to trick users. As discussed in section 2.2.2, these manipulations can often be difficult to identify, even for those with extensive training.

One common set of tricks is called *"mangle"*: this is where a brand or company name has letter substitution, misspelling or non-ASCII characters that appear to be similar to their English counterparts [5]. The use of non-ASCII characters means this is very difficult to be picked out at a brief glance and since users do not typically spend a while focusing on URLs in the page [72], these would be hard to detect.

URL manipulations can also occur to confuse the user intentionally. For instance, a common set of tricks is called *"obfuscate"*. These tricks aim to confuse the user by substituting the URL domain with an IP addresses in place of the domain, or shortened links [5]. Link shorting, for instance, does not always have to have a malicious intent, though. Service provided by companies such as Bitly [8], can help to reduce the length of URLs for users to more easily sharing for example.

These manipulation tricks are difficult to avoid and often need a lot of experience when trying to understand them. These tricks could be a core focus in developing a theoretical anti-phishing tool, and this tool could specifically allow users to more clearly understand these tricks.

2.3.3 Domain Indicators

There are a number of phishing indicators which are specifically related to the domain name of a site. A domain name is a label which identifies a network domain - a unique group of computers that form part of a central administration or authority. These domain names are managed by the Domain Name System (DNS). The domain of a site can be embedded in a site's URL, as can be seen in figure 2.1 on page 16.

One of the indicators of phishing related to domains is the age of the domain itself. Therefore the days of domain registration can be looked at, in information stores such as the WHOIS [80] domain database. "A Whois domain lookup allows you to trace the ownership and tenure of a domain name. Similar to how all houses are registered with governing authority, all domain name registries maintain a record of information about every domain name purchased through them, along with who owns it, and the date till which it has been purchased."[80]

As phishing sites tend to be newly created, information stores such as WhoIs can be queried to get the creation date of the website. This can then be evaluated against thresholds for the typical creation date of phishing sites.

Therefore domain indicators are an important indicator of whether a site is a phishing site. Since these indicators are not apparent to the user when inspecting sites, these indicators could be highlighted to better inform users.

2.3.4 Page Indicators

Page features use information about pages which are calculated using reputation ranking services. These are some of the most useful indicators of a URLs safety if they show a site is popular, but can equally be an indicator of phishing where a site is shown to be unpopular. In this sense, they give information about how reliable a site is.

For instance, one page indicator is the relative popularity of a website which can be determined by looking at information by using Alexa Top Sites [3] to get a rank of the most popular domains [41, 86]. Popular sites tend not to be phishing sites since users do not tend to revisit a phishing site. In this sense, the highly popular sites in a list such as Alexa's Top Sites can be used as a whitelist; whereas the most unpopular sites from this list (or no inclusion in the list) can be taken as being a malicious site.

Therefore by highlighting these page indicators to users, and using this information in the calculation of automated security indicators, a phishing detection tool could increase the users' contextual knowledge of phishing.

2.4 Other Phishing difficulties

Another common trick employed in phishing is to make the displayed text for a link suggest a reliable destination when the link actually goes to the phishers' site. This is also known as covert redirect or cloaking. Many desktop email clients and web browsers will show a link's target URL in the status bar while hovering the mouse over it - called a mouseover. This behaviour may in some circumstances be overridden by the phisher [12] which can lead to users visiting malicious sites without their prior awareness.

A further attack that can be employed against a victim is a cross-site scripting attack. This involves an attacker using the flaws in a trusted website's own scripts against the victim. These types of attacks are particularly problematic because they direct the user to sign into a genuine service such as a bank, where everything from the web address to the security certificates appear correct. In reality, the link to the website is specifically crafted to carry out the attack, making it very difficult to spot without specialist knowledge. This type of attack was used in 2006 against PayPal [45].

A drive-by download is an additional example of an attack that can occur as a consequence of a phishing attack. This refers to the unintentional download of malicious code to a target's computer which leaves them exposed to further attacks. Drive-by-download malware often uses small pieces of code designed to slip past simple defences and go largely unnoticed. This code is often very simple as its main purpose

is only to contact another computer to introduce the rest of the malicious code. Often the malicious code is distributed by compromised websites, so simply visiting a site without the appropriate security can leave user exposed to this attack [38].

2.5 Summary

This chapter underlines what phishing is, and provides an explanation of the different vehicles used to deliver phishing attacks. It discusses the indicators used to identify these attacks and the depth of technical knowledge that is often required in order to understand what constitutes a phishing attack. Further attacks that can occur as a result of phishing are outlined as an example of the potential sophistication of these attacks.

Chapter 3

Related Work

The chapter looks at related work to the phishing tool and discusses how the tool expands and builds on this work through the course of the project.

3.1 Usable Security

Usable security as a field focuses on the juncture between human-computer interaction and computer security, with a focus on the human factors of computer security and privacy. Researchers working in the computer security field often produce necessary security tools intended to protect users, but these can have a bounded usefulness for actual users due to their limited usability. An example of this is the PGP 5.0 encryption study [79] in which a selection of computer science students were not able to adequately encrypt an email using the given security tool. This demonstrates the limited benefit in creating a security application that is not usable. Papers such the one presented by Cormac [32] expand on this, by discussing how the creation of further tools, and specifically the security terms associated with them, can actually lead to users' confusion by increasing the amount of terms they need to be familiar with.

An understanding of how users interact with computer security systems is a central aspect of Usable security. This is the focus of the Human-in-the-loop framework proposed by Cranor [15], which discusses how security systems should strive to remove humans from security systems wherever possible, due to their general unreliability. However, where they cannot be removed from the system design, security failures should be analysed by system engineers to allow users to understand and engage with the security system in the most effective way.

This should not, however, lead to a perception that users are somehow the enemy of secure computing systems. Adams et al. [1] found that whilst users do indeed compromise computer security mechanisms, this is "often caused by the way in which security mechanisms are implemented, and user's lack of knowledge". Instead, through adequate training, users can be useful components of a security system as discussed by Pfleeger et al. [52]. This is particularly important as humans are often more capable at

spotting unique and unknown patterns than machines.

An understanding of usable security is necessary to effectively design a security system if it is to be usable for end users. For instance, a security system such as phishing learning and detection tool would have to consider the human-in-the-loop in order to design a usable system which prevents users from being affected by malicious phishing.

3.2 Theoretical Phishing research

There are many different approaches to tackling phishing. The common element between these approaches is the analysis of phishing indicators. The main difference between these approaches is how they use this analysis: whether that is user training or calculating the likelihood of the URLs being malicious using machine learning techniques.

The amount of user interaction these approaches have is varied. For example, approaches involving automated phishing detection may have no user interaction whatsoever. Approaches such as these which have limited user interaction, deprive the user of learning opportunities. These learning opportunities are needed to bridge the malicious URL availability window - the amount of time it takes for a URL to be caught by an automated detection system. According to Canova et al. [11], this time window was around 32 hours and 32 minutes in the first half of 2014. This means that users can be affected by these malicious URLs during this time, before general security tools can remove or block them. This availability window can be caused by the failure of these automatic approaches to detect newly created URLs which do not conform to existing patterns. The only defence in this case are the users themselves. Therefore it is important users have the appropriate knowledge to protect themselves from the threat of malicious phishing URLs.

Each of the approaches to tackle phishing discussed in this chapter has it's unique advantages and disadvantages. The goal of my work is to understand how the benefits of these approaches may be combined into a single phishing learning and detection tool.

3.2.1 Automated Phishing Detection

Automated phishing detection is an approach favoured by large organisations and companies. This approach employs machine learning techniques to analyse URLs to discern if they are malicious.

These techniques can be used to populate stored blacklists. These are extensive databases of phishing links which are known to be dangerous. These databases are often maintained independently by companies, with some publicly available for querying. One of the significant benefits of using blacklists is the reduction in computation time needed to process whether each encountered URL is malicious. Blacklists tend to be highly reliable when considering the URLs present in them. Since they are not complete lists

of all malicious URLs, they can not be fully relied on. This means that URLs not present within blacklists still need to be analysed.

The accuracy of automated approaches is greatly improved by access to large datasets. For instance, Google utilise the massive datasets they have access to from their search engine to produce a classifier that works on noisy data with a 90% accuracy [78]. Their approach utilises URL feature extraction and the fetching of page content to match the defined heuristics for their classifier.

The main concern with these automated approaches, particularly from a usability perspective, is that these approaches can result in false warnings that decrease user confidence in the effectiveness of the prevention systems. This is because such systems do not have a 100% accuracy. The false warnings that occur as a result, can cause users to ignore future warnings [61]. Therefore automated approaches such as machine learning are not effective as a singular indicator of phishing attacks.

3.2.2 Automated Security Indicators

A further approach to tackling phishing, is by using automated security indicators. The basis of the automated security indicator approach is to feed information about security practices back to the user so they can make decisions about the security status itself. The idea behind this approach is that users can pick up on information that machines cannot. The automated security indicator approach can incorporate features such as informative mouseovers or the highlighting of important information for the users.

An example of this is presented in the Faheem slackbot [5], which breaks down a URL into its various components and presents this information to users. This tool presents the URL information to users in a standard format with contextual information about what each URL component might mean. The concept underlying this tool, which is indicative of the general approach, is that this allows users to make an educated decision about the safety of the URLs analysed by the tool.

A further example of automated security indicators is given in the URL report research [4]. This research presents a detailed breakdown of how a URL's different components can be presented to a user. This allows for extensive contextual information about each URL element, whilst highlighting the most important aspects of a given URL to a user. The report was created with the use of focus groups in order to be an effective and useful presentational tool. It breaks down the information about URLs into three primary sections: summary, URL manipulation tricks and facts about the URL. For each fact, there is a clear explanation of why that fact is significant. This is presented alongside an example of the URL component itself, to make the connection between the fact and URL component clear to the user. The report itself expands on the Faheem Slackbot by including further information such as the level of encryption and the domain and page popularity. This URL report has not been implemented as part of an existing tool. A theoretical phishing detection and learning tool could incorporate this URL analysis design. This would be beneficial since this report has already been extensively designed with user feedback.

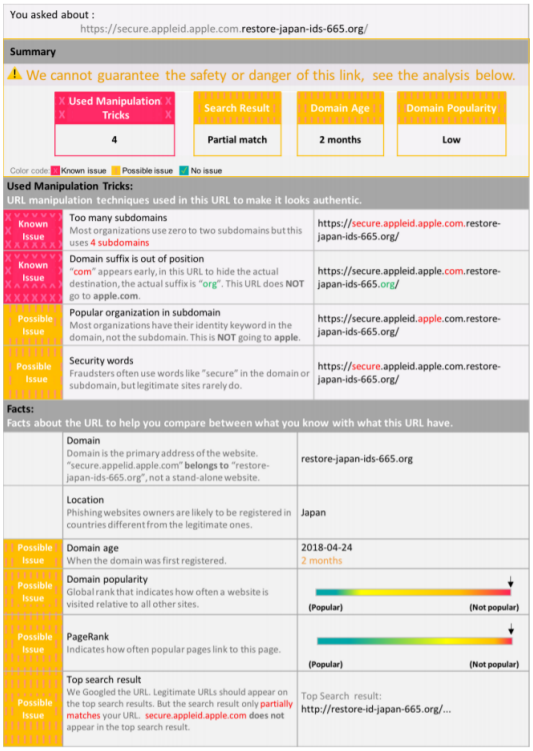

Figure 3.1: Usable URL Report [4]

Established security toolbars such as Netcraft's are an additional example of automated security indicators. This tool is designed to present information about the quality of the site to indicate to users it is phishing or not.

Figure 3.2: Netcraft extension interface [46]

These security toolbars are often not designed to convey information clearly to users, and essential heuristics such as page site rank can be ignored by users as a result. This can be due to a lack of contextual information explaining why this information is important to users. Wu et al. [85] discuss the usability of these security toolbars in their paper. Netcraft's security toolbar, as seen in figure 3.2 is one example included in this paper. The paper itself raises future design principles for anti-phishing tools, drawn from research into the usability limitations of the example security toolbars themselves. The design principles suggest "active interruption like the pop-up warnings [are] far more effective than the passive warnings" but they "should always appear at the right

time with the right warning message" in order to ensure user trust in the system.

Active interruption is another notable feature which could be included in a theoretical phishing detection and learning tool. This could be used to prevent users from visiting malicious websites in the first place, rather than just warning them of these sites. This feature combined with a clear information breakdown of the URL components, as outlined in the URL report, might make an effective tool for phishing protection.

3.2.3 Training users

Training users is a critical approach of tackling phishing, as it can improve the ability of users to make effective decisions when faced with phishing attacks. This touches on Usable security fundamentals as users are often an under-considered element of computer security. It is advised that were users cannot be removed from the application loop; systems are designed to make information as clear as possible to them. This is where training users can be beneficial.

There are multiple approaches to training users and many different tools designed to help with this. Facets of ideal training approaches are: engaging the user, presenting relevant information to the subject and having clear learning objectives. Training through the use of educational games is one example approach. This approach is often intended to engage the user by presenting information in fun environment. This, however, is subject to the quality of the game. One common difficulty with this, and training in general, is the user's ability to retain information taught to them after some arbitrary period of time. After learning new information, users tend to show increased knowledge and skills on the subject immediately afterwards, but this is shown to deteriorate over time. This is a concern with computer security applications especially as it is essential that users continue to have the knowledge and awareness that allows them to adapt to threatening issues as they appear.

Studies related to the phishing area focused on training users include the NoPhish app [11]. This is an app intended to teach users about phishing using informative pieces of information. As a result of the longitudinal study included in the paper, they found that users were more successful after the teaching, but their success rates dropped 5 months later when tested on the same material again.

A theoretical anti-phishing tool could inform users about the components of malicious URLs, and the reasoning behind why these components are malicious. Where this theoretical tool could focus, is on embedded training which is intended to help users retain the taught information for a more extended period by continuing to teach them at key moments.

3.2.3.1 Embedded Training

The aim of embedded training is to teach users in the moment they make an error, about why that error has occurred. For instance, when users are making an error,

this type of training teaches them by presenting information to them about why these errors have occurred. This helps to improve user retention times by creating memory anchor points which help users recall the learned information when faced with similar situations to the learning experience. For instance, Jerry clicks on a malicious URL but is prevented from reaching it and presented with information that it has too many sub-domains which indicates why this URL is malicious. In the future, Jerry is more cautious about clicking on URLs and watches out for how many sub-domains are in the URL.

One problem with embedded training, despite success rates, is that is is very hard to deploy in real life situations. In corporate environments for instance, this is because it is hard to set-up a scenario for embedded training where the employees do not know it is occurring. The training therefore fails to train users appropriately as they adapt their behaviour to the training environment. It is also difficult to create the environment in which these kinds of training can occur without a lot of set-up and intervention into typical user routines.

A theoretical anti-phishing tool could incorporate these elements by presenting information to users about the mistakes they make when clicking on malicious URLs. The tool could give users the choice of proceeding to the website regardless in the case that it has made a mistake. By incorporating a feature such as a whitelist (a list of safe sites) the accuracy of such a tool could be improved by allowing the tool to adapt the user input, to correct situations when the tool has made a miss-classification.

3.3 Existing Phishing tools

There are some existing phishing tools. The majority of these are incorporated into general computer security-focused software such as anti-virus software. This means that their particular focus on the user on this topic is limited. These comprehensive software packages also tend to focus on providing the user with a sparse amount of high-level information. This is designed to keep users engaged with the tool by demonstrating its functionality whilst encouraging the users to continue to engage with it. This is coupled with the fact that these tools are often paid for and the average user tends to have limited knowledge of computer security in general.

Tools specific to chrome fall into a number of categories:

- general computer security extensions that are also incorporated
- tools build into chrome itself
- ad-blocker extensions
- specific phishing extensions

3.3.1 Further Anti-Phishing Tools

An example of a phishing tool built into a web browser itself are Google site warnings, as can be seen in figure 3.3. These are presented to prevent users from visiting malicious sites, by using the information calculated as part of Google's automated phishing detection. When visiting links that are malicious, this warning can sometimes be thrown to indicate that the subsequent site is malicious. This is useful as an intermediary to prevent users from going to the site, and prevents them being immediately affected by the malicious site. However, the usefulness of this warning is limited by the accuracy of Google's aforementioned classifier work. This can reduce users trust in these warning and encourage them to ignore them.

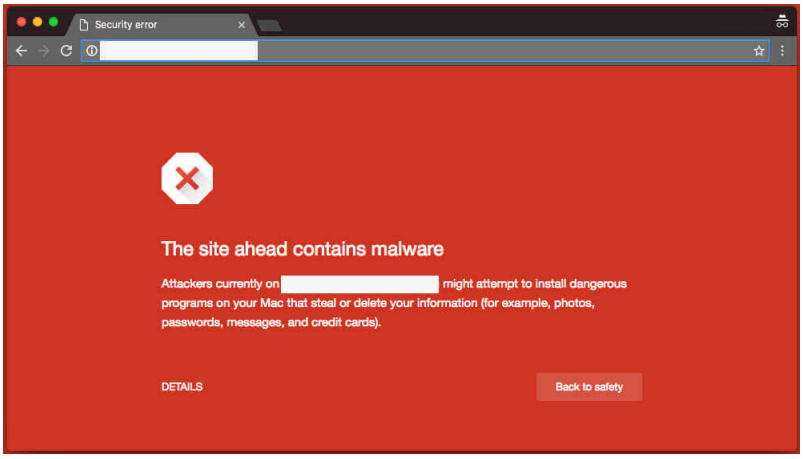

Figure 3.3: Google site warning [29]

To improve on this approach, a theoretical anti-phishing tool could present the details of why the user has been blocked to help users build an understanding of why they have been blocked. Since the design for Google's site warning is well known by users of the Chrome browser, a theoretical anti-phishing tool could use a similar design to keep this consistent with what users are already familiar with. This would help to make the intention of this page clearer to potential users.

3.4 Summary

In this section, we covered information about the current approaches taken to tackle the problem of phishing attacks. This chapter discussed how automated phishing detection approaches can be limited by their inability to adapt to new URL patterns. A theoretical phishing tool could improve on these approaches by presenting information to users at key points to allow them to make informed decision on their own safety. One way this was suggested, was the incorporation of embedded training into such a tool, to allow users to retain their learning over a longer period of time. By providing information in the form of automated security indicators at key points invention points - the hope is this theoretical tool could help to reduce malicious phishing.

Chapter 4

Requirements Gathering

My goal is to develop a tool that informs users about and protects them from malicious URLs, through web request interception. I anticipate this tool would work well in a browser or application plugin. Designing such a tool involves the incorporation of numerous sources of information: an understanding of Usable security, existing approaches of tackling phishing, and an understanding of secure programming methods.

This chapter outlines how the requirements for developing an explanatory anti-phishing tool were gathered, alongside the design decisions for doing so.

4.1 Initial Concept Design

The issue that I am attempting to address is the fact that users do not know much about malicious URLs, and the large amount of phishing attacks that can occur as a result. The lack of effective tools to inform users about why a URL is malicious and prevent users from being effected by them exacerbates this problem. Thus the research question to which I aim to provide an answer to is:

How might we develop a phishing learning and detection tool that will protect from, and inform users about, malicious URLs?

The initial concept of this project was to develop on the work of the Faheem slackbot [5] by implementing a system which would be able to present the user with explanatory URL information to help better inform their decisions around URLs. This was also influenced by the work done by Volkamer et al.[72] to provide feedback to users in the form of a mouse-over for each URL in an email. Their approach was designed to help users make informed decisions before they click on a link. The idea was to build on this research and develop a more comprehensive system on a platform which would have a wider reach.

One of the key considerations was the ability to incorporate active intervention to prevent the user being directed towards a malicious site. This was shown as a suggested approach in the work of Yang et al. [86], which used a chrome extension to warn

users of malicious phishing. Their field experiment suggested an active intervention approach could be incorporated into a future anti-phishing tool, as long as the user was presented with an understanding of why they had been intervened. This complements the inclusion of embedded training in such a tool by displaying this malicious URL reasoning to the user at appropriate points of intervention. To implement such a system, a web browser was considered as the best vehicle, since this is the primary way of viewing graphical web pages. These web pages are linked to by other applications who wish to display or access web pages for example [68]. This would mean that any web request arising from the clicking of a URL from an application outside the user's browser should be able to be filtered by this system.

The work done in Section 3.1 to design an idealised URL display, also presented an opportunity to be incorporated into a potential tool. As this pre-existing work has already been evaluated with users, the incorporation of this display would have the benefit of limiting the amount of design required to develop the tool.

As browser extensions are the primary means of extending the functionality of web browsers, these were selected as the means to implement this system. This led to the concept of an extension which would provide user feedback about each page's current URL in the omnibox (the box at the top of a browser which contains the current page's URL). Since it was pointed out that this approach would only be helpful for users who have landed on a phishing site already, discouraging them from entering their details perhaps, but not being able to tackle the issues that might occur with phishing sites such as drive-by-downloads. This necessitated an expansion of the scope of the design to incorporate all of the links on a page, before the user had clicked on them.

4.1.1 Browser Choice

The Chrome browser was chosen as the primary deployment vehicle of the tool because it has largest market share of any currently used browser (62.5% globally). [73] This means the tool as a greater reach, with an ability to benefit more users.

Developing for this browser is also well documented, and it includes clear official documentation and access [24]. The browser also had a wide range of inbuilt framework API access, and importantly, access to a 'webRequest' API [26] - which would allow the processing of a users URL requests before they have visited the site. This easy access to useful APIs was similarly the case for the Mozilla Firefox browser [43], when comparing them both to other browsers, but this has a much smaller market at (6.3%). Therefore this presented the Chrome browser as the ideal development platform.

4.1.2 Identifying the Userbase

Identifying the ideal user base was a core consideration of the development of this tool. As discussed in chapter 2, phishing is a ubiquitous problem which affects users of all technical backgrounds. Whilst deceptive phishing attacks are more easily detected by users of above average technical skill, this user group are as similarly susceptible to

increasingly popular attacks such as spear phishing are. Due to their ability to detect low quality phishing attacks, those with above average technical skill can be over-confident in their knowledge of phishing even without having a purposeful education in the subject, such as indicated by the interviews outlined in this chapter. This can subsequently increase these users susceptibility to these attacks [76].

A major consideration of mine in this identification involved considering the body of work I wished to incorporate into the design of the tool. Specifically the work on the URL report by [4], focused its design on above average technical users. This reasons behind this build on the research done to demonstrate how technical knowledge disseminates itself among the wider population. These users, with above average technical, are most likely to be asked about and share their knowledge with users with less adequate technical skills [77, 54, 57]. This means that targeting this user population allows a wider benefit to be shared to the wider population as they all benefit from the education of this group.

This was my primary motivation when choose above average technical users as the main user-base of the tool. Whilst the tool might intuitively be targeted towards those with no experience of phishing, the large amount of technical knowledge required to fully understand the workings of sophisticated attacks limit the ability of the tool to attract these users.

4.2 Interviews

The necessity for the user to get feedback on every URL on the page, provided the basis of my research to find the ideal point of user interaction: what information would be required for the user and how it would be presented to them, and what they would like to see this on.

The main goal of the interviews was to find these interaction points, such that the users would be happy using the tool in their own lives. The interviews also focused on what might be useful to the user given the user interface options available within a chrome extension.

The secondary goal of the interviews was to understand how much knowledge users have on the topic and provide explanations. This was to ensure each participant had a shared basis of knowledge in order to contribute to the discussion of the design.

4.2.1 Method Selection and Reasoning

The purpose of choosing interviews as a research method was to get more focused feedback from a user level. By being able to ask users direct questions on the subject, I could build a depth of understanding as to what user's might mean about a subject. This was very useful at the design stage of the project.

Since the interviews also included the explanation of information as well as the gathering of it, this meant that this was a better choice than a survey - which would have meant that the users would have to read somewhat of an essay to get information. It would also allow me to further question users to explain their previous answers.

4.2.1.1 Participants

Since the idea of the interviews is to better understand how the userbase interact with the tool, the participants included in the study were drawn from the same criteria - people with above average technical skill.

We separated our participants according to computer security, to detect if there were any trends withing participants view of the application relative to their declared experience. Therefore all of our participants had above average technical skills with a mixed range of computer security experience.

4.2.2 Methodology

The interviews themselves having varying levels of structure depending on the topic. The interviews were scripted to ensure information given to participants was the same across interviews. Since the participants were giving some personal data, they were also asked to fill in a consent form. Both the script and consent form can be found in Appendix A.

Since I had little knowledge on how much the participants knew about the Phishing and URL terms, the first portion of the interview was dedicated to explaining the these terms. A structured approach was taken to ensure that the participants were familiar with the terms and context before we reached the User Interaction portion of the interview. The Phishing and URL portion of the interviews was structured to ensure that all of the participants had a consistent basis of knowledge.

To explain these terms, after being asked about their knowledge of the subject in general, the participants were presented with a number of examples of each topic and asked to give an explanation of the presented examples. Afterwards, they were then told what they got right or wrong about each example. This was to help users' give more educated feedback on what they would like to see in the tool. This portion of the interviews was also useful in analysing how participants approach each topic.

The third section of the interviews was the User Interaction portion and this was semi-structured. This was intended to allow for further questions to be asked of the user to explain what they mean. This consists of asking the user how they picture themselves interacting with such a the tool from a top-down perspective: beginning with what they think the priorities of such a tool would be for themselves, down to which combination of user interface elements they would benefit from and how these could be presented.

4.2.2.1 Protocol

An outline of the structure of the interview:

- Start with consent form
- Brief general interview questions to get participant information
- Overview of participant of what specifics are required
- Phishing understanding section
- URL understanding section
- Gained Knowledge test
- User Interaction Section
- Thank the participant and offer informative talking Points

The Background section was used to gather information about the participant, so they can be referenced and be classed as one of the three core user groups.

For both the Phishing and URL topic overviews, the participants were first asked for their initial understanding of the respective term. They were then asked to analyse images containing examples of the topic, and asked to give their opinion on whether it the example was malicious or not. At the end of analysing the examples in topic, the users were presented with the answers to the examples for that section.

The UI Interaction portion was intended to capture the users thoughts on the design of the phishing detection tool itself. The users were asked a serious of questions intended to understand what features they would like in such a anti-phishing tool. Since this was a semi-structured portion, further explanatory questions were asked of the participants to help better explain their ideas on the tool.

For the last question in the interview, participants were asked how likely they would be to use the tool itself, and asked to explain why they choose their answer. This was intended to get an idea of how many users would wish to use the tool, and if there were any aspects that would have to be particularly focused on as part of the development.

The interviews were concluded by offering further information of phishing tips if users wished. This was intended as a means of thanking the participant and giving them something back for their experience.

4.2.2.2 Data

The final amount of participants in the study was 17, but the results of only 16 participants were analysed as part of these interviews since one participant (**P12**) later decided to reevaluate their estimation of their own technical skills. This lead to them being outside the scope of the target participant group of the study, which meant their results could not be included in the final analysis of the data.

Each interview lasted around one hour, depending on the individual participant's enthusiasm for discussing the tool's potential UI. The information from the interviews was transcribed by myself, and an audio recording of the interviews was made to be used as a further reference for information that the transcript had missed.

4.2.2.3 Analysis

Since many of the initial training section had clear finite answers, this allowed this part of the process to have quantitative data, which could be evaluated. The qualitative results of this approach were analysed by using basic statistical analysis.

From all of the parts, there was a lot of quantitative data which had to be analysed. The decision was made to analyse this though open coding. This is the process of finding meaning in qualitative data by labelling sections of the data with codes. These codes are labels which summarise chunks of data - just based on the meaning that emerges from the data. This occurred for each section of the interviews, which allowed us to pick out key themes and trends for each section. After this process was completed, the relationships between these open codes were identified to form themes which underline trends in the data - a thematic analysis.

Due to the amount of transcripts to process, this analysis was completed with the help of a friend, who helped with data entry and the final thematic analysis.

4.2.3 Results

The full results of the quantitative analysis and open coding can be seen in Appendix B. The results of the study are summarised and discussed in the subsequent sections for each topic covered in the interviews.

4.2.3.1 Phishing Understanding

Participants were largely able to detect the phishing emails and point out relevant indicators, with the average success rate for phishing identification across users being 97.91%. Most participants focused on the quality of the email itself and many pointed out the email address as a focus of their concerns. This is in line with Phishing recommendations, and participants that had no self-declared knowledge on these subject were still able to point to these characteristics as a general aspects of the interview. This may be due to the educational structure of the interview themselves - in that they were provided with the definition before proceeding with the tasks.

What may require investigation in future work was participants focus on the current relationship with and knowledge of the sender for the emails as an important aspect of understanding and how this relates to their susceptibility to spear phishing attacks. This was outside the scope of these interviews since the phishing section was focused on deceptive phishing.

4.2.3.2 URL Understanding

Participants on the whole largely knew the purpose of URLs, as demonstrated in B.5, but had less of an ability to adequately read them.

The participants particularly struggled with detecting more difficult aspects of URL reading, such detecting non-ASCII characters hidden within URLs. A significant number of participants were unable to understand shortened links well either. This suggests participants need these aspects to be highlighted for them in particular.

The participants confusion around the buzzword questions for URL reading also suggests some participants do not understand URLs well enough to detect the malicious aspects of them. This is completed with many participants basing their answers on the knowledge of the source, around 50%. Participants also had a number of misconceptions of URLs at times which were interesting, but may be related to the varied cultural background, in line with the universities statistics, the participant pool was drawn from.

Different strategies were employed by the participants depending on the URL. This may be due to the nature of the questions themselves, with figure A.4 being an example participants would be unlikely to encounter normally.

4.2.3.3 User Interaction

Users generally wanted the tool to work before they visit a site, in that it works when they click on a link, with many of them wanting it to focus on malicious sites. They had a strong focus in incorporating active intervention into the tool.

They largely wanted this information to be presented with a dedicated intervention page with some participants specifically requesting that this be unmissable (**P11** - “it should make a big noise so I don’t miss it”). Participants also strongly suggested a traffic light reputation system for highlighting the status of the URLs. They were also clear that the intervention page should give the details of why their routine had been intervened, and some participants felt that these details should be clear and accessible at any time.

When asked how participants would like to be presented with this information in the form of a Chrome extension, participants had a variety of ideas of how and what should be presented. A majority of participants, however, suggested the tool should incorporate link annotation shields to highlight the status of URLs, with a popup including an information breakdown, which could be used to explain how this status had been derived. They also requested a context menu entry to show a breakdown of the link details for any link they may wish to analyse. The participants also had ideas for additional features: with one participant focusing on the ability to report URLs as part of both a popup and a context menu entry. Some participants focused on the tool’s ability to provide high level information about the site in general: through either an overall information breakdown or a badge with a count of all the links in a page. Relative to

Chrome extension UI Elements		
Themes	Codes	Participants
Link Annotation	link annotation shields	P8 P6 P5 P13 P4 P14 P1 P15 P10
	details on shields click	P8 P1
Popup	popup with information breakdown	P8 P7 P2 P16 P14 P1 P15 P10
	popup with overall site details	P9 P13 P4
	ability to report in a popup	P17 P1
Icon and badge	icon behind omnibox	P14
	badge that shows number of malicious links	P7 P9 P16
Other Pages	learning about URLs page	P7
	tutorial page	P7 P1
Context Menu	context menu for link details	P11 P13 P3 P2 P4 P16 P14
	context menu for reporting links	P17

Figure 4.1: Thematic Analysis for Chrome extension User Interface elements

other UI elements, participants generally had no strong opinion on use of the badge and the icon.

As for help information, participants thought it would be useful to have a guide displayed when they first installed the tool and also a tutorial page on how to use the tool. Some made the further suggestion of having a learning page on URLs for further details on this topic too. They largely wanted this information to be presented as "only pictures and text" with clarity and time both being recurring reasons for this. A small number of participants felt they would not need such help features at all as they would intuitively know how the tool works when it was first installed.

As a recurring number of participants wanted the tool to have minimal obtrusiveness in their normal routines, as seen in the table B.7. This request for minimal obtrusiveness by participants was closely associated with participants request to have a low false positive rate and high accuracy with the tool. The interview participants diverged into three main groups: those that wanted a lot of intervention, those that wanted minimal intervention only when necessary and those that enjoyed the mix of both but had other reservations such as not wanting to see the link annotations all of the time.

4.2.4 Likelihood to use the tool

Participants on the whole were likely to use the tool, with around 50% describing themselves as being *Fairly Likely* to use the tool. Some participants really enjoying the idea of the tool, with multiple participants signalling out the many different aspects of the tool that they would like to use. A vocal minority of participants however felt that they had enough knowledge in the topic such that they would not need to use the tool at all. Instead, they felt that the tool was more suitable for non-technical people, even though this ran contrary to the overall URL accuracy results for harder to detect URL features.

Looking at figure 4.2, we can there is a trend among the participants based on computer security experience. We can see that participants who have self-declare their

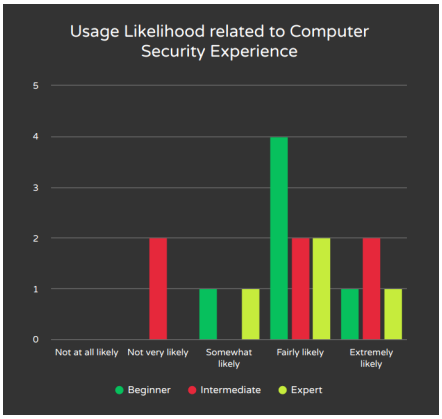

Figure 4.2: Usage Likelihood related to Computer Security experience

experience as **Intermediate** have a lower weighted average towards *Somewhat likely*, than the other **Beginner** and **Expert** groups, which have a higher weighted average of *Fairly Likely*. Analysing the open codes for this question, this may be due to overconfidence on the part of these participants, in that they feel they know enough not to need features of the tool.

4.2.4.1 General Analysis

From the participants' understanding of phishing and URL reading, this would suggest that participants would benefit from more information in this subject. Particularly, with the potential consequences of not understanding the more detailed aspects of URLs which are harder to pick out. This is a positive indicator that the goal of developing the anti-phishing tool to inform users would be useful for the intended userbase. This is complimented by the positive overall response when users were asked how likely they were to use the tool with 68.75% of users, feeling they were *Fairly likely* to use the tool. The participants also singled out many of the features that are intended to be incorporated into the tool as being features they would wish to see as part of it.

4.2.5 Recommendation

Based on the previously outlined results, the User Interface elements were chosen by selecting the most popular features and combining these into a compatible cohesive combination. Since the intervention group was the largest of the identified grouping - it was decided to focus on the wishes of these participants when designing the full capabilities of the tool. This was easy as this group were largely happy with the features that were already being considered for incorporation within the tool. The second group were catered for by adding settings to limit the UI options such as the display of the link annotation shields. The feedback from the last group in particular helped to derive the efficiency requirements of the system.

The UI elements chosen were:

- a context menu entry (on right click) to access link details
- a badge over the plugin icon to display the overall site information
- link annotations with a status for each link on a page
- a popup with an information breakdown on how the tool's status value had been derived
- a tutorial and settings pages with a text and image focus

It was also decided that the system would need a low rate of false positives to help cater for the second identified grouping. Further studies would also be recommended to better understand how to reach out to the overconfident grouping identified in the interviews.

4.3 Requirements

As a result of these interviews, and analysis of the literature, the high level design goals of this system were defined follows:

Requirement 1: Classifying every URL according to one of three states as part of a traffic light system

This classification should occur for those present in any given page, the page URL itself and any URLs user requests from any platform which would be processed by the browser. This would allow the tool to be a comprehensive solution to phishing, and potentially flag other security exploits which employ malicious URLs.

Requirement 2: Using the browser to prevent the user visiting any URLs that have been classified as malicious without explicit user consent

This is intended to cut down unintentional direction to malicious URLs, caused when on platforms such as email and social media.

Requirement 3: Present the information on each URL to the user in an understandable way, both at appropriate points of intervention and on user request

This is to encourage the user to learn through embedded training, and allow users to catch URLs themselves through this training and contribute to reducing their own danger.

Requirement 4: That the end built system includes the best practice of software engineering: maximising efficiency and accuracy

That the system works effectively as and when required, such that users are not do not remove it due to efficiency concerns.

4.4 Summary

This chapter outlined the design process which occurred in choosing to develop an anti-phishing tool for the Chrome web browser. As a result of the interviews and a prior literature review, the features of the tool were able to be selected. The proposed system should provide contextual information on every link a user visits on every page. It should also incorporate active intervention, to prevent users from visiting any page that it deems malicious, and present them with why this is the case.

The UI features of the tool, as they will be incorporated in the Chrome extension, are also derived as part of the interviews. This provides a solid base on which to establish a more developed system design.

Chapter 5

Design

In order to build on the requirements I had gathered from users for the anti-phishing tool, in this chapter, I outlay the design of the tool in order to evaluate how effective the design will be at achieving the desired requirements. I begin this process by prototyping the user interface with paper drawings before developing a proposal for the technical implementation of the URL calculation system. I evaluate these designs by consulting with an expert to give further feedback and improve on the design.

5.1 Prototyping the User Interface

I initially began by prototyping the user interface by drawing out my design on paper. The goal of this was to test the design of the tool prior to implementing it, as a basis to get feedback on this.

The main layout of these pages was primarily drawn from figure 3.1, as outlined in paper [4]. Since this, the purpose of this paper was to prototype an ideal means of communicating URL status information to users. The communication of this URL status information was iteratively designed with users drawn from the same target userbase. I choose to base the primary view of this tool on this as this presented the advantage of reducing the amount of time I would need to test the way I display URL information to the users. However, one of the difficulties of this decision was that the URL report design had not been technically implemented due to the complexity of implementing this as a functional user interface. In addition, the URL information display did not fit with the look of a typical chrome extension, being originally designed as a report rather than an interface. This meant that it had to be adapted for usage as part of a functional UI.

Figure 5.1 illustrates an example of one change which had to be made to the URL report to use it as part of a functional UI. This is demonstrated with the addition of a further details expansion panel, rather than having this information being constantly displayed as part of the URL report. During the interviews, the users highlighted that they would not want to see all the URL details at once, so this design decision had the

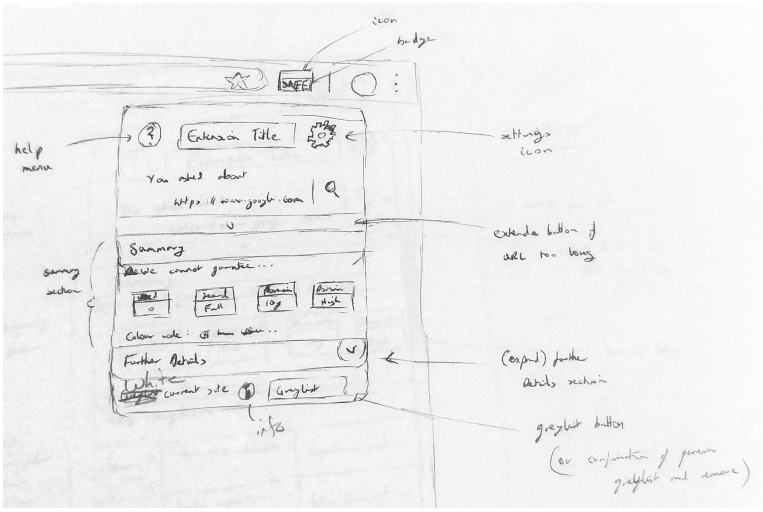

Figure 5.1: Popup Summary

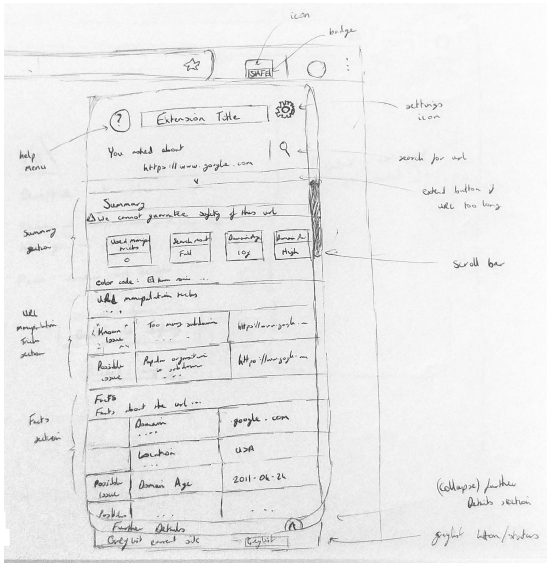

Figure 5.2: Popup Summary with Further Details

benefit of meeting their request. Figure 5.2 shows a paper design of the tool with the further details panel displayed (after having a user click on further details).

The user intervention screen was designed by drawing inspiration from Google Chrome's site warning feature, as can be seen in figure 3.3. This demonstrates the incorporation of embedded training within the extension through the display of the URL analysis at the time of this intervention page being shown.

Further paper designs of the tool's features are provided in Appendix C. The paper designs for these tool features were partially inspired by other high-quality user interfaces of existing Chrome extensions, such as Ghostery [21].

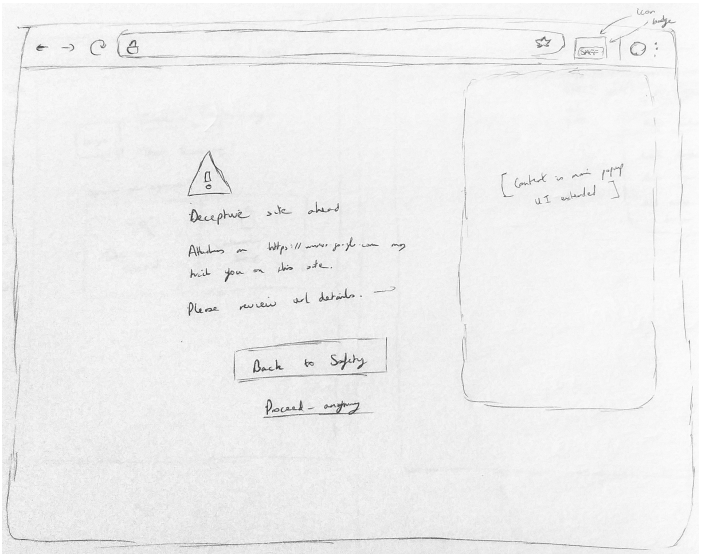

Figure 5.3: Intervention page

5.2 Back-end Design Proposal

The Back-end Design Proposal primarily involved outlining how the tool would classify URLs in order to be displayed by the extension's front-end. There were a number of competing issues in consider in the design of this algorithm. The report itself, comprising a full list of the included heuristics, can be found in Appendix D.

5.2.1 Selecting the Algorithm Type

First off with limited resources, such as a lack of labelled data, it would have been needlessly challenging to develop an algorithm which employed machine learning. In particular, it would have been difficult to develop an algorithm which could have beaten the accuracy rates of Google's own, as referenced in Chapter 3. This was coupled with the knowledge that the tool would be able to benefit from the results of Google's algorithm, with the ability to easily access Google's safe browsing data [30].

Secondly, there was a need to process the URLs in such a way that would provide information for the front-end to display. In machine learning approaches, it is possible to produce a label and associated probability for the data but much more complicated to derive why the URL would have been labelled in this way. This was compounded by my lack of in-depth knowledge of machine learning at this time.

Therefore, this lent itself to employing a heuristic algorithm. This type of algorithm could be used to match the URL against a set of defined heuristics. This would result in a status value for each heuristic which could be then be aggregated with the values of the other heuristics to form an overall count of the status types. The overall count of these types can then be matched against a set of given thresholds. The issue with this approach is the difficultly in selecting appropriate heuristics which cause the algorithm to give a high accuracy. This is difficult for phishing in particular as it requires a

significant amount of research in what is a literature dense research field. The benefit of this approach is that the heuristics would be able to be used in the User Interface calculation of the tool itself. For example, if a heuristic is present in a URL, this could be clearly referenced in the front-end, such as in figure 5.2.

This is why I chose to use a heuristic algorithm. The core of this algorithm, beyond what has been described, was the association of each heuristic with a severity level. These severity (or issue) levels being, in order of decreasing urgency, one of the following: “known”, “possible”, or “none”. The algorithm was designed to then use the severity level of these heuristics to indicate an overall status value of for a URL: “safe”, “warn” or “alert”.

5.2.2 Decision-Making Algorithm

Each of the severity levels is associated with a colour: “safe” is green, “warn” is yellow and “alert” is red. This is intended to mirror the traffic light system which is incorporated into the UI.

The decision-making algorithm’s input will be the given URL and all required information needed to consider the classification of this URL. The algorithm begins by considering each case where a URL might be considered purely green (i.e. considerably safe). If any of these indicators, such as the inclusion of the URL in a reputable whitelist, are true then the URL is classified as safe, and the algorithm returns the URL has having a green state. Otherwise, the algorithm will analyse the URL and classify the given URL as having either a yellow or red state.

To achieve this, the algorithm analyses each set of features using the heuristics outlined in the subsequent chapters. Each set of features is a useful indicator of a URL’s safety, which can be used to classify how likely a given URL might be phishing. These indicators are referred to as issues in the Back-end Proposal, and each of these issues is categorised with one of the three severity levels.

To classify a URL, a count is kept of each issue that arises from the data parsing. This is matched with threshold values which are used to classify this URL. A list of these classifications can be seen in table 5.1.

Known Issues	Possible Issues	Personal Whitelist	Output
¿=1	¿=0	False	Red
¿=1	¿=0	True	Warn
0	¿=5	False	Red
0	¿=5	True	Warn
0	¡5	False	Warn
0	0	False	Green

Table 5.1: Thresholds for Output

Since the algorithm favours red classifications rather than yellow, to provide a stronger safety net for the user, this potentially creates a higher false positive rate for the user

Trick	Explanation	Check	Data Needed	Issue
Too many subdomains	Can redirect user to alternative site	regex search all '.' characters and take a count of resulting array	Amount of common URL domains; The URL domain	Possible

Figure 5.4: An example manipulation heuristic (drawn from the report in Appendix D)

than what they might wish. Active intervention with a high false positive rate has been shown to have a reduced effectiveness over-time [85].

The primary learning component of the tool was designed to help resolve this problem. This component is the implementation of a personal whitelist - a list for users to store the links they have marked as being safe. The intention is for users to be able to update this whitelist with URLs whenever they are analysed by the tool. This is therefore intended to be a constant element of the UI and will be particularly useful for URLs wrongly classified as being in the red state (false positives). After the user whitelists a site, it will not be given a red state by the system. Instead, it will be given a yellow state which means active intervention will not occur for this URL. The user however, will still be made aware they are not entirely safe in the presentation of the same analysis details. This should reduce the false positive rate of the tool for each user over time. How the personal whitelist affects the status classification of the URL can be seen in the Personal Whitelist column of table 5.1. In practice, the ideal outcome would be that the tool tailors itself to users common sites over time, therefore only showing sites that are concerning which the user has never visited before.

5.2.3 Malicious Heuristics

There were a number of heuristics outlined as part of the tool design that will be used to identify whether a URL is malicious or not. Each set of heuristics requires different resources in order to process them. The design proposal outlines information for each heuristic: the heuristic title, a brief explanation, how the heuristic will be calculated, a summary of the data required and an severity level this heuristic should have. An example of this can be seen in figure 5.4.

5.2.3.1 URL Parsing Heuristics

The design of these URL Parsing heuristics focus on parsing the URL into its distinct components, which are then used to pick out elements in the URL itself which might suggest it is malicious.

Processing each of these heuristics involves using natural language tools such as regular language expressions (regex) to match the contents of each URL against any concerning flags. These often involve the request of some data in order to complete each check accurately; the limited amount of required API accesses means these checks can

be performed more quickly and efficiently. Each metric relates to an severity level based on how much of an indicator that metric is.

5.2.3.2 Domain Heuristics

The domain heuristics are designed to measure indicators which focus on the details of domain names. An example of such a heuristic, involves checking a domains registration status. In doing passive queries related to the domain name, the plan is to detect indicators based on the known trends of malicious URLs. To handle each of these indicators, external data must be requested using APIs to reason what the status of a given URL is in related to the Domain Name System.

5.2.3.3 Page Heuristics

The page heuristics naturally involve a high amount of integration with a number of APIs due to the need to gather contextual information about the pages. An example of this is the search results heuristic, which involves the checking for inclusion of the page in a queried search engine's results. Since phishing websites have short lifespans they are not usually in these results. Therefore to implement this, a search engine API needs to be integrated into the tool in order to do this check. Each of the other heuristics also requires the use of a number of external data sources to be able to accurately function.

5.2.4 Safety Thresholds

As part of this algorithm design, the tool is intended to match heuristics based on the safety of the URL before the analysis of any other heuristic. This was a design choice intended to reduce the computation time required for each URL. If the URL can be established as being safe, there is little need for users to be presented with further contextual information on why it is wrong.

The issue with this approach is selecting the appropriate features which can be used to accurately indicate a URL is safe. This is very challenging as selecting the wrong features could lead to false positives in which a malicious URL is inaccurately marked as safe.

The proposed heuristics are a combination of a site having a high page rank, a high popularity (due to its inclusion on a list such as Alexa's top sites) and a high amount of social media shares. The first two are primary features, with the social media shares being a secondary feature used to decide edge cases (as this is less reliable). The thresholds for these features will be the subject of further research.

5.2.5 Data Sources

As part of the algorithm proposal, the potential data resources that could be used to populate the information of the tool are outlined. This is not an entirely complete list of resources and is intended to present a picture of critical areas where information would be sourced. For instance, a list of the most popular sites to be used in safety metrics sourced from Alexa Topsites.

5.3 Expert Design Evaluation

While designing the tool, I approached an expert in phishing and URLs, Kholoud Althobaiti, to evaluate the design proposal for the tool. Expert evaluations involve using the experience of experts in a field to analyse potential problems in proposed designs. As an evaluation method, it is useful for evaluating difficult material and can help identify any user interface or technical issues early in a design process - before more costly implementation or user studies [71].

The expert evaluation was focused on two main aspects: the user interface and the back-end of the URL decision-making components. The goal was to help identify any potential issues in my design to allow me to improve these before implementation.

5.3.1 User Interface

As part of this evaluation, I met with Kholoud in person to outline my initial plans for the tool. Kholoud gave me her feedback on the User Interface and the general approach of the tool.

I presented her with the paper prototypes of the tool I had made before, as in Section 5.1. Kholoud suggested that the User Interface was generally suitable for the outlined purposes. Initially, she highlighted that the details of the URLs should not be displayed in the expanded further details panel, but borrow from tools such as Netcraft and open in a new web page to highlight this content more clearly for users. However, after we discussed that this might make users less likely to access this information, Kholoud suggested the current approach was sufficient. She further advised that the summary section 5.1 be simplified to make it clearer to users what the status of the URL was. Kholoud felt the badge alone was not sufficient to display the status of the URL and that this information needed to be more clear. The Netcraft interface (in figure 3.2 on 24) was drawn on as an example of how I might display this status information - by referencing the status bar included at the top of the Netcraft interface. These changes to the summary were very useful, and not some that I had considered including as part of the tool.

To help with this, it was suggested I reduce the size of the extension title, so there was more space for the summary. Kholoud was not sure that the Google search feature was necessary as part of the tool, as this information would be displayed as part of one of

the heuristics already. She also suggested some changes to the buttons and the general layout to make them more clear for a user.

As an additional feature for the tool, Kholoud recommended linking the tool with an blacklist such as PhishTank [53] which allow the user to report malicious URLs they encounter to this blacklist. This is intended to benefit the whole community since the tool has been designed to use blacklists such as PhishTank in its calculations.

5.3.2 URL Decision making

Alongside, the feedback on the User Interface I also discussed the design for the URL decision-making algorithm. I outlined my plans at the meeting to design a heuristic algorithm based on the indicators that have been established as common across URLs. The intention behind this is to both analyse the URLs and build a list of the issues that are displayed to users.

Kholoud was generally positive about this approach and pointed me to further resources which would allow me to increase the breadth of heuristics involved in the tool. Subsequent to this discussion, I finished preparing the 18-page Back-end Design proposal discussed before. Kholoud agreed to review this proposal and the severity ranking I allocated to each heuristic as part of our expert evaluation.

5.3.3 Evaluation of Design Proposal

After reviewing this report, Kholoud provided extensive feedback to improve the phishing heuristic algorithm. The feedback both discussed the overall algorithm and gave specific feedback on some the heuristics.

One of the major improvements Kholoud was originally confused as to why I had chosen the arbitrary threshold of five when classifying a URL as yellow or red, see table 5.1 on page 44. This was chosen at the time based on my intuition that this number would represent a satisfactory mid-point for users false positive rates. Kholoud advised this was not enough to justify this threshold, and this could be improved by varying this threshold in a future user study to find the ideal threshold to minimise false positives.

Another major improvement that Kholoud suggested was having two metrics for each heuristic: a metric for a known issue and also a possible issue. For example, with the URL manipulation heuristic - the number of subdomains - this heuristic could be a possible issue if it has less than three subdomains, but more subdomains would indicate a known issue. This was suggested to improve the depth of the heuristics and improve thefore improve the accuracy of the tool.

Kholoud also gave feedback on the individual heuristics in the report too, suggesting improvements for the issue level and calculation means. For instance, such as dropping the HTTP/HTTPS heuristic as one the URL manipulations due to its unreliable nature as a phishing indicator.

5.4 Final Design

Overall, Kholoud's feedback was invaluable in helping to analyse design of the tool. Due to the depth of work involved in implementing each heuristic and particularly the integration of all the necessary APIs, this presented a clear opportunity to distinguish this section of the project for work next year. Due to the feedback on the back-end design, further work and research is needed to more accurately determine the improve the accuracy of heuristics designed for this tool, with reference to the appropriate literature. This future work is further discussed and justified in chapter 8.

As a result of this feedback a number of changes were made to the design at this point in the project:

- The presentation of the URL status was improved in the summary section of tool's main view
- The ability for users to report URLs was added to the interface design
- The summary information was ensured to be simple and clear when presented to the users in the final implementation

5.5 Summary

This chapter outlines and evaluates the initial designs of the tool, drawn from the requirements outlined in the previous chapter. As a result of the expert evaluation discussed in this chapter, the project plan was formalised by the decision to make the URL decision-making components be the focus of next year's work. As well as this, a number of improvements were made to the final system design such as improving the clarity of the tool's status summary and the depth of the designed heuristics outlined in the Back-end Design Proposal.

Chapter 6

Implementation

The CatchPhish system was implemented to leverage existing web technologies to derive information about any given URL in any web page, or entered into the browser. The system achieves this by extracting all the URLs from each page the user arrives on. The tool then sends each URL to a purpose-built decision-making server which parses and handles the analysis of the URL. This information is then presented to the user using a combination of URL presentation research[4], and interaction points, backed by the requirements gathered in Chapter 4. The resulting system provides extensive information about each URL in a user-friendly way.

This chapter discusses how this system was implemented and the design decisions that were taken to achieve this.

6.1 Project Timeline

As part of the final implementation of this project, there were 4859 lines of code (discounting libraries, framework modules or any other code not written by myself) for the final implementation of the system across three distinct development frameworks. I chose cloc[2], as an Ubuntu [10] code counting tool (which discounts comments, blank lines and non-text files) to count the lines of code. This is because GitHub [23], which was used as the remote version control provider, gives a more generous but misleading amount, see figure 6.1.

However, figure 6.1 also demonstrates the project timeline well. I began the project doing background reading as part of a literature review, whilst to help formulate the project goals. After initially prototyping some back-end features, such as incorporation of the URL-parse [50] module. The next stage of the project was to plan out the interviews I would use to gather requirements from the intended user base. I then prototyped the tool over the Christmas break, specifically working on the extension infrastructure. Then spent two subsequent weeks conducting the interviews. These were completed, before spending the rest of the semester implementing the front-end and the decision-making elements of the system.

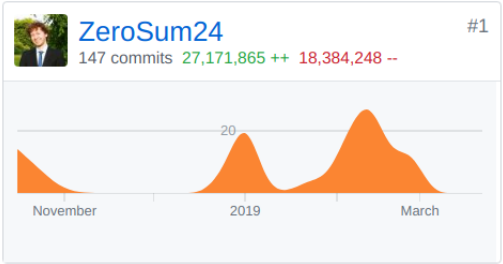

Figure 6.1: Lines of code and commit timeline in the project according to Github. The lines of code include a count of any changed lines of code including the development build libraries.

6.2 System Overview

6.2.1 Components and Interaction

There are three major components across the system:

- **Chrome Extension Infrastructure** - handles the overall application logic, the extension's interactive elements and page alterations
- **Front-end** - responsible for the main user URL breakdown display, along with tutorial pages, built using a React app incorporating Material Design
- **Server** - responsible for the URL calculations, built using the node.js framework

Figure 6.2 demonstrates how each of these components fit into the overall system architecture. The figure shows what information they communicate with one another, and the major sub-components in each component.

6.2.2 Choice of Technologies

6.2.2.1 Extension Technologies

The primary technologies used to develop a Chrome extension are also the core web technologies - HTML, Javascript and CSS [34].

- **HTML** provides the basic structure of sites, which is enhanced and modified by other technologies like CSS and JavaScript.
- **CSS** is used to control presentation, formatting, and layout.
- **JavaScript** is used to control the behaviour of different elements.

Since these are used throughout the web, there is extensive documentation in developing with these technologies [75].

The use of these technologies was an implicit requirement for this project since they are a necessary part of developing a Chrome extension. As I had limited prior experience

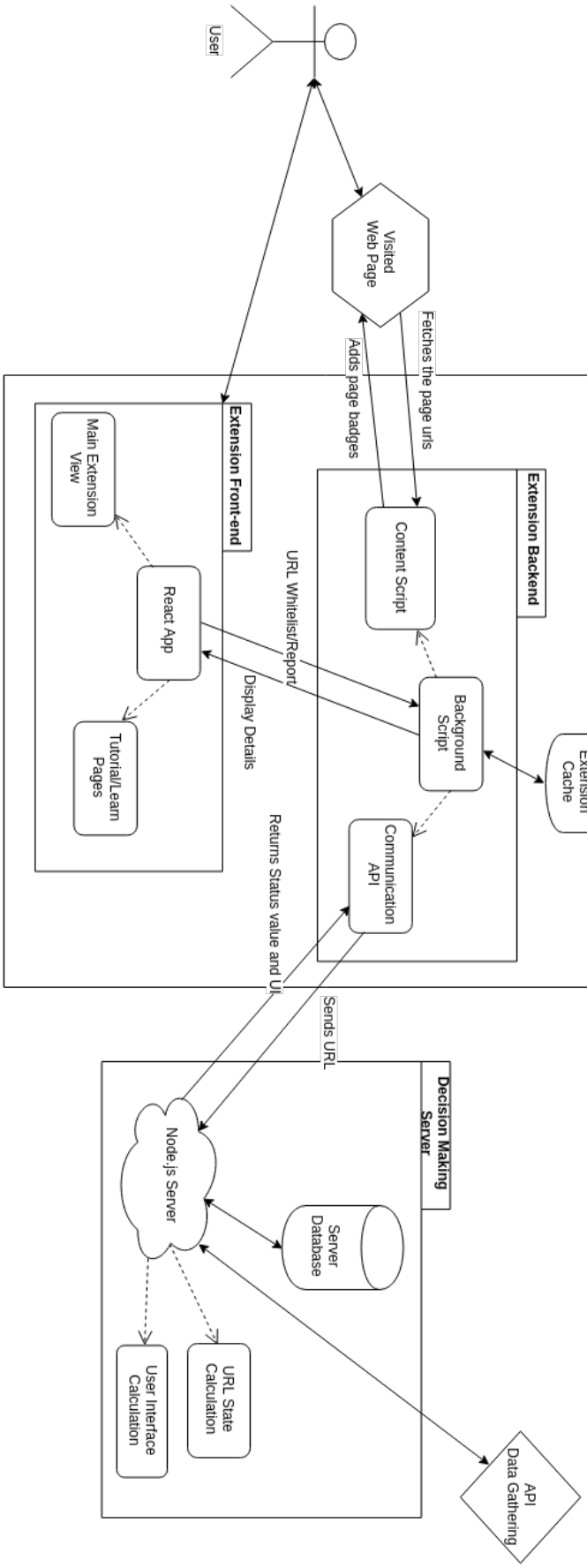

Figure 6.2: System Component Overview

in developing with these technologies, the project presented an opportunity to develop these skills.

6.2.2.2 Front-end Technologies

Choosing the Component Library:

Initially, I attempted to implement the extension front-end using HTML. However, this proved difficult with the limited amount of pre-built resources present within HTML: requiring each layout button and style to be created from scratch. The time it would take to develop using this technology alone would have been too long, particularly to develop a clear and consistent user interface.

As a result of the slowness of this approach, I began looking into other libraries and frameworks that I could utilise to improve the speed of my development and the resulting quality. I compared AngularJS [6] and React [56] as potential option: being two popular web development libraries available that can be integrated into chrome extensions [22, 63].

In exploring AngularJS, I found it to have too many unnecessary features. There was little need for a full Model View Controller (MVC) model, this is an architectural pattern commonly used for developing user interfaces that divides an application into three interconnected parts [58]. The full model was unnecessary in this instance since I was able to handle the Model and Controller elements through the extension backend. Due to the larger amount of features, it was also more difficult to learn and touched on my time constraints for the project. Since React only handles the View element of the MVC model, this was better suited for this project.

React was thus the favoured option and I began developing the front-end using this technology.

Choosing the User Interface Design Technologies:

Material design [17] was something I wanted to incorporate into the project due to the high-quality appearance of the assets and consistency with the chrome design. This is a design language created by Google and incorporated into their products which focuses on using responsive animation, padding, and depth effects to make the interactions clear to the users. The main purpose of which is to incorporate design principles with a modern technical user interface implementation details.

Contrasted with styling paradigms such as Bootstrap [9], which has a similar collection of assets, Material Design was favoured due to its consistency the design with Chrome which is intended to help with the overall user experience of using the tool. Due to React's ease of incorporating components from other component libraries, I was able to use bootstrap components in at least one part of the application where the Material Design library was not suitable.

Material design integrates well with React, and between them they have a large amount of documentation available to help with development. This allowed me to create a

visually striking and reactive implementation.

6.2.2.3 Server technologies

As part of implementing the tool, the original intention was to incorporate the URL decision making as part of the extension infrastructure itself. However, due to the implementation of Chrome extensions I found there would be difficulties in incorporating the high amount of APIs required. This is due to the need to explicitly declare each one of these to the user and the Chrome extension's closed environment policy which is designed for user security. Since these APIs are needed to process many of the decision-making heuristics, it became necessary to use some external tool to handle the decision-making. This led to the decision to implement a server as a central resource to handle the decision making for all deployed extensions. This design decision had the further benefit of reducing the number of computations needed on each user's individual device for each URL analysis, in favour of stored central calculations.

I created the decision-making server using *node.js* [49], an open-source cross-platform run-time environment that executes Javascript code outside of the browser. The HTTP module [74] as part of this environment is what I used to run the server. This was chosen due to its simplicity and ease-of-use.

The chosen server technology allowed me to prototype the server quickly, due to my experience with this technology when testing the URL parsing components earlier in the project. The use of this technology had the further benefit of being lightweight compared to more dedicated server technologies such as *apache* [20], which would have required significantly more time to learn to incorporate into this project.

6.2.3 Understanding Chrome Extensions

6.2.3.1 What is a browser extension?

A browser extension works by extending the functionality of a browser by adding further components. These are typically designed to incorporate further features such as to improve the ease of use of a user or provide features such as improving the user's security.

6.2.3.2 Chrome Extensions

Chrome extensions have access to a number of specified APIs to allow extensions to interact with the browser in a controlled environment. This environment acts as a limited framework for developing extensions. At a high-level involves having a manifest file to specify the details of the extension including its name and the icons to be used as part of the extension. Coupled with this are separate scripts, each with different levels of access:

- **Content script** - access to the web page but with very strict access to a limited set of APIs
- **Background script** - full access to extension APIs but no direct access to the web page
- **Popup script** - access to the popup.html file (the main display of the extension) but limited access to APIs

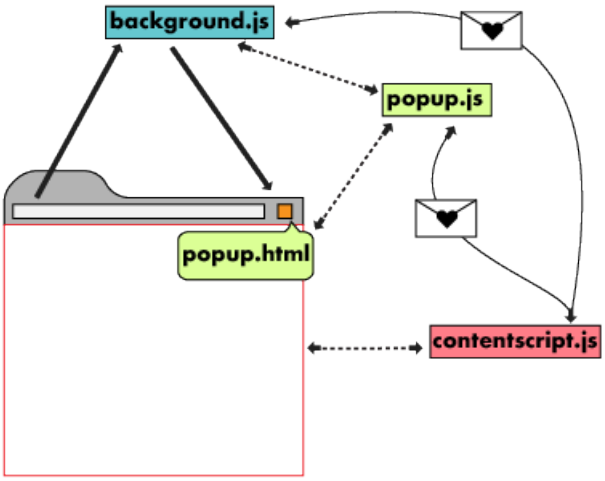

Figure 6.3: Extension component overview[25]

Each of the different scripts are isolated environments able to communicate with one another using the chrome extension message passing API, as illustrated by figure 6.3. This allows data to be passed from each of the extension elements to another.

As part of developing a Chrome extension, developers are provided access to a number of API as part of the development environment. This provides a way for developers to access features of the Chrome browser they would otherwise not. These APIs also allow access to the visual components that Chrome extensions have access to.

6.3 Extension Infrastructure

The extension infrastructure was developed by using each of the components outlined in the Chrome extension.

6.3.1 Content Script

The content script works on every page that the user visits and extracts and updates all the links on those pages. To achieve this, it incorporates the jQuery [36] library, a lightweight Javascript library which simplifies a number of common tasks done in Javascript, allowing these tasks to be written with a single line of code. This is why I

choose this library when compared to using plain Javascript, as it greatly improves the code readability and reduces the code complexity.

For each page, the jQuery library is used to find all of the anchor tags on a page. "An anchor tag is an HTML tag used to define the beginning and end of a hypertext link"[81]. Users click on these tags to reach the destination of the link. Each anchor tag contains an attribute called a *href* which contains the destination URL for that anchor tag. JQuery finds all the anchor tags in one line of code:

```
anchor_tags = $('a');
```

From each anchor tag with a href, the content script extracts these and sends each of these to the background script for classification. Whilst these are being processed, the content script updates all the relevant links in the page with a small grey shield annotation. This was a design decision intended to show the user that the tool is still working at this point. This is also done using jQuery, by finding all the anchor tags in the page and updating the ones where the href is being processed.

Once a link is processed and sent back by the background script, using the message API, the extension annotates the link with an appropriate coloured shield of the colour representing its status. This means that the links in the page are updated as they are processed since the message API is an asynchronous process. This helps reduce the page load for the user, rather than waiting for all the links on the page to be classified before they can view the page. Response time has to be considered from a user perspective as it influences the design of the tool. Response times slower than ten seconds influence how a user interacts with the page [47].

Further justification of link annotation feature from a user perspective can be found in Section 6.4.2.3.

6.3.2 Background Script

The background script handles the overall application control. It interacts with the extension cache, initialises the extension UI elements, handles the application message passing to the front-end and processes the server classifications. It also filters the web traffic of the user.

Each of these extension UI tasks is done with integration with and manipulation of Chrome extension APIs such as the *browserAction* API.

6.3.2.1 Extension data storage

The extension handles different data types individually, using a combination of different Chrome APIs.

To implement the users' personal URL lists, it stores each URL to a synchronised storage state which is connected with the users Google account. This allows users to have the same personal URL lists maintained across each usage of their browser.

The extension stores each URL that the user lists in local storage in the form of a cache. This is intended to reduce the computation time for each URL since it can consult the cache if the URL has previously been computed rather than consulting the server. This uses Google Chrome's local storage API.

These storage APIs are both asynchronous so do not affect the efficiency of the system. These APIs are also very lightweight, so they are called whenever there is a new value to be updated since there is no noticeable systems impact in doing so: the Chrome browser handles this with a purpose-built file queue system.

6.3.2.2 Web request intervention

As part of the tool, the extension interrupts any web requests the user makes that are deemed malicious. Each of these is considered by the server and stored in the local data. A web request is blocked based on the flags associated with the URL. Any request which is an alert is blocked. The request is analysed *onBeforeRequest*, so no content from the page is loaded before it is blocked.

When a page is flagged, rather than go to the malicious site, a site warning page is shown with the option for progressing to the malicious or returning to their previous site. The extension interface is opened at this point, and the user is encouraged by the text on the site to read the information on the extension.

To do this, the web request API that is provided as part of Google's APIs are used. This is a very useful API which allows every URL to be checked as it is called on the browser. I have chosen to filter the web request analysis to the "main_frame" web requests. These refer just to those request which are loading full pages. This allows us to filter out the number of web requests to analyse but also allows us to focus on the malicious pages themselves since the contents of the page (non main_frame requests) are not loaded before the main_frame. After which, the system requests the information for the URL and checks if it is an alert status along with whether is has been previously flagged by the user trying to access it. This flag is used to indicate if the user is currently on the site warning screen to allow the user to progress to the site if they wish.

Overall this prevents the user from being affected by any flagged malicious site if they choose not to visit it. It also presents an ideal learning point for them to learn more about the details of the malicious URL in an embedded learning scenario.

6.4 Front-end and User Interaction

The primary goals of implementing the frontend where to implement the designs outlined in Chapter 5, in order to present information to the user in a clear and concise manner.

As part of this, the user interface relies on calculation and display of the URL calculations displayed by the server, with the UI incorporating reference to the state cal-

culation *'safe'*, *'warn'* or *'alert'* outlined in the decision making heuristic calculation in Back-end Design Proposal Section 5.2 to make the status of each URL clear to the user.

The front-end has also been made with features designed to improve user support and understanding when using the tool, this is outlined below.

6.4.1 Research Interface Implementation

The research interface implementation discusses the implementation of the *popup.html* and incorporates the paper designs outlined in Section 5.1. This is the primary means of breaking down and displaying the details of the URL analysis to the user.

6.4.1.1 Developing with React and Material

React as a Javascript View library incorporates a number of small pre-built components which can be slotted together to form larger more complex visual elements. It provides a framework for these components to be integrated together, which is somewhat analogous to slotting together pieces of a jigsaw. To style and position the components, it like HTML works with the CSS technology. React integrates easily with other component libraries such as the Material Design component library, which means developers are able to use these custom components within the React development framework. This allowed me to use these the higher quality Material Design components to make a more consistent reactive user interface.

React incorporates prototype-based programming, which means the parameters for each function are passed top-down to each relevant component of the tool, like a tree structure. To work with this type of programming, I had to design a naming system that would allow the parameters to filter to the correct locations. This required significant planning before the implementation to maintain consistency. The issue with this programming approach is that the tool does not properly render until it has received the appropriate parameters. What this means from a user perspective, is that for a fraction of a second a white box appears rather than the tool frontend itself. This could be improved in future iterations by adding a loading state to the frontend.

6.4.1.2 Viewing the extension popup

When a user clicks on the popup as in the chrome browser, or a link through the context menu on the page, this sends a message to the background script using the message API to get the popup UI details. A response is sent and passed to the user interface details to allow it to be popular across the tool. The heuristics and the content of the summary boxes are calculated server side, as outlined in Section 6.5.0.2, but some of the UI is calculated within the extension itself.

6.4.1.3 Popup Implementation

Figure 5.1 displays the front-end of the tool, the main components of which are:

- **Top navigation bar** - containing the help icons and extension title.
- **URL Status bar** - the coloured bar indicating the status of the URL
- **URL expansion panel**- contains the URL that the user asked about
- **Summary boxes** - presenting an overview of the URL status for different categories
- **Further details section** - an expanded section which contains the details of the URL heuristics
- **Bottom navigation bar** - handles the whitelisting and reporting of the tool

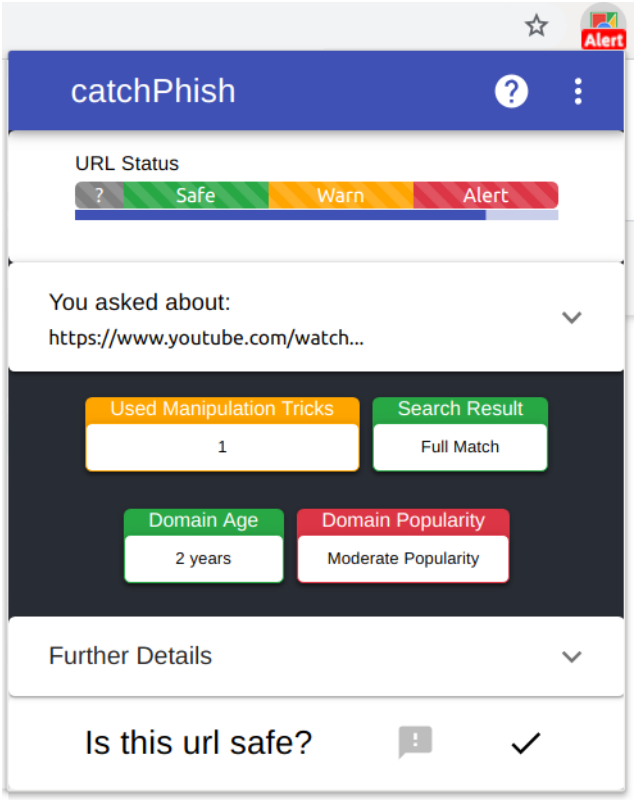

Figure 6.4: Popup Summary Page

The URL status bar was implemented with a combination of Bootstrap and React components. The coloured bar with the status labels was implemented using the bootstrap stacked progress bars object to meet the design goals outlined in Chapter 4. This was because the react material did not have the range of objects to implement this type of detailed content containing progress bar. The blue status bar below this is an example of the inbuilt React Material status bar component. One of the benefits of this design is that it operates as a colour code for the users. The intention is to convey the status

of the current URL and also inform the user about what each of the different colours of the summary boxes means.

The URL expansion panel was implemented due to the size of the URLs. A number of URLs, and particularly phishing ones, are too long to be able to fit into a single line. Therefore this design displays as much of the URL as possible in the single line, with the remainder being ellipsed. This lets the user see which URL they asked about. In order to view the full URL, the user must click on the URL expansion panel, as shown in figure 6.5. This was necessary due to the limited space that the tool would be able to occupy on the screen, and the desire to ensure that users were only presented with the necessary information.

The summary boxes were implemented to give users a high-level overview of the most important heuristics and this was a prominent feature of 3.1. They have been implemented here in different colours to indicate their relative safety value in terms of their contents. This is because many users may not know that a domain age of 2 years is a safe indicator from their limited knowledge of URLs , but allows the tool to indicate it is without them knowing fully why.

The further details expansion panel allows the user to see the list of all the heuristics used in the calculations. Figure 5.2 demonstrates an example of how these are laid out. There are two main components of this: URL manipulation tricks, which details the tricks that have been used, and Facts which details the domain and page details. The URL manipulation tricks are only shown if there are some heuristics present, otherwise these are hidden by the tool so as not to confuse the user with an empty section.

The structure of each of these two sections is largely the same. The section is headlined by a title and followed by a brief description of what the section means. Each heuristic is then displayed underneath in a list. From left-to-right, the heuristic has its issue-level displayed with the appropriate colour of concern and the issue text placed above. This is to help avoid confusion with the status values such as “safe”, but the colours have the same meaning so these remain the same. The heuristic title is then displayed with a brief description of what this is underneath. The last box contains the specific section of the URL, or content derived from the heuristic to be displayed to the user. To implement this, the height of each of these heuristics is dynamically sized in order to fit the heuristic content. This means that the heuristics display is fully adaptable to what is sent by the server.

The bottom navigation bar is the main component which has it’s state stored and calculated by the extension. If the user chooses to report the URL for example, the extension replaces the “Is this URL safe?” message with an option informing the user “This site has been reported”. The status of the URL displayed to the user is also updated, but not the heuristics themselves. Before a user can report a URL they are presented with a small popup over the extension which explains the consequences of doing so and asks them for confirmation. This is to ensure the user is familiar with these terms.

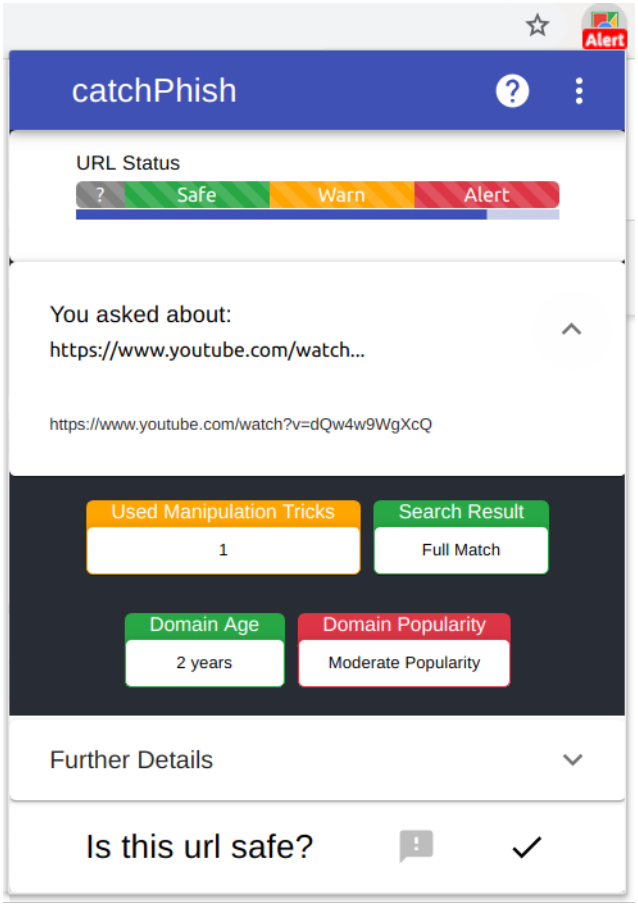

Figure 6.5: Popup URL expansion

6.4.2 Choice of Extension UI Elements and Intervention

To implement the user intervention points that are were derived in the requirements gathering section, I had to incorporate a number of different UI elements as part of the tool. This largely involved integrating with the Chrome extension API. This section presents a summary of the implementation of these components.

6.4.2.1 Context Menu

On chrome, a context menu is created every time a user right-clicks on the chrome browser. The extension framework allows developers to add items to this context menu which can call functions within the extension. This can be customised by designating when the context menu item will be added depending on what type of information the user clicks on.

This feature is used in my extension for when the user right-clicks on links. At this point, a menu item is added which the user can then click on. This opens the extension popup and displays the information to the user on that URL. To do this, the tool captures the URL that the user clicks on and when the request for the popup URL is made

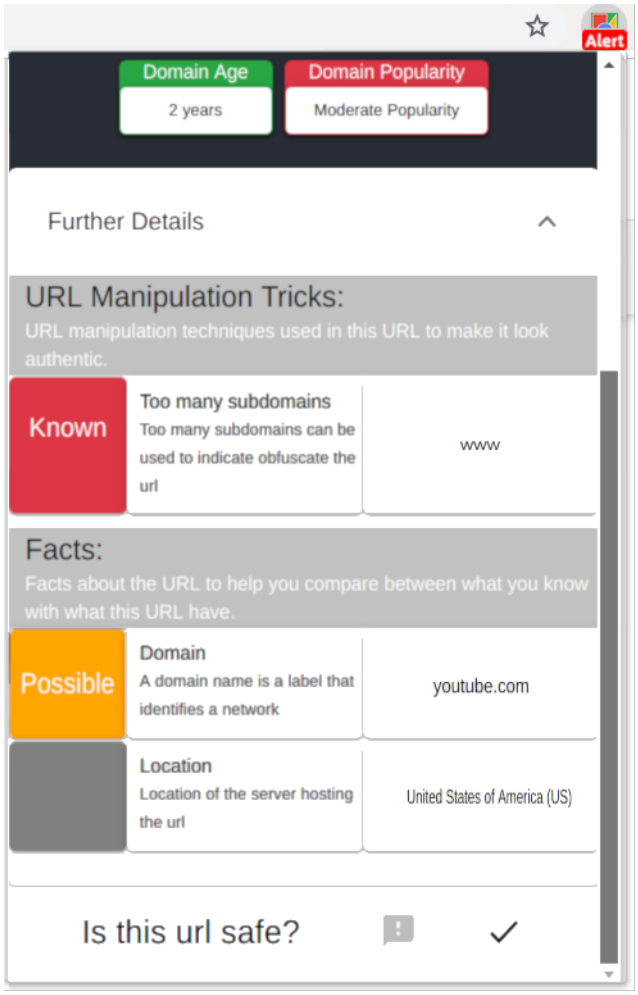

Figure 6.6: Popup Further Details

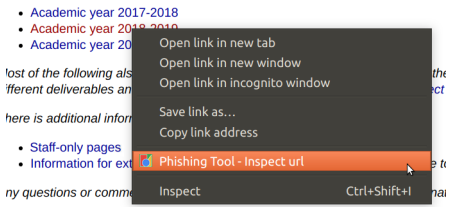

Figure 6.7: Extension Context Menu Entry

by the frontend, it is the sent to it.

This uses a chrome extension API called contextMenus to implement this feature.

6.4.2.2 Badge System

A badge is a small icon that appears over the extension icon itself. This can include up to five characters in text and coloured how the developer wishes. Therefore a limited amount of information can be conveyed by a badge.

The badge system is used as part of my system to display the status of the URL in the top bar on the page load, so the user has an immediate idea of how safe the current site URL is. This is also updated for any URL in the popup that is being analysed such as a link in the page, called by the context menu. This helps to keep the badge consistent with the status information displayed in the extension user interface itself.

Figure 6.8: Extension Top Bar Badge

The badge is set with a colour depending on what the status of the URL is. This conforms to the traffic light system: safe - green, warn - amber and alert - red. Safe, warn and alert are also the characters used to provide information to the user. This helps to highlight the information for those that perhaps have more difficulty understanding the colours. The badge is updated for each link as the page is loaded.

6.4.2.3 Implementing Link Annotations

Each link in the page is annotated with a small shield as shown, this is demonstrated in figure 6.9. These are used to indicate the status page as are implemented by a combination of using the jQuery library, and pre-stored shield assets, as part of the content script implementation. Each of these shields is a different colour depending on the status of the URL. These conform to the same traffic light system as the badge implementation.

This is intended to give users an immediate indicator of whether a link is safe or not, based on the colour of the shield alone.

Figure 6.9: Extension link annotation

If a user clicks on a link that has not yet been processed, this is immediately prioritised by the tool so that the user can see the necessary information.

6.4.2.4 Intervention page

The intervention page design is drawn from the pre-existing Google site warning, discussed in Chapter 3. This was a design decision taken to increase user understanding of this screen, since they are already likely to be familiar with it as part of using the Chrome browser.

This screen, as seen in figure C.2, is shown to the user, whenever they click on a malicious link, in order to prevent them from reaching it. To meet the tools aim to

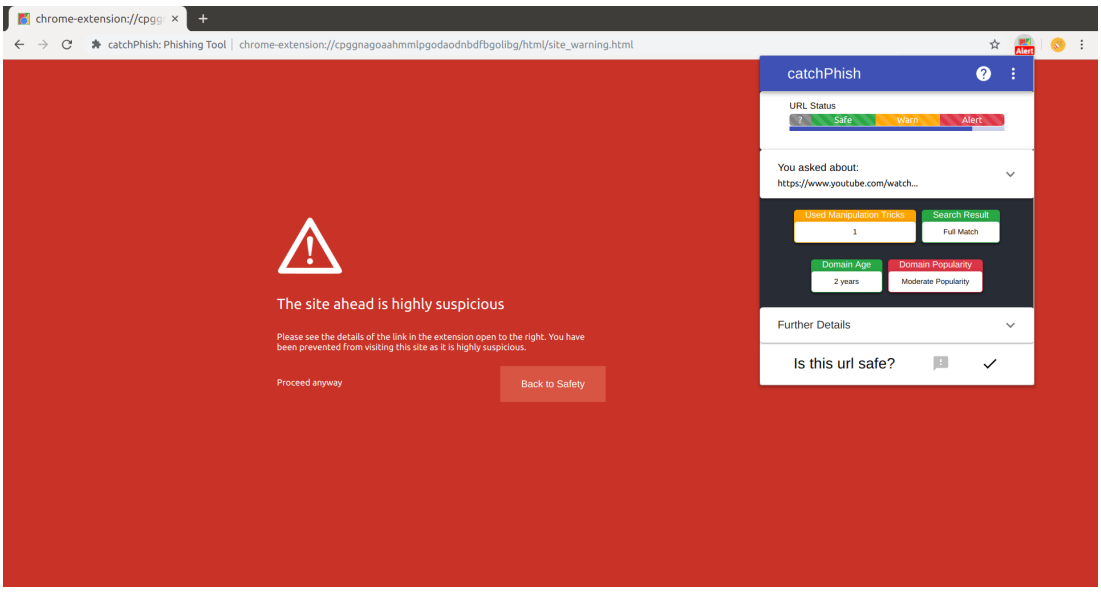

Figure 6.10: Extension Intervention page alongside the extension screen

explain to users why the tool has blocked them from visiting the site, I was able to program the extension such that it would always open at this point.

6.4.3 User support and understanding

As part of the design of the tool, some further pages were designed and added as part of the extension. The intention of these pages was to make the tool easier to use.

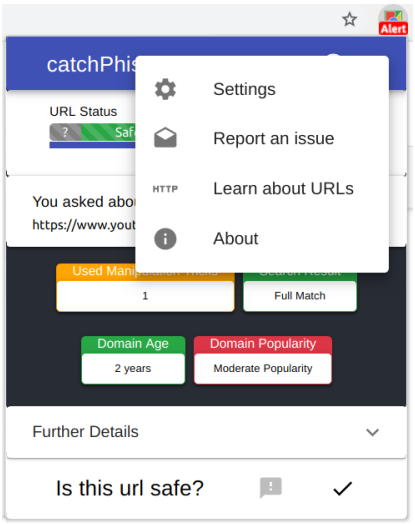

Figure 6.11: Extension Settings Menu

6.4.3.1 Tutorial and Learn URLs Pages

The tutorial and learn URL pages were both implemented with the goal of providing users more context on the tool and the URL subject area respectively. These are both implemented in a similar way, incorporating the text and images view favoured by users in the interviews.

Each image on the page has an associated text description which provides greater context to the user.

6.4.3.2 Report an Issue page

The report page was added to allow users to be able to provide feedback on the tool in case of bugs or issues that might occur. This is a common feature of a lot of tools and extensions. Developers can find this useful to get further information on issues they otherwise might have missed, due to the limited ability to exhaustively test the application, and this allows the tool to be more easily improved with further iterations.

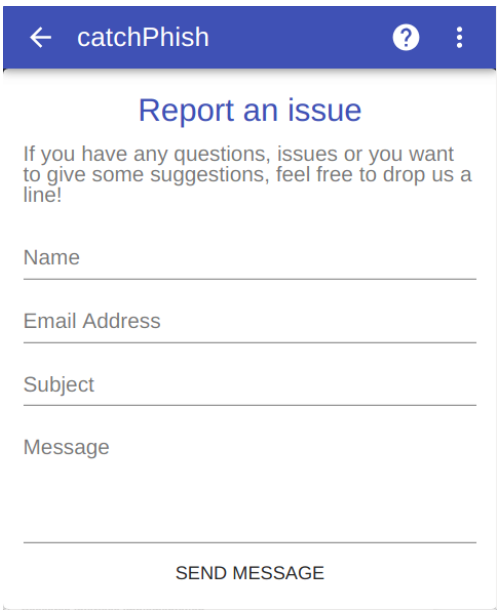

Figure 6.12: Report and Issue Page

Figure 6.12 illustrates the page itself: a simple form allows the user to send the developer an email with the content of their choice. This was one of the few screens that was not designed before implementation, the inspiration being drawn from other tools during development. This page is integrated with the main extension view.

6.4.3.3 Settings Page

The settings page was added with a limited range of settings available for the user to toggle. This page will be extended with further user options, but further research will

have to be derived to find what these are. The main settings that are core to the tool are the ability for the user to delete their data. Also, another setting is the ability to turn off link annotation. This was a setting that arose from some users' desire for very minimal tool obtrusiveness. The tool also provides users with the option of sharing additional analytics data, the whitelisted and reported URLS, with the server to improve the tool's overall performance. The tool does not currently send this information to the server due to security concerns, as outlined in Section 6.6.1.1.

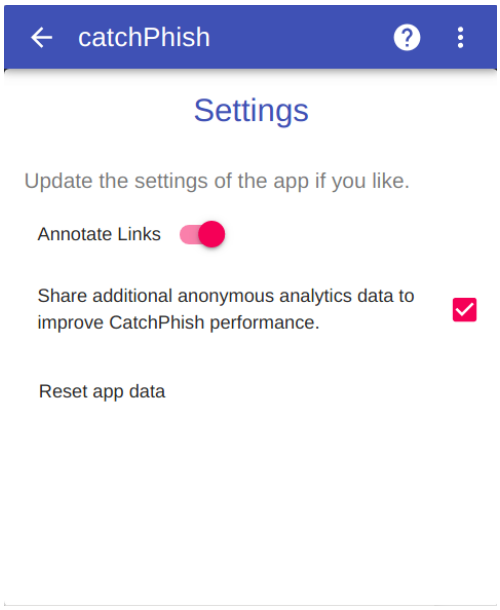

Figure 6.13: Extension Settings Page

Figure 6.13 illustrates the settings page and the options available. This could be further improved in future work by making it more polished such as by improving the selection options' alignment. With the addition of further settings, I could add more sophisticated navigation. This page was designed to be clear and readable to the user.

6.4.3.4 About Page

The about page is another feature which is common across tool and applications. It is useful for reminding users of the purpose of the tool and giving them more contextual information on its development.

Figure 6.14 does this simply by outlining the goal of the tool, when it was built and who participated in the project. It does this in a clear way with different size of text to highlight the different information.

6.5 Decision-Making Server

The decision-making server is responsible for calculating the status of the URL and running each of the heuristics used to calculate this. It is also responsible for calculat-

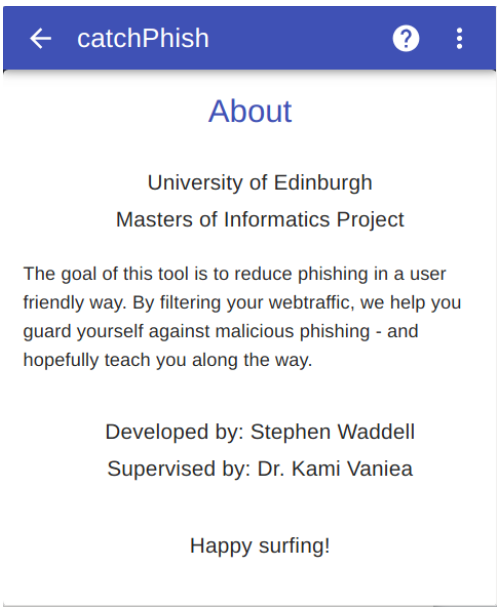

Figure 6.14: About Page

ing how the heuristics will be displayed to the user as part of the User Interface.

There is still more to build and develop with the server and decision making. This is outlined and justified in more detail in Chapter 8. What has been implemented this year is the overall algorithm, the code structure for all the heuristics with placeholder code, around four of the URL decision making heuristics and the User Interface calculations.

Figure 6.15 demonstrates the modular structure of the server code and illustrates how each component fits into the server. This was derived by running the bash tree command in the server directory.

6.5.0.1 Calculating the URL Status

The overall algorithm was implemented as outlined in the design Chapter 5. Currently, the server runs each of the heuristics, which returns the calculated status value, the section of the URL it refers to (if any), and the title of the heuristic. Each of these is added to an object containing all the heuristics of that set, and these, in turn, are added to an object containing all the heuristics. To calculate the status value, a count is taken of the occurrences of the status values across all the heuristics. This is processed by the algorithm which returns the overall status value of the URL.

6.5.0.2 Dynamic User Interface Construction

The tool only shows the heuristics which the tool has detected as part of the URL and no more, so these are what is generated. The intention is that it dynamically generates each user interface statement based on the contents of the URL. For instance, when explaining the domain of a URL to the user, it includes the domain of the current URL

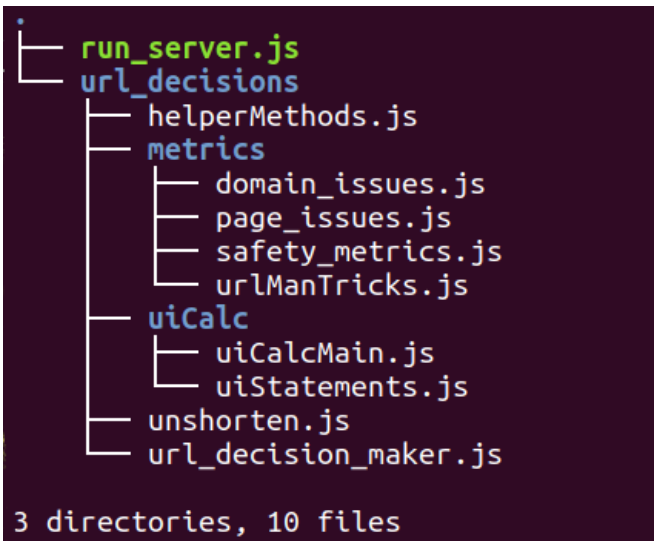

Figure 6.15: Server file tree

in the tool to display to the user. This is the main user interface element calculated by the tool, and the decision was made to handle this server-side as this allows for more adaptability with the potential addition or subtraction of further heuristics in the future (as well as its consistency across users for the same URL).

The server stores a list of pre-crafted summary statements in one object, one for the title of the issue and one for the explanation. The title remains consistent across URLs but the explanations have been crafted with blank spaces to be filled in depending on the URL. Where relevant, the section of the URL stored in the heuristic calculation is used to replace this value. There are multiple explanations for different issues as they have to be expressed to the user depending on the issue level of the heuristic.

6.5.1 Implemented Heuristics

To demonstrate the server is working effectively, I choose to implement some of the heuristics which could be done without the need for API integration. Four URL decision-making heuristics were chosen to achieve this, due to their ease of implementation.

One example of these heuristics is *too many subdomains*. The extension uses the domain which is part of parsed URL information to isolate the subdomain. To do this, the domain is split by each of the full stops and a count is taken of the subdomains it contains. This approach was chosen due to the need to implement different issue levels for each heuristic as outlined in Chapter 5. This allowed me to set thresholds such that the number of subdomains might indicate it is a known issue, rather than a possible issue. This causes the tool to incorrectly flag the University's Informatics websites as malicious sites. Being safe sites, this is clearly not ideal, so the threshold will need to be further refined.

6.5.2 Unshorten Links

A further feature I decided to incorporate into the decision-making server this year, was the ability for the tool to discover the destination of shortened links. This was chosen as an implementation priority after the interviews showed participants' inability to understand these links.

To do this, I was able to adapt an existing web application - **unshorten.link** [70] which provides a dedicated service to expand shortened links. I found that I could use this service as part of the CatchPhish system by making use of the site's URL query string. Each of the URLs proceeded on the site can be done by updating the sites query string with the URL that the system wishes to unshorten, such as in the example URL below.

https://unshorten.link/check?url=*http://www.ign.com*

To incorporate this into the extension I created an API to request the details from this service. I first check if a URL conforms to the pattern of a shorten link before sending a request to the service. This returns the site's HTML, which I parse to extract the unshortened URL. This list of shortened link services is stored locally on the extension which has the limitation of becoming outdated in the future,as such this would be a future I would wish to further develop in the future.

6.5.3 Chrome Extension API

To interact with the server, I made a purpose-built API that the chrome extension could use.

For each URL, the extension first checks it's local cache to see if it has already been processed (and requested) by the server. If it has, then it returns those values, and it makes the request to the server.

This works by sending each URL to the server as part of the server's reference query string. This was a decision taken for development speed and does not represent good security practice whatsoever since the user's history could constitute sensitive data. This is one of the areas I will improve upon next year. Once the URL is processed the server sends back a JSON [37] string which is parsed, and the values are stored in its local cache (for the users' future access). The extension then consults the user's whitelist and report list to see if the URL has been stored in there. If it has, then the status value calculated by the server is updated. If the server calculated value was alert and the URL has been whitelisted, then the status value is updated to alert. If the URL was green or amber and the URL has been reported, then the status value is updated to alert.

JSON as a lightweight data-interchange format was chosen due to its debugging readability and its easy integration with the chrome extension framework and the decision-making server itself.

6.6 Additional Considerations

6.6.1 Efficiency and Security

To protect user security several considerations had to be made. First of all, secure data storage was considered. It would have been a key concern to store all of the URLs a user visits in plain text. This would have meant that users, on an attack, could potentially have had their entire web history shared, along with the occurrences of, any potential attacker. This consideration was solved by hashing and salting each of the URLs before addition to the data structure. This means that each URL was treated with the same level of protection as a user password might in the best case security model.

The URLs are converted into a one-way string representation of some finite size, or hash, and are then salted, by mixing it with a randomly generated string called a salt. This hash is then stored alongside the salt so that it can be reapplied when the URL needs to be compared. Whenever a URL needs to query the list, it is hashed and compared with the unsalted hashes in the list to check if it is included in the list. The reason salting is employed is to ensure that the hashes cannot be easily compared against lists of known hashes in the application. To do this with the application, I used the *crypto* [48] library as part of *node.js* as this incorporates a set of well-tested functions to allow developers to easily incorporate these features into their *node.js* applications.

In the CatchPhish system, hashing is implemented using the *SHA-512* hashing algorithm and used to secure the storage of the URLs within the user's personal URL lists. All that is needed for this feature to work effectively is the knowledge of whether a URL is included in one of these lists. This is why I have implemented hashing for the personal URL lists.

6.6.1.1 Further Security Improvements

The CatchPhish system could be improved in the future by encrypting the communications between the server and the chrome extension to protect users' privacy. Since the URLs sent to the server need to have their integrity maintained for them to be both effectively calculated and displayed as part of the front-end, this could be done with the addition of symmetric key encryption. To provide communication security between these two primary components. Further to this, the decision-making can focus on pulling information from the databases such as WhoIs [80] to the extension, rather than querying such databases for individual URLs. This would have the further benefit of reducing the latency when checking each URL (improving the efficiency of the application for the user), and increase the general userbases' privacy by reducing the number of URLs which could be maliciously intercepted. This would, however, come at the expense of an increase in the memory consideration of the extension.

6.7 Summary

This chapter outlined the various components of the CatchPhish system. These components are:

- the chrome extension infrastructure
- the chrome extension front-end - implemented as a React app
- the decision-making server for calculating both the URL status and user interface contents

It discusses how each section of the system was designed and implemented throughout the course of the project.

Chapter 7

Evaluation

The focus of the evaluation was to build an understanding of the usability of the tool and how successful I had been in completing the user interface element of the project this semester. To do this, I chose to use three different usability studies in a process call triangulation to build a picture of how usable the system was from different perspectives.

7.1 Study Preparation

To prepare for the studies, I needed to create a closed environment that I would be able to use to demonstrate the tool functioning. This was important with the limited amount of heuristics implemented as part of decision-making server. The resulting preparation allowed the tool to be fully demonstrated by making use of certain pre-processed URLs embedded in an example web-page.

The goal was to develop the example website with a variety of anchor tags using different styles: those incorporated within images and icons, and those with a plain text appearance. This was to demonstrate how the tool interfaced with each of these types. Faced with the option of creating a whole new site for this purpose or adapting an existing site, I chose the latter. The primary justification for this decision was the amount of time it would have taken to develop a purpose-built website from scratch. Particularly when considering the time taken to develop it to the high quality required for the studies, with the resulting site having limited useful beyond its use in the subsequent studies.

Instead, I borrowed the template of a popular site that incorporated the variety of anchor tag styles that I was looking for - **ign.com** [35]. I used a website cloning tool called *httrack* [33], to clone the index file and associated CSS files of this site. Since my tool does not function with a user's static local files, I implemented a *node.js* server, similar to that outlined in section, to display these cloned **ign.com** files on request. This allowed me to display the full functionality of the tool, including the tool's link annotation features.

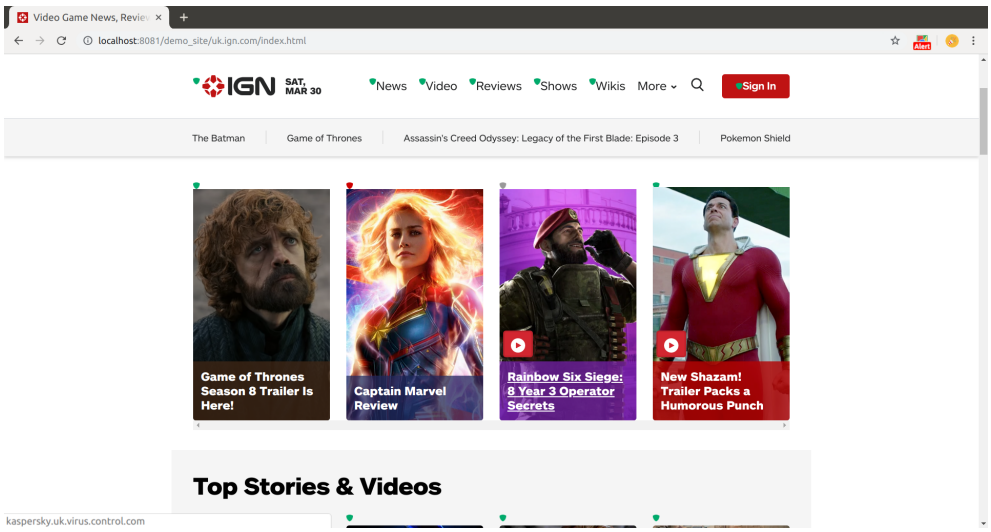

Figure 7.1: Demo site using IGN template

7.2 Demonstration with System Usability Scale

As part of the Informatics project demonstration day, I was able to conduct a study into the usability of the application. This is an event which encourages everyone from the Informatics academic body to attend and view the results of the undergraduate projects.

As part of this, I set-up a stall, created a poster and enjoyed the buffet. I was also able to use a large wide screen television, lent to me by my supervisor, to demonstrate my tool in the environment.

I demonstrated the features of the tool for each person that attended my stall, and gave them a short opportunity to interact with the tool. Afterwards, each person was asked to complete a short paper survey, which can be viewed in Appendix E. This survey contained the questions which form the System Usability Scale, alongside basic demographic details such as the gender and the year of the student. The questions included in the survey are drawn from the evaluation method itself, and the survey template itself was made as part of previous years' usability projects.

7.2.1 Methodology

The System Usability Scale is a quick means of measuring the usability of the tool. It consists of a ten item questionnaire with five response options for respondents: from *Strongly Disagree* to *Strongly Agree*. The benefit of this approach is that it is very quick to complete, and therefore easy to administer to participants, whilst giving a strong approximation of the result.

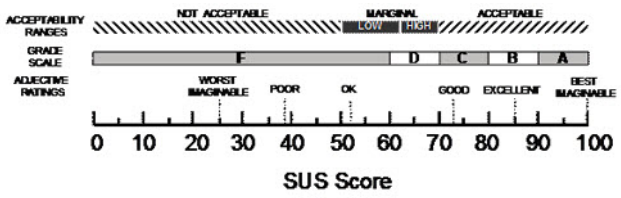

Figure 7.2: An overview of SUS scores and how they evaluate to a grade[7]

7.2.2 Results

As part of the demo, I was able to get 43 responses to the survey. After calculating the score, involving a large amount of data entry, my tool got a score of 81.3 - which puts it in the range of very good. To better understand this score, I broke the score down by the various demographic groups of the participants involved. There was no significant deviation in the results, and the score was broadly consistent based on the student year - so computer science experience had no noticeable impact on the score.

7.2.2.1 Further Respondent Feedback

Other than the score itself, I received a lot of feedback from respondents. A lot of people when faced with the tool found it difficult to understand who the tool was targeted for, and this required explanation. Many participants felt the tool would be very useful for their parents rather than themselves, as they felt they have a strong knowledge of phishing and URLs themselves.

This might suggest there is an attitude problem among participants: in that they have a high perceived amount of knowledge which not always match their training in the phishing field. The best way to test this would be a longitudinal survey with a focus on measuring the participants knowledge of the topic before and after using the tool.

Some participants also thought the inclusion of a confidence value beside the URL status would be a useful feature for the tool. After discussing the purpose of the tool with them, the users were less keen on the addition of this feature. This was because I explained that one of the purposes of the tool is for users to develop their own confidence in the tool's classification, and by extension a better understanding of malicious URLs. The tool does this by explaining why it has arrived at each classification, in its presentation of the heuristics it has used.

Participants on the whole were largely impressed by the tool, along with the size of the television used to display it. Looking at the breakdown of the questions, as given in Appendix E, participants were more likely to select that there was a high amount of knowledge required to use the tool. This could be due to the demonstrative nature of the demo when compared to the normal usage means of installing and using the tool. This might give users a perception they would need me to discuss the features of the tool, since I was already doing that, in order to use them. However, this will have to be considered further in the future implementation of the tool.

7.3 Think Alouds

The Think Alouds were conducted to get a more detailed understanding of specific issues users might have with the CatchPhish tool. They are one of the most common means of usability evaluation, as they provide researchers a tried-and-true approach to understand how users interact with an application's features. The evaluation method itself involves participants working through a given set of tasks as they use the application whilst talking aloud: voicing their thoughts and actions. The participants' emotional responses allow us to build an understanding of how users actually use the application so that the implementation of it's features may be corrected or improved [31].

The study itself involved a think aloud followed by a brief interview of the participants. This post-interview was intended to capture any information that participant's were not able to provide as part of the Think Alouds - such as their overall impressions of the tool.

7.3.1 Preparation

To prepare for this study, I created a consent form and wrote a script that I could use to ensure my input was consistent across participants. This script was adapted from a template provided as part of the Human-Computer Interaction course. Both of these can be found in Appendix F. The script itself incorporates the interview questions I created and asked on the conclusion of the Think Alouds.

The tasks themselves were intended to get participants to use the breadth of the tool's features. These focus on the participant's using each of the tool's various features.

7.3.2 Methodology

As part of the think aloud, the participants worked through seven tasks that would demonstrate all the functional elements of the tool. The tasks participants were asked to complete are:

1. Find a malicious URL on the page, and proceed to the page after viewing the details of the URL
2. Find the settings page and turn the annotate link feature off
3. Use the tool help menu to read more about the tools web redirection features
4. Analyse the details of a URL on the webpage
5. Add a URL to your whitelist
6. Analyse any URL on the page and report it
7. Use the tool to report an issue with the webpage

For example, the goal of task one was to understand how users would interact with the links on the page: whether they would interact with the badges, use the context menu, go to the page itself or just not understand the problem altogether.

When the participant indicated that they had completed all of the tasks, I then asked a few brief questions about their experiences with the tool. This allowed participants to provide further information about their general experiences with the tool, and express the particulars of what they found easy or difficult when using the tool.

For each think aloud, a written transcript of the participants responses was made. An accompanying audio recording was made to bolster the usefulness of the written transcript, and account for any information that may have been missed.

7.3.3 Results

The results of the Think Alouds were analysed using open coding, with a subsequent thematic analysis, such as in section 4.2.

The data analysis was split into two key sections: feedback on the tool's features (Appendix G) and an explanation of the participants' likelihood to use the tool, given in table 7.1. The full results of the quantitative data from the interviews can be found in the appendix H.

The Think Alouds were largely positive and users on the whole had few reported issues with the tool itself.

7.3.3.1 Analysis of the tool's features

As a result of the study, there were a number of suggestions made to improve the quality of the tool.

One particular aspect to improve would be making the *Report an issue* and *Report a URL* features more clear, as this was a particular source of confusion for participants. This was clear as **P4** and **P8** were not able to complete tasks 7 and 8 respectively due to the confusion between these features. **P4** in particular thought they had solved the task by using the *Report a URL* feature and therefore did not find, or check for the *Report and issue* feature in the settings menu. This was a similar but opposite case for **P8**. This seemed to be due to the similar names of the features and could be improved in further iterations with a simple name change.

The participants further highlighted more minor issues with the tool which did not impact their completion of the tasks. Specifically, they highlighted that there could be more user confirmation when using certain features in the tool - such as when reporting an issue. They also felt that the *whitelist* feature could be made more clear, with some participants being unsure what this feature was initially. This suggests a primer for the participants for the tool's key terms would be useful. This would compliment participants suggestion in the interviews for a guide on first install.

Open Codes	Themes	
Doesn't know any other tools with functionality; impressed by tools quality; would recommend the tool; very easy to use	Positive Impressions	Reasons to use tool
Clear information immediately; redirection feature would be useful; thinks the tool is useful in a security context;	Useful features	
Suggested for a less technical person; perception they don't need it; already know enough about urls; feel browser habits are secure	Already know enough	Reasons not to use
Concerned about the amount of false positives	Concerns	
Already have a lot of extensions; doesn't use extensions	Extension Use	

Table 7.1: Think Aloud Likelihood of use open codes

There were a number of features that users liked within the tool. For instance, many users liked the link annotation feature and the ability to undo an action. This complimented users' impression of the tool who were unanimously impressed by the quality of the product. The ability to navigate around the tool was also a positive feature of the UI that was highlighted: with multiple participants feeling it was easy to navigate around the tool. Users were also able to understand the popup explanations being one of the tool's most well understood features - with users highlighting the clarity of the URL analysis and intervention page in the features they like.

When asked in the interview, participants on average found the tool to be *Fairly easy to use*. This suggests participants are happy with their experiences when using the tool.

7.3.3.2 Participants' likelihood to use the tool

When asked how likely each participant was to use the tool, participants indicated they were *Fairly likely* to use the tool, with only one participant indicating they were less likely than this average - giving a response of *Somewhat likely*. When asked why they gave their rating, a surprising amount of users suggested they did not need the tool as they already had enough knowledge. This continues the trend seen with the demonstration day surveys, and from the results of the interviews. This may need to be tackled in future work, potentially by making the frequency of phishing attacks and their exposure clearer to users.

7.4 Expert Evaluation

For the final evaluation I consulted two experts in phishing, the PhD student I had been working alongside - Kholoud Althobaiti and Sara Albakry, an informatics PhD student researching phishing URLs. As part of this expert evaluation, I provided a brief

overview of the project goals, a demonstration of the tool features in use as well as a analysis of the code.

Both experts overall impressions were very positive about the tool. Kholoud was particularly impressed with the amount and quality of work achieved in the limited time since our design evaluation. Sara was similarly impressed and envisioned a future for the tool, beyond its usage as a user focused tool, as a research platform that would allow academics to experiment with the different possible combinations of user intervention and phishing issue presentation to a wider audience - particularly as this is an area of active research.

7.4.1 User Interface

Both experts really liked the link annotation feature, and it was suggested this could be a novel contribution to the field. This would have to be verified in a future literature review however. They specifically praised how the link annotation feature interacts with all web pages. There were some areas they thought this could be improved. Kholoud and Sara were both concerned with how users interacted with the links on the page, and how the shields played into this. Kholoud suggested that one improvement could be allowing the shields to open the popup itself. Sara felt for users unfamiliar with the extension, that it was possible that the shields might be confused with the design of a web page itself. They believed this could be further evaluated in an additional study by comparing how the shields appear across multiple sites including different content. This would allow be useful to understand how user interact with these, and the utility of this feature in a security context.

Sara thought it was useful that the tool only highlights external links in a given web page. This was expressed because it reduces the amount of information on the page, and demonstrates the assumption that sites are generally safe if they still within the same domain as the origin site.

Kholoud also suggested some further improvements to the main popup user interface. She recommended the tool should ensure that the URL domain is always displayed in the main view, as research shows this is the most important feature to highlight to users. She further suggested that the tool could incorporate the work on colour blindness that the URL report [4] had incorporated. However, Kholoud also expressed an understanding that this was a very difficult problem to solve, out-width the scope of this project.

7.4.2 Implemented Backend

Both experts had further advice on the implementation of the tool's backend, and specifically the implementation of the server heuristics. Kholoud advised that the implemented heuristics could be improved by using specific Javascript modules rather than personally crafted regular expressions, even though the implementation of these was sufficient. As these modules typically have more features, and are well tested

among the Javascript community, this would be more desirable. For instance, she recommended implementing URL unshortening by using a redirect package to get the full history of the URL requests. Sara highlighted the incorporation of this package too. Particularly, as this would also be useful to implement the redirection heuristic (such as in Appendix D).

Kholoud also thought the optional gathering of the URLs that users choose to report on the whitelist and server, was a useful feature. However, they felt this could be improved beyond the outlined security improvements, by analysing the URLs for personal details. For instance, information hidden in the query strings that users visit might contain personal information.

7.4.3 Further Advice

Security considerations with the server communication were also highlighted by Sara specifically. After discussing this, Sara was happy with the suggestions I outlined for improving these security considerations, as part of my work next year.

Sara also highlighted that it was essential that the tool explain the terms that it uses more clearly. This is due to an over-saturation of terms in computer security, which are shown to limit users' understanding of the overall computer security picture, as discussed in Chapter 3. Sara highlighted that the tool should ensure that terms such as 'safe' are quantified with relation to the specific phishing scope of the tool. For instance, by highlighting what the tool actually marks as safe before the user downloads it. They also suggested this might be less of a concern because of above average technical experience of the intended user-base.

Kholoud complimented the amount of work occurring in the project, and expressed an understanding of the difficulty of implementing the research paper user interface in this field. The improvements outlined, and also the positive feedback given to the project, are highly appreciated mean a lot to me and my work over this piece.

7.5 Discussion

Overall the tool is popular with users across the various studies. The results of these evaluations were positive about the usability of the tool, as given by the SUS rating of *Very Good* and the average response of *Fairly easy to use* from participants during the Think Alouds. The evaluations also highlighted there were several items which could be improved next year.

In particular, improving the names of the features so that users are more familiar with them, will one of the priorities for improving the User Interface. Another priority will be to display the tutorial as a start up guide for users so they become familiar with the key terms of the tool.

The expert evaluation also highlighted some areas of the system design that could be improved. For instance, the security aspects and the methods of approaching some of the heuristics. This feedback will be very useful in implementing the subsequent heuristics next year.

The positive results of these evaluation are useful when considering how the requirements of the tool have been met.

7.5.1 Requirements of the tool

The requirements derived for the tool, as outlined in Chapter 4 have been largely met as part of the implementation and design of the CatchPhish system.

Through the implementation of the system, the first requirement, *to classify every URL according to one of three states as part of a traffic light system*, has been met. The traffic light aspect of this requirement has been met with the addition of both the link annotation and badge features which highlight the calculated status of each URL according to a colour associated status value. The classification of every URL has been met through the implementation of the content script, and the processing of every URL on the page by the server.

The implementation of the web request intervention feature and related intervention page, have allowed the second requirement *to use the browser to prevent the user visiting any URLs that have been classified as malicious*, to be met as well. The need for users to click to continue to the blocked URL on the intervention page also meets the need for the explicit user consent outlined as part of this requirement.

The third requirement *to present the information on each URL to the user in an understandable way, both at appropriate points of intervention and on user request* was more complex to meet, but this has also broadly been achieved. The results of the Think Aloud study indicated users felt the information presented within the URL analysis was clear. This helps to meet the first part of this requirement that the information should be presented in an understandable way. Since each user can request information about any URL on the page, by clicking on both the extension icon and using the context menu feature, this meets the user request aspect of this requirement. The intervention feature also helps to meet the appropriate points of intervention part of this requirement. A final conclusion on whether the requirement for appropriate intervention has been met cannot be made until the accuracy of the tool has been further tested.

7.6 Summary

This chapter outlines the three different evaluations techniques used to evaluate the usability of this tool:

- Demonstration with System Usability Scale Survey

- Think Aloud studies
- Expert Evaluation

It outlines what the results of these were and some suggestions are made for areas which could be further improved next year. The positive results of these studies and discussion of the implemented components themselves, are used to evaluate how well the requirements outlined in Chapter 4 are met. These are broadly regarded to have been achieved, with an understanding of how these requirements relate to the user interaction focus of the project this year.

Chapter 8

Further Work

While I have done several significant pieces of work as part of the project this year, there are still many tasks I plan to implement next year as part of the final year of the Minf.

I have tried to ensure there is a self-contained section of the project to help make the work manageable over the two years. The main focus of my work next year will be implementing the decision-making code and the further tasks outlined in this chapter.

8.1 Decision-Making Components

The URL decision-making components will involve implementing each of the heuristics outlined in the back-end design report, discussed in Chapter 4. This may involve a further literature review to update and improve the outlined heuristics. Whilst the server and implementation structure has already been implemented, the heuristics themselves need to be completed.

It was recommended in the design evaluation that the URL decision-making heuristics each need their own set of metrics to evaluate which status value they should return. This will involve further research to justify these metrics.

Implementing this code also involves integrating the CatchPhish system with a number of APIs to get the requisite data for each heuristic. For instance, to implement the unusual top-level domain heuristic, a list of the most common top-level domains will have to be gathered. In order for the tool to be adaptable to internet trends, this has to be dynamic and therefore requires using an API to allow for this. Alternatively, I could hard-code the values of the top level domains. However, this would limit the tool's future utility to users.

To increase the efficiency of the tool, a further plan is to pre-populate the whitelist based on the user's history. Research into this feature was conducted this year. This research found the user's history cannot be directly accessed by a Chrome extension, only the amount of times a given URL has been visited by the user. This requires

knowledge of the URLs in the user's history in order to be able query them. To implement this, the user's history can be queried using the user's top sites and bookmarks to find the number of times the user has accessed these URLs - with a high amount suggesting these can be marked as safe. Therefore this research just needs to be implemented in the tool.

8.2 User Interface Improvements

The user interface has some improvements that need to be made as well. These are specifically related to user interface elements, closely related to the server decision-making heuristics.

To fully implement the design laid out in the report, each heuristic requires a sentence explaining what it means for each heuristic state. Each of these sentences needs to be dynamically generated based on the contents of the URL. This means, for instance, that the domain of the URL has to be injected into the correct part of the explanation sentence. The challenging part of this is creating each of theses sentences since they are closely related to the implementation of the heuristics. Therefore they cannot be created until the heuristics themselves have been implemented next year.

The goal of highlighting this information to the user by displaying it in a different colour, is coupled with this. Since there are so many possible sentences, it is unfeasible to hard-code these in the React front-end itself, which would also reduce the system modularity. Instead, the goal is to craft the React components on the server-side and have the front-end render them. This has proven challenging within the scope of the project this year, and due to the strong link with the back-end code, this has also been allocated to next year's work.

In addition, since by the end of next year the intention is to publish the tool, I also aim to polish the user interface for the tool if I have time. This will involve minor improvements such as animations, text styling and incorporating the feedback given in Chapter 7.

8.3 Longitudinal Study

In addition, to build an understanding of the efficiency, accuracy and educational value of the tool for users, the further work for next year will include a longitudinal study to gather information about these values. This will involve asking a select number of users to use the tool for a period of time.

To test the efficiency of the tool I can add analytics into the tool, for the purposes of the study, to measure the page load times in order to measure how long it takes the tool to calculate and mark each URL on a visited web-page. The intention is that this would not capture personal user information such as the pages the user is visiting, and only

the usage statistics. These analytics along with user reported issues will hopefully give a good understanding of how the tool works.

To evaluate the educational value of the tool I intend to do pre- and post- testing with the users to evaluate their understanding of malicious URLs, before and after they use the tool for a significant amount of time. I can use this testing to measure how much users have learnt when using the tool (and filter this information by how much they use the tool). This could also be followed up with a later test to measure how well users retain the information after using the tool, such as that conducted by Canova et al. [11]. This may not be necessary as the tool is intended to be permanently used by the users rather than for some finite time such as with the NoPhish app.

This has been planned for next year due to the significant amount of time this will take. Along with the need for the back-end and user interface components of the tool to be fully completed before this occurs. Therefore, whilst there has been a significant amount of work done this year, there is still a generous amount of work planned.

8.4 Summary

In this chapter, I discuss the work planned to be completed next year. This work includes:

- Implementing the Decision-Making heuristics on the server
- Adding the user interface features related to the heuristics and polishing the UI
- Conducting a longitudinal study after the implementation of the tool is completed to understand the effectiveness of the tool

I provide an overview of these components and discuss what will be needed to complete each component.

Chapter 9

Conclusion

9.1 Overview

As we have seen, phishing is an increasing security concern, which due to a lack of knowledge on the primary way it is spread, impacts all users including those with extensive training experience on the subject. To tackle this problem, my project was to develop a phishing learning and detection system, to answer the question:

How might we develop a phishing learning and detection tool that will protect from, and inform users about, malicious URLs?

I chose to develop the system as a browser extension to filter all of a user's encountered URLs and chose Chrome as the browser to develop for due to its dominant market share.

The system was developed for above-average technical users since these are shown to be the most beneficial group to teach about computer security matters. To better understand how these users would like to interact with such a system, I conducted interviews with 17 participants. As a result of these interviews and a prior literature review, the following requirements were able to be defined:

1. Classifying every URL according to one of three states as part of a traffic light system
2. Using the browser to prevent the user from visiting any URLs that have been classified as malicious without explicit user consent
3. Presenting the information on each URL to the user in an understandable way, both at appropriate points of intervention and on user request
4. That the end built system includes the best practice of software engineering: maximising efficiency and accuracy

From this basis, I outlined the design of a system, including the design of a URL analysis algorithm, which was subsequently evaluated by an expert in the field. This

allowed me to develop the user interface and infrastructure of the system and complete the goals outlined for the project this year.

The system was evaluated with regards to its usability, and how well it met the system requirements. The goal was to assess how suitable and easy to use this system is for analysing the details of malicious URLs and preventing users from visiting them. To do this, I conducted a survey of 43 participants who all filled out the SUS survey, eight Think Alouds and a further expert evaluation.

In conclusion, it would seem that my system is usable and not overly complicated, which fulfils the goals that were intended to be achieved in the implementation of the user interaction and system design.

Bibliography

[1] Anne Adams and Martina Angela Sasse. Users are not the enemy. *Commun. ACM*, 42(12):40–46, December 1999.

[2] Github AlDanial. cloc - count lines of code. https://github.com/AlDanial/cloc, [Accessed March 30, 2019].

[3] Alexa. The top 500 sites on the web. https://www.alexa.com/topsites, [Accessed March 30, 2019].

[4] Kholoud Althobaiti and Kami Vaniea. Is this url safe to click on? supporting users' comprehension of phishing features. In Submission, 2018.

[5] Kholoud Althobaiti, Kami Vaniea, and Serena Zheng. Faheem: Explaining urls to people using a slack bot. 2018.

[6] AngularJS. Angularjs. https://angularjs.org/, [Accessed February 19, 2019].

[7] Aaron Bangor, Philip Kortum, and James Miller. Determining what individual sus scores mean: Adding an adjective rating scale. *J. Usability Studies*, 4(3):114–123, May 2009.

[8] Bitly. Bitly. https://bitly.com/, [Accessed March 30, 2019].

[9] Bootstrap. Bootstrap. https://getbootstrap.com/, [Accessed February 19, 2019].

[10] Canonical. Ubuntu. https://www.ubuntu.com/, [Accessed February 19, 2019].

[11] Gamze Canova, Melanie Volkamer, Clemens Bergmann, and Benjamin Reinheimer. Nophish app evaluation: Lab and retention study. 2015.

[12] Softpedia Catalin Cimpanu. Hidden javascript redirect makes phishing pages harder to detect. https://news.softpedia.com/news/hidden-javascript-redirect-makes-phishing-pages-harder-to-detect-505295.shtml, [Accessed March 30, 2019].

[13] Kang Leng Chiew, Kelvin Sheng Chek Yong, and Choon Lin Tan. A survey of phishing attacks: Their types, vectors and technical approaches. *Expert Systems with Applications*, 106:1 – 20, 2018.

[14] Cisco. What is cisco anti-spam's catch rate, false-positive rate, and throughput? https://www.cisco.com/c/en/us/support/docs/security/email-security-appliance/118198-qanda-esa-00.html, [Accessed March 27, 2019].

[15] Lorrie Faith Cranor. A framework for reasoning about the human in the loop. In *Proceedings of the 1st Conference on Usability, Psychology, and Security*, UPSEC'08, pages 1:1–1:15, Berkeley, CA, USA, 2008. USENIX Association.

[16] A. Y. Daeef, R. B. Ahmad, Y. Yacob, and N. Y. Phing. Wide scope and fast websites phishing detection using urls lexical features. In *2016 3rd International Conference on Electronic Design (ICED)*, pages 410–415, Aug 2016.

[17] Material Design. Material design. https://material.io/design/, [Accessed February 19, 2019].

[18] Rachna Dhamija, J. D. Tygar, and Marti Hearst. Why phishing works. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 581–590, New York, NY, USA, 2006. ACM.

[19] Inc Dropbox. Dropbox. https://www.dropbox.com/?landing=dbv2, [Accessed March 30, 2019].

[20] The Apache Software Foundation. The apache software foundation. https://www.apache.org/, [Accessed February 19, 2019].

[21] Ghostery. Ghostery. https://www.ghostery.com/, [Accessed March 30, 2019].

[22] Medium Gil Fink. Building a chrome extension using react. https://medium.com/@gilfink/building-a-chrome-extension-using-react-c5bfe45aaf36, [Accessed February 19, 2019].

[23] Github. Github. https://github.com/, [Accessed February 19, 2019].

[24] Google. Chrome extension getting started tutorial. https://developer.chrome.com/extensions/getstarted, [Accessed March 27, 2019].

[25] Google. Chrome extension overview. https://developer.chrome.com/static/images/overview/contentscript [Accessed March 27, 2019].

[26] Google. chrome.webrequest. https://developer.chrome.com/extensions/webRequest, [Accessed March 27, 2019].

[27] Google. Google. https://www.google.com/, [Accessed February 19, 2019].

[28] Google. Google drive. https://www.google.com/drive/, [Accessed March 30, 2019].

[29] Google. Google site warning. [Accessed 01/02/19].

[30] Google. Safe browsing site status. https://transparencyreport.google.com/safe-browsing/search?hl=en_GB, [Accessed March 30, 2019].

[31] Bruce Hanington and Bella Martin. *Universal methods of design: 100 ways to research complex problems, develop innovative ideas, and design effective solutions*. Rockport Publishers, 2012.

[32] Cormac Herley. So long, and no thanks for the externalities: The rational rejection of security advice by users. In *Proceedings of the 2009 Workshop on New*

Security Paradigms Workshop, NSPW '09, pages 133–144, New York, NY, USA, 2009. ACM.

[33] HTTrack. Httrack website copier. https://www.httrack.com/, [Accessed March 30, 2019].

[34] HubSpot. Web design 101: How html, css, and javascript work. https://blog.hubspot.com/marketing/web-design-html-css-javascript, [Accessed February 19, 2019].

[35] IGN. Ign. https://www.ign.com , [Accessed March 30, 2019].

[36] jQuery. What is jquery? https://jquery.com/, [Accessed February 19, 2019].

[37] JSON. Introducing json. https://www.json.org/, [Accessed February 14, 2019].

[38] Kaspersky. What is a drive-by download? https://www.kaspersky.com/resource-center/definitions/drive-by-download, [Accessed March 30, 2019].

[39] Katharina Krombholz, Heidelinde Hobel, Markus Huber, and Edgar Weippl. Advanced social engineering attacks. *Journal of Information Security and applications*, 22:113–122, 2015.

[40] Linkedin. Linkedin. https://uk.linkedin.com/, [Accessed March 30, 2019].

[41] Samuel Marchal, Kalle Saari, Nidhi Singh, and N Asokan. Know your phish: Novel techniques for detecting phishing sites and their targets. In *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, pages 323–333. IEEE, 2016.

[42] Dean Blackbourn Mark Button, David Shepherd and Martin Tunley. Annual fraud indicators 2016. Technical report, University of Portsmouth Center for Counter Fraud Studies, 2016.

[43] MDN. Browser extensions. https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions, [Accessed March 27, 2019].

[44] Thomas Nagunwa. Behind identity theft and fraud in cyberspace: the current landscape of phishing vectors. *International Journal of Cyber-Security and Digital Forensics*, 3(1):72–84, 2014.

[45] Netcraft. Paypal security flaw allows identity theft. https://news.netcraft.com/archives/2006/06/16/paypal_security_flaw_allows_identity_theft.html, [Accessed March 30, 2019].

[46] Lily Hay Newman Netcraft. Internet security and data mining - anti-phishing. https://www.netcraft.com/, [Accessed January 25, 2019].

[47] Jakob Nielsen. *Usability engineering*. Elsevier, 1994.

[48] node.js. crypto. https://nodejs.org/api/crypto.html, [Accessed February 19, 2019].

[49] Node.js. Node.js. https://nodejs.org/en/, [Accessed February 19, 2019].

[50] npm. url-parse. https://www.npmjs.com/package/url-parse, [Accessed February 19, 2019].

[51] University of Dayton. Phishing, scams and spam. https://udayton.edu/udit/safe-computing/spam.php, [Accessed March 30, 2019].

[52] Shari Lawrence Pfleeger, M Angela Sasse, and Adrian Furnham. From weakest link to security hero: Transforming staff security behavior. *Journal of Homeland Security and Emergency Management*, 11(4):489–510, 2014.

[53] PhishTank. Phishtank. https://www.phishtank.com/, [Accessed February 19, 2019].

[54] Erika Shehan Poole, Marshini Chetty, Tom Morgan, Rebecca E Grinter, and W Keith Edwards. Computer help at home: methods and motivations for informal technical support. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 739–748. ACM, 2009.

[55] proofpoint. Quarterly threat report q3 2018. Technical report, 2018.

[56] React. React. https://reactjs.org/, [Accessed February 19, 2019].

[57] E. M. Redmiles, A. R. Malone, and M. L. Mazurek. I think they're trying to tell me something: Advice sources and selection for digital security. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 272–288, May 2016.

[58] Trygve Reenskaug and James O Coplien. The dci architecture: A new vision of object-oriented programming. *An article starting a new blog:(14pp) http://www. artima. com/articles/dci_vision. html*, 2009.

[59] Search Security. email spoofing. https://searchsecurity.techtarget.com/definition/email-spoofing, [Accessed March 27, 2019].

[60] Wombat security technologies. State of the phish 2019. Technical report, 2019.

[61] Lorrie Faith Cranor Serge Egelman and Jason Hong. You've been warned: an empirical study of the effectiveness of web browser phishing warnings. pages 1065–1074, 2008.

[62] Steve Sheng, Bryant Magnien, Ponnurangam Kumaraguru, Alessandro Acquisti, Lorrie Cranor, Jason Hong, and Elizabeth Nunge. Anti-phishing phil: The design and evaluation of a game that teaches people not to fall for phish. volume 229, pages 88–99, 01 2007.

[63] Medium Subodh Garg. How to build chrome extension with angularjs & googles natural language api. https://medium.com/@subodhgarg/how-to-build-chrome-extension-with-angularjs-googles-natural-language-api-370f9a4953e, [Accessed February 19, 2019].

[64] Symantec. Catch rate and effectiveness of spam caught by gateway products. https://support.symantec.com/en_US/article.TECH195964.html, [Accessed March 27, 2019].

[65] Symantec. Internet security threat report. Technical report, 2018.

[66] WH Symantec. Advanced persistent threats: A symantec perspective. *Symantec World Headquarters*, 2011.

[67] tetraph. Oauth 2.0 and openid, covert redirect vulnerability. http://tetraph.com/covert_redirect/oauth2_openid_covert_redirect.html, [Accessed March 27, 2019].

[68] the balance everyday. Internet browsers: A layman's guide to how they work. https://www.thebalanceeveryday.com/what-is-internet-browser-892819, [Accessed March 30, 2019].

[69] tripwire. 6 common phishing attacks and how to protect against them, the state of security. https://www.tripwire.com/state-of-security/security-awareness/6-common-phishing-attacks-and-how-to-protect-against-them/, [Accessed March 30, 2019].

[70] Unshorten.link. Unshorten.link. https://unshorten.link/, [Accessed February 14, 2019].

[71] All About UX. Methods for expert evaluation. https://www.allaboutux.org/expert-methods, [Accessed March 28, 2019].

[72] Melanie Volkamer, Karen Renaud, Benjamin Reinheimer, and Alexandra Kunz. User experiences of torpedo: Tooltip-powered phishing email detection. *Computers & Security*, 71:100 – 113, 2017.

[73] W3Counter. Browser & platform market share. https://www.w3counter.com/globalstats.php, [Accessed March 27, 2019].

[74] w3schools.com. Node.js http module. https://www.w3schools.com/nodejs/nodejs_http.asp, [Accessed February 19, 2019].

[75] w3schools.com. w3schools.com. https://www.w3schools.com/, [Accessed February 19, 2019].

[76] Jingguo Wang, Yuan Li, and H Raghav Rao. Overconfidence in phishing email detection. *Journal of the Association for Information Systems*, 17(11):759, 2016.

[77] Rick Wash and Molly M. Cooper. Who provides phishing training?: Facts, stories, and people like me. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pages 492:1–492:12, New York, NY, USA, 2018. ACM.

[78] Colin Whittaker, Brian Ryner, and Marria Nazif. Large-scale automatic classification of phishing pages. In *NDSS '10*, 2010.

[79] Alma Whitten and J. D. Tygar. Why johnny can't encrypt: A usability evaluation of pgp 5.0. In *Proceedings of the 8th Conference on USENIX Security Symposium - Volume 8*, SSYM'99, pages 14–14, Berkeley, CA, USA, 1999. USENIX Association.

[80] Whois. Whois domain lookup. https://www.whois.com/whois/, [Accessed March 30, 2019].

[81] Ryte Wiki. Anchor tag. https://en.ryte.com/wiki/Anchor_Tag, [Accessed February 19, 2019].

[82] Wikipedia. Email attachment. https://en.wikipedia.org/wiki/Email_attachment, [Accessed March 30, 2019].

[83] WIRED. Google says its ai catches 99.9 percent of gmail spam. https://www.wired.com/2015/07/google-says-ai-catches-99-9-percent-gmail-spam/, [Accessed March 27, 2019].

[84] Lily Hay Newman Wired. Google wants to kill the url. https://www.wired.com/story/google-wants-to-kill-the-url/, [Accessed January 25, 2019].

[85] Min Wu, Robert C. Miller, and Simson L. Garfinkel. Do security toolbars actually prevent phishing attacks? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 601–610, New York, NY, USA, 2006. ACM.

[86] Weining Yang, Aiping Xiong, Jing Chen, Robert W Proctor, and Ninghui Li. Use of phishing training to improve security warning compliance: evidence from a field experiment. In *Proceedings of the Hot Topics in Science of Security: Symposium and Bootcamp*, pages 52–61. ACM, 2017.

**

Appendix A

Interview Materials

RT Number: #3154

‘Phishing Detection and Learning Tool’ Consent Form

This project aims to gather information around users’ understanding of URLs (links) and how they approach both malicious URLs and phishing scenarios.

Today I will be interviewing you about your knowledge of URLs, and the steps you take when presented with malicious URLs. During this interview, I will show you descriptions and screenshots of URLs to discuss your knowledge on the topic. I will then discuss ideas with you about the design and implementation of a proposed Phishing tool, in relation to how you currently deal with malicious URLs.

I will be audio recording during the interview. If you feel uncomfortable about this at any time, you may stop the recording or tell me that the next bit should not be quoted.

This study will be used to learn about phishing and specifically malicious URL engagement, so we can design a tool that is likely to be helpful to students and professionals. The audio and transcripts will be kept for a maximum of one year and then destroyed. Anonymized quotes or short audio clips may be retained longer for use by future students on this project.

The project is supervised by Dr Kami Vaniea (kvaniea@inf.ed.ac.uk) and conducted by myself, Stephen Waddell (s1346249@sms.ed.ac.uk).

[ ] I understand that I am participating in a study as part of the “Evaluate the usability of a security or privacy tool” project.

[ ] I am willing for the audio to be digitally recorded and transcribed for the use as part of the research project

[ ] The researcher may use **audio/ literal quotes** from the interview in publications provided that the quote is anonymised and cannot be connected back to me.

Participant: _____ Date: _____
Researcher: Stephen Waddell Date: _____

Phishing Interview Contents

Start with Consent Form, after signing:

Hi *Participant*, thank you very much for agreeing to participate in our study today on *date* and signing the consent form. It is currently *time*.

Brief interview to cover participant information

To start off, I have a few background questions about yourself and your knowledge of Computer Security:

- Are you currently a student?
    - Whereabouts and what do you study?
    - What year of study are you in?
- What technical experience do you believe you have?
    - If you had to describe according to the scale, which would you choose: (None, Little, Average, Above Average, Expert)
- What experience with Computer Security do you have?
    - For example, have you taken a class in computer security, or has your job offered training in things like identifying malicious communications?
    - Description of Computer Security - provided if participant is confused about terminology at this point
    - If you had to describe this according to the scale, which would you choose: (Beginner, Intermediate, Expert)

Thank you for your that information *participant name*.

Overview for participant of what specifics are required

Just give you an overview of the specifics of the study:

As you might or probably know already: Computer security, also known as cybersecurity or IT security, is the protection of information systems from theft or damage to the hardware, the software, and to the information on them, as well as from disruption or misdirection of the services they provide[1].

[1] https://en.wikipedia.org/wiki/Computer_security

So what will be focusing on today? Well I am developing a phishing detection and learning tool as a means to help train users about, and protect users from, malicious phishing. The purpose of the interviews is to help gather requirements for this tool by better understanding how potential users think and engage with phishing problems.

As part of this interview, we will be covering a number of topics: Phishing, URLs and the Phishing Detection tool itself. As part of this I will be presenting images relating to these topics for discussion, and would like to ask some questions about how you might interact with a proposed phishing detection tool.

Phishing Overview

1. What do you understand by the term Phishing?

Thank you very much for you answer. I can now give you an official definition of phishing.

Phishing is the fraudulent practice of sending emails purporting to be from reputable companies in order to induce individuals to reveal personal information, such as passwords and credit card numbers.[2] Phishing is a form of fraud in which an attacker masquerades as a reputable entity or person in email or other communication channels. The attacker uses phishing emails to distribute malicious links or attachments that can perform a variety of functions, including the extraction of login credentials or account information from victims.[3]

So I am now going to begin showing you some images. For each image, please tell me whether you think the image is phishing or not and why.

Begin presenting images - show 3 images.

Thank you for your advice, I will now give you the answers for each of these images.

Begin presenting and discussing the same images.

URLs Overview

2. What do you understand by the term URL?

Thank you very much for your answer. I can now give you an official definition of URLs.

---

[2] https://en.oxforddictionaries.com/definition/phishing
[3] https://searchsecurity.techtarget.com/definition/phishing

If you've been surfing the Web, you have undoubtedly heard the term URL and have used URLs to access HTML pages from the Web.[4] A Uniform Resource Locator (URL), colloquially termed a web address, is a reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it. [5]

URLs typically consist of three pieces:

1. *The name of the protocol used to transfer the resource over the Web.*
2. *The name of the machine hosting the resource.*
3. *The name of the resource itself, given as a path.[6]*

So I am now going to begin showing you some images. For each url, please tell me whether you think the image is phishing or not and why.

Begin presenting images - show 2 images.

Thank you for your responses, I will now give you the answers for each of these images.

Begin presenting and discussing the same images.

Phishing Detection Tool

Thank you for your help during these topic overviews, I would now like to discuss your thoughts on the Phishing Detection Tool itself.

- What is your browser of choice?
- What kind of features do you think would be useful in a phishing tool?

User Interaction Questions

As I said previously, I am building a Phishing Learning and Detection tool, and an important part of this is how this information is presented to yourself as a user. In a proposed phishing tool, how would you like to see the information about phishing presented to you?

The intention is for this tool to be developed as a Chrome extension on the Chrome browser. The possible User Interface (UI) elements that can be included in a Chrome extension are:

point to each UI element in the browser when explaining

- The extension icon at the top of the browser
- A badge which goes over the extension icon

[4] http://supportweb.cs.bham.ac.uk/documentation/java/tutorial/networking/urls/definition.html
[5] https://en.wikipedia.org/wiki/URL
[6] https://www.w3.org/TR/WD-html40-970708/htmlweb.html

- A popup which can display more detailed information
- A context menu entry which occurs when you right click
- An alert box which pops-up in the center of the page to display information
- Full web-pages that the extension can open

Which combination of UI elements do you think would be most useful to you for displaying the phishing detection information to yourself?

A major component of the tool is training you as a user more about urls. How do you think this information could be presented to you to best help you learn about urls?

Do you think you would need further information on how this tool works if you were to use it? If so, what information would you need and how would you like this information to be presented to you?

How likely would you be to use the tool in the future? Please answer according to the scale:

- Not at all likely
- Not very likely
- Somewhat likely
- Fairly likely
- Extremely likely

For what reason, did you select *insert user's answer*?

Thanking the Participant and Optional Advice

Thanks very much for your time today, *participant's name*, we greatly appreciate your help. If you would like we can give you some pointers about what makes a UI easier to use to conclude the interview.

1. **Check for spelling mistakes:** Companies are serious about their email communications. Legitimate messages usually do not have major spelling mistakes or poor grammar. Read your emails carefully and report anything that seems suspicious.
2. **Analyze the salutation:** Is the email addressed to a vague "Valued Customer?" If so, watch out—legitimate businesses will often use a personal salutation with your first and last name.
3. **Beware of urgent or threatening language in the subject line:** Invoking a sense of urgency or fear is a common phishing tactic. Beware of subject lines that claim your "account has been suspended" or your account had an "unauthorized login attempt."
4. **Review the signature:** Lack of details about the signer or how you can contact a company strongly suggests phishing. Legitimate businesses always provide contact details.[7]

[7] https://blog.returnpath.com/10-tips-on-how-to-identify-a-phishing-or-spoofing-email-v2/

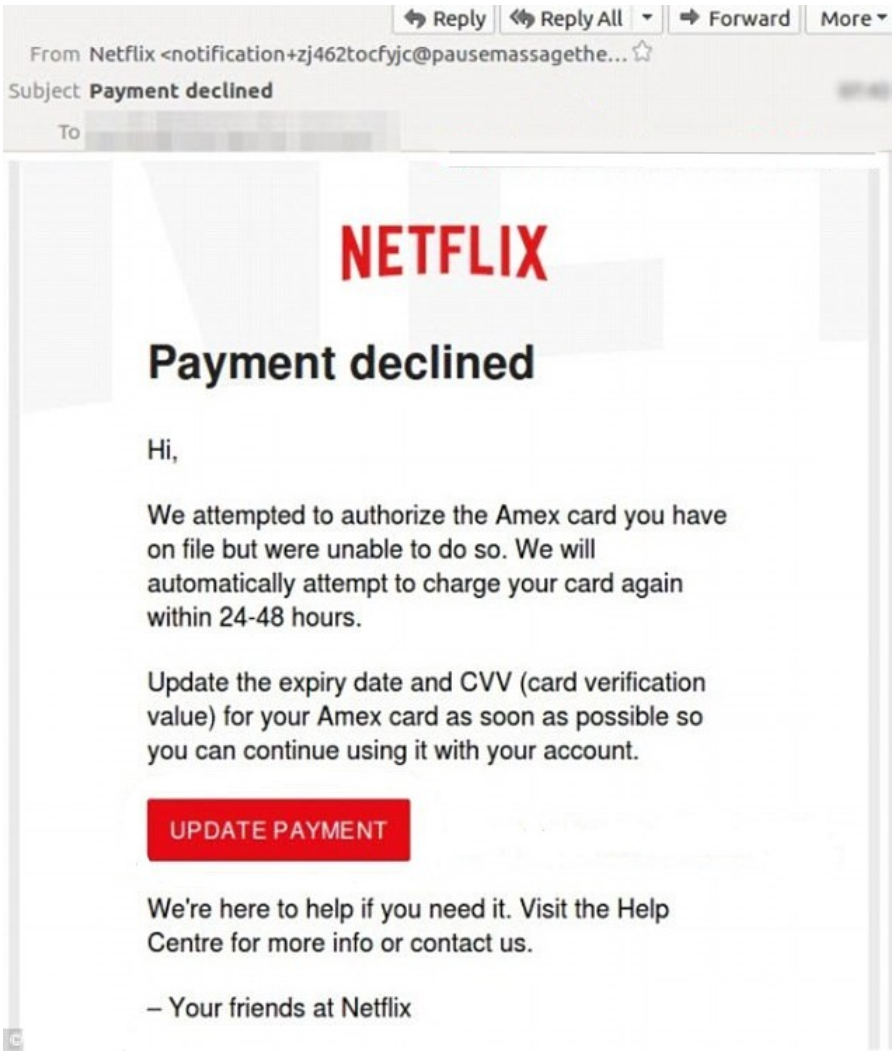

Figure A.1: Presented interview phishing image - 1

Figure A.2: Presented interview phishing image - 2

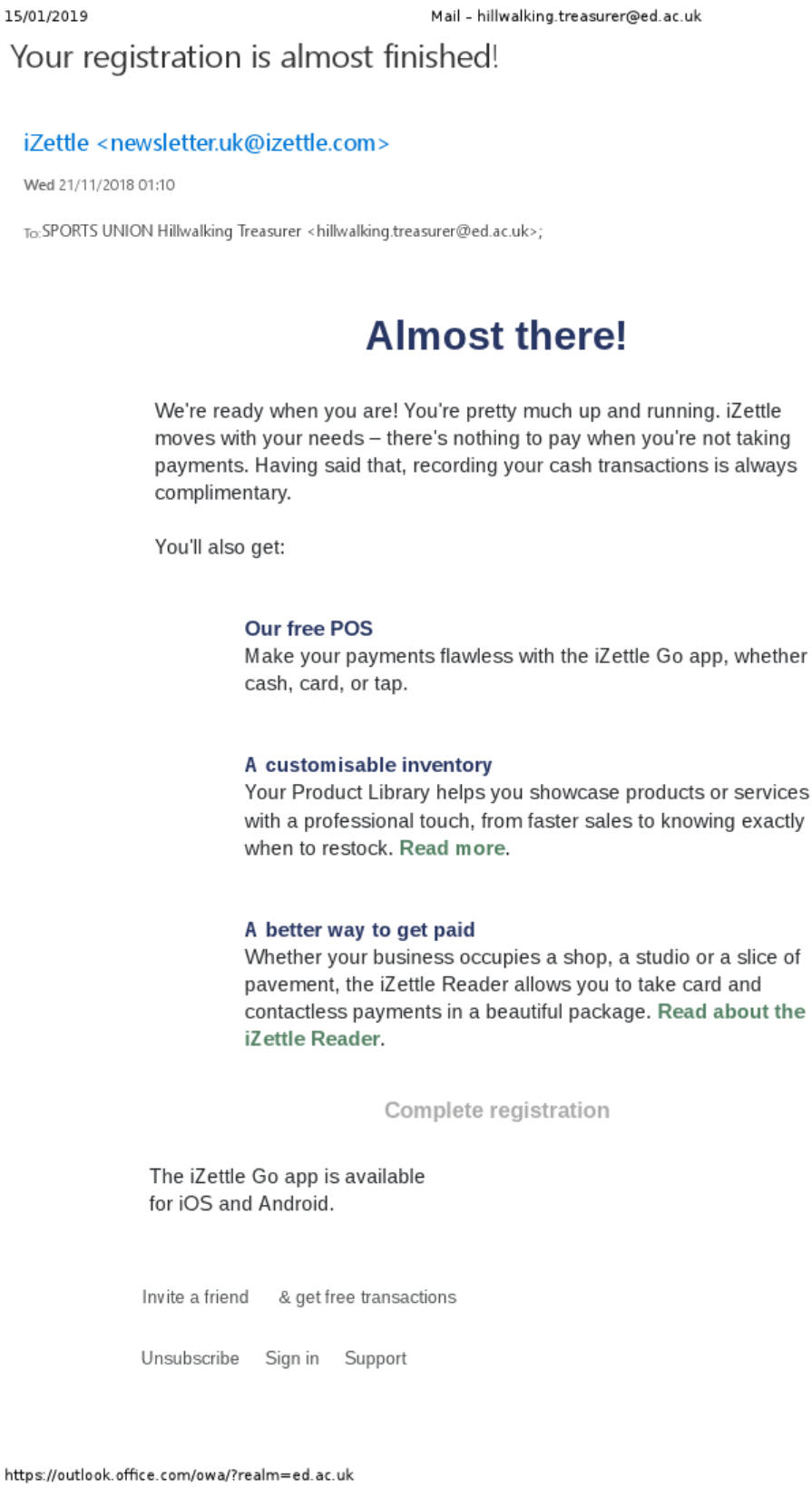

Figure A.3: Presented interview phishing image - 3

1 myetherwallêt.com
2 myetherwallet.com
3 myetherwallet.com
4 myetherwallėt.com
5 myetherwallet.com
6 myetherwallët.com
7 myetherwallet.com
8 myetherwallét.com
9 myetherwallet.com
10 myetherwallet.com
11 myetherwället.com
12 myetherwállet.com
13 myetherwałlet.com
14 myetherwałlet.com
15 myetherwallęt.com
16 myetherwalleţ.com
17 myetherwalleţ.com

Figure A.4: Presented URL image - 1

http://bit.ly/1cY78R

http://www.facebook.net

https://www.lovemoney.com/news/74356/ryanair-free-tickets-flights-whatsapp-fake-congratulations-share

Figure A.5: Presented URL image - 2

Appendix B

Interview Results

Participant	Tech. Exp.	CS Exp_1	Phishing Img 1	Phishing Img 2	Phishing Img 3	URLs Img 1	URLs Img 2.1	URLs Img 2.2	URLs Img 2.3	Browser	Usage Likelihood
P1	5	1	Phishing	Phishing	*Non-Phishing*	10	Possibly	Malicious	Non-Malicious	Chrome	4
P2	5	3	Phishing	Phishing	*Non-Phishing*	None	Possibly	Malicious	Non-Malicious	Chrome	4
P3	5	2	Phishing	Phishing	*Non-Phishing*	2	Possibly	Non-Malicious	Malicious	Chrome	2
P4	4	1	Phishing	Phishing	*Non-Phishing*	10	Malicious	Malicious	Non-Malicious	Chrome	4
P5	5	3	Phishing	Phishing	*Non-Phishing*	10	Possibly	Malicious	Malicious	Chrome	5
P6	4	2	Phishing	Phishing	*Non-Phishing*	None	Possibly	Malicious	Malicious	No Preference	4
P7	4	2	Phishing	Phishing	*Phishing*	All but 13,14	Possibly	Malicious	Malicious	Chrome	5
P8	4	1	Phishing	Phishing	*Non-Phishing*	10	Malicious	Malicious	Malicious	Chrome	3
P9	4	1	Phishing	Phishing	*Non-Phishing*	10,1,8,9,11,12	Non-Malicious	Malicious	Malicious	Firefox	5
P10	5	3	Phishing	Phishing	*Non-Phishing*	10	Malicious	Malicious	Non-Malicious	Chrome	4
P11	5	2	Phishing	Phishing	*Non-Phishing*	10	Possibly	Malicious	Non-Malicious	Firefox	5
P13	4	2	Phishing	Phishing	*Non-Phishing*	10	Malicious	Malicious	Non-Malicious	Chrome	4
P14	4	3	Phishing	Phishing	*Non-Phishing*	10	Possibly	Malicious	Non-Malicious	Safari	3
P15	5	2	Phishing	Phishing	*Non-Phishing*	None	Possibly	Malicious	Non-Malicious	No Preference	2
P16	4	1	Phishing	Phishing	*Non-Phishing*	10	Non-Malicious	Malicious	Malicious	Chrome	4
P17	4	1	Phishing	Phishing	*Non-Phishing*	10	Possibly	Malicious	Malicious	Chrome	4
Correct Answer:			n/a	*Phishing*	Non-Phishing						
Count/Most Frequent:	4	1	16	16	15	3	10	15	8	Chrome	4
Percent of Occurance:	56.25%	37.50%	100.00%	100.00%	93.75%	18.75%	62.50%	93.75%	50.00%	68.75%	50.00%

Figure B.1: Interview Quantitative Analysis

Phishing Definition		
Themes	Codes	Participants
How its achieved	trick people	P11 P13 P14 P1 P6 P8 P9
	deception used	P11, P13 P14 P2 P1 P3 P6 P7 P9 P10
Phishing Intentions	stealing credentials	P13 P3 P5 P6 P2 P10
	causes and specifies malicious actions	P14 P4
	hidden capture of data	P10
	accessing sensitive info	P15 P16 P17 P2 P1 P9
No self-declared knowledge	no idea about phishing	P4 P8
Phishing Vehicles	spam emails	P5 P15 P17
	use of website	P11 P2 P3 P10
	use of malicious link	P13 P16 P17 P10
	use of emails	P15 P16 P17 P4
	spear phishing	P5 P16

Figure B.2: Interview Thematic Analysis - Phishing Definition

Phishing Images		
Themes	Codes	Participants
Contextual Knowledge	recipient destination safe	P17 P3
	current relationship with sender	P16 P17 P5 P8 P15
	knowledge of sender	P13 P16 P17 P1 P8 P10
Security Practices	sensitive information request	P11 P4 P1 P3 P6 P8 P7 P9 P14
	strange email security request	P16 P17 P4 P8 P14
	suspicious email address	P11 P13 P16 P17 P2 P1 P3 P5 P7 P9 P15 P14 P10 P4
Analysis Strategy	email domain focus	P7 P9 P15 P14 P10 P11 P1
	concern of link destination	P11 P17 P6 P14 P10 P9 P15
	general email state	P7 P9 P10 P2
Quality of Email	non country specific references	P13 P16 P8 P10
	strange salutation	P13 P6 P7 P9
	not matching company quality	P13 P16 P17 P1 P3 P9 P14
	spelling and punctuation errors	P13 P16 P2 P6
	non matching language used	P17 P2 P1 P6 P8 P15
	threatening tone	P3
	no contact details	P9 P15
	no personal indicators	P16 P4 P6 P8 P15 P14 P10
	expect more details	P14
Further Checks	check by visiting site	P4
	check with company	P10
	cross reference emails	P10

Figure B.3: Interview Thematic Analysis - Phishing Images

URL Definition		
Themes	Codes	Participants
Limited knowledge	no self-declared knowledge of	P16 P10 P7
	confusion with web pages	P7
Potentially Malicious	might be malicious	P4 P17 P15 P8
	malicious redirection	P8
Purpose of URLs	emails are a form of URL	P7
	links to resources	P11 P3 P1 P13 P14 P15
	used on internet	P11 P16 P3 P5 P1 P2 P17 P13 P14 P6
	access web pages	P3 P11 P5 P1 P4 P2 P17 P10 P13 P14 P9 P8 P6
	used in hyperlinks	P9 P6
Reading URLs	knowledge of structure	P11 P16 P2 P10 P13 P15
	importance of domain	P16 P14 P7 P6
	relation to file path	P16
Internet Infrastructure	IP address link	P3 P1 P10 P17 P13
	Works with DNS	P3 P1 P2 P17
	registered with ICANN	P7

Figure B.4: Interview Thematic Analysis - URL Definition

URL Images		
Themes	Codes	Participants
Misconceptions	accent is safe	P3 P17 P10 P9 P7
	doesn't trust any shortened links	P8 P4
	thinks https is safe	P9 P8
Knowledge of Structure	knowledge of file path	P11
	knowledge of domain source	P11 P1 P4 P3 P13 P14 P8 P6
	understanding of top level domain	P11 P5 P2 P3 P17 P10 P15 P9 P7
Detailed Knowledge	noticed non-ASCII	P2 P15 P6
Strategies for images	looks for ascii vs unicode	P11 P8 P6
	fooled by buzzwords	P5 P16 P3 P17 P9 P7 P6
	odd one out selection	P5 P11 P16 P1 P4 P14 P8
	looks at protocol	P16 P13 P14 P7 P8 P6
	visit to find out	P1

Figure B.5: Interview Thematic Analysis - URL Images

Phishing Tools		
Themes	Codes	Participants
What information they want highlighted	see protocol of url	P14 P8
	details of URL highlighting	P13 P11 P2 P1 P16
	details of why	P14 P7 P8 P6 P5 P17 P3
	highlighting undetectable indicators	P5 P10
How they want information to be highlighted	passively warn the user	P8 P7 P11
	reputation based visual system	P11 P13 P17 P3 P2 P4
	obvious visuals	P3
Other wishes for the tool	options to visit malicious anyways	P14 P15 P7 P3
	minimal obstrusivness	P17 P3
Compatible	compatible with email	P15 P8 P16
	can be used with social media	P5
Should occur before visiting	expand shortened links	P6
	preloaded look at website before acccesing	P13 P16
	details before visit	P1 P2 P13 P16
	block from visit malicious site	P14 P15 P9 P10 P17
Tool Scope	focus on high risk sites	P14
	automatic detection	P7 P10 P1
	work for all links	P6 P5 P17 P1 P7
	identify malicious sites	P9 P14 P4 P7 P3 P11 P16

Figure B.6: Interview Thematic Analysis - Features in the Phishing Tool

General Info Presentation		
Themes	Codes	Participants
Information they want to see	URL learning page	P10 P1 P16
	highlighted protocol	P15 P1 P6
Means of presenting info	traffic light reputations	P15 P14 P4 P2 P3 P13 P11 P7 P8 P6
	link annotation of status	P14 P16 P4 P2 P7 P8
	alert popup	P16 P9
	overall count of malicious links	P11
	mouse over link info	P15 P16 P17 P7
Intervention Feature	intervention only when malicious	P14 P2 P17 P8
	wants intervention page	P4 P14 P2 P3 P17 P10 P5 P11 P6 P7 P8
	intervention should be unmissable	P11 P7
Clarity of explanations	detailed explanations	P2 P13 P7 P8
	wants a clear explanation page	P13 P7
	wants easily accesible info	P13 P5

Figure B.7: Interview Thematic Analysis - General Phishing tool UI Features

Training or Intervention Presentation		
Themes	Codes	Participants
Means of Interaction	would only look at status shields	P16 P11
	blocked after clicking malicous URLs	P14 P1 P15 P2 P3 P5 P11 P6 P9 P8
Bonus Features	Options to toggle link annnotations off	P10 P4 P6
Important Features	limited obtrusiveness	P16 P13 P6 P8
	wants whitelist	P4 P13 P10 P5 P11 P8
	doesn't want access to malicious website	P13 P9
Information presentation	brief info	P17
	details why blocked	P14 P1 P15 P2 P3 P17 P5 P11 P6 P7 P8
	alerts instead of page	P17 P9

Figure B.8: Interview Thematic Analysis - Training or Intervention Presentation

Help Information		
Themes	Codes	Participants
Guides they would like	display guide on first install	P14 P16 P3 P17 P8
	tutorial on how to use the tool	P14 P17 P3 P16 P9
	learn page on URLs	P16 P7
Information presentation	only pictures and text	P14 P15 P10 P4 P3 P17 P5 P6 P7
	interactive tutorials	P15
	wants video	P16 P9
Unneeded Feature	intuitively know how it works	P15 P10 P8

Figure B.9: Interview Thematic Analysis - Help Presentation

Likely to Use Tool		
Themes	Codes	Participants
Enthusiasm for tool	no known similar applications	P10 P6 P11 P9
	loves idea	P5 P11
Features they really like	wants personal URL list	P11
	wants training elements	P8 P9 P4
	wants block feature	P15 P10 P2
	wants safety net	P10 P13 P6 P7 P4
	wants training elements	P8 P9 P4
Aspects they hope it has	comprehensive for URLs	P16 P13 P11 P6 P7
	wish it to be a general security	P16 P17
	wants low false positives	P2 P3
	want a high accuracy	P14 P2
Reasons not to use	other alternatives	P15
No personal use for tool	already have enough knowledge	P1 P3 P10 P13
	wants for other non technical people	P17 P10 P13 P6

Figure B.10: Interview Thematic Analysis - Likelihood to use

Appendix C

Additional Paper Designs

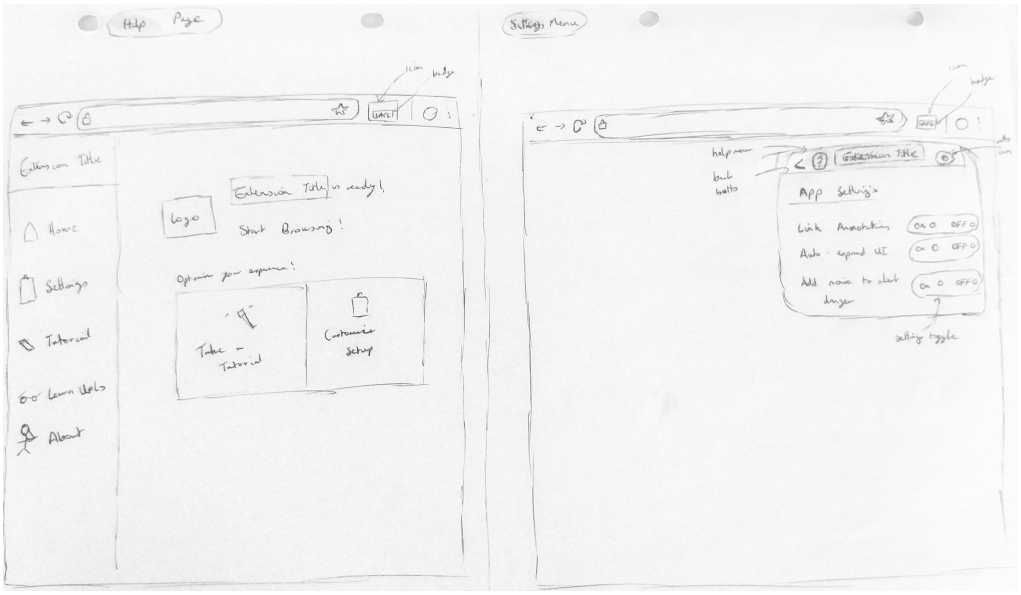

Figure C.1: Help and Settings pages

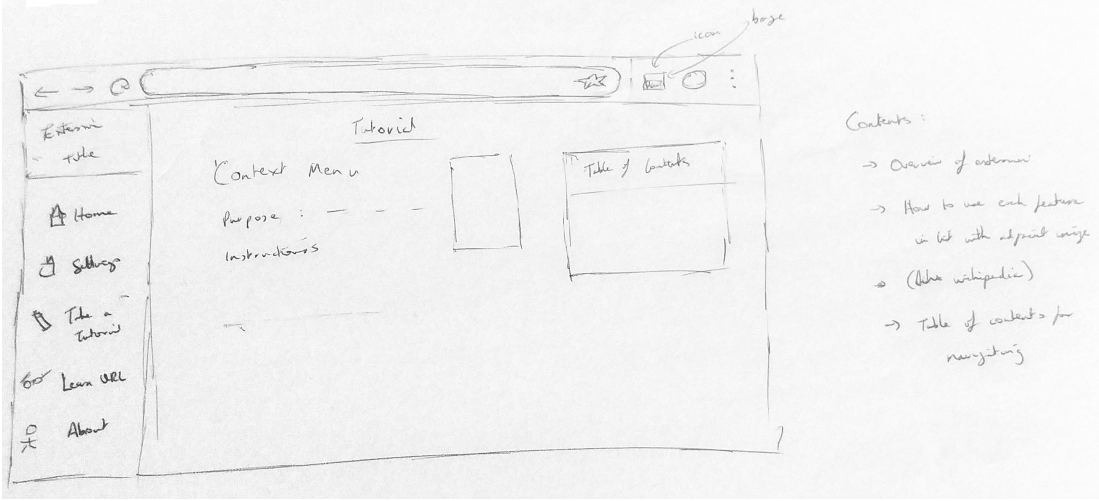

Figure C.2: Tutorial page

Appendix D

Phishing Heuristics Proposal

UNIVERSITY OF EDINBURGH

Design Proposal for a User-facing Phishing Learning and Detection Tool

Stephen Waddell

March 28, 2019

CONTENTS

1 Introduction **3**
- 1.1 Goals of the tool . . . 3
- 1.2 Proposal Outline . . . 3

2 Overall Algorithm **5**
- 2.1 Decision Making Algorithm . . . 5
- 2.2 Limitations . . . 6

3 Safety Metrics **7**
- 3.1 Metrics . . . 7
- 3.2 Approach . . . 7

4 URL Parsing Features **8**
- 4.1 Manipulation Tricks . . . 8

5 Domain Features **12**

6 Page Features **13**

7 Data Sources **15**
- 7.1 Blacklists . . . 15
- 7.2 Reputation Ranking . . . 15
- 7.3 URL Information . . . 15
  - 7.3.1 Required regex . . . 16
- 7.4 Domain Information . . . 16
  - 7.4.1 Top-level domains . . . 16
  - 7.4.2 WHOIS Database . . . 16
- 7.5 Page Information . . . 16
  - 7.5.1 Certificate Validation . . . 17

1 INTRODUCTION

In developing a Phishing and Learning Detection tool to be deployed onto the Google Chrome platform as an extension, I have prepared an overview proposal of the back-end functionality I believe will be needed as part of this. This proposal covers a number of different points about the functionality required to classify a URL into a a respective safety category.

1.1 GOALS OF THE TOOL

The primary design goals of this tool are as follows:

Goals	**Justification**
Classifying every URL: those present in any given page, the page URL itself and any the user is directed to outside the browser	This intended to be a comprehensive solution to phishing and other security exploits which employ malicious URLs
Using the browser to prevent the user visiting any URLs that have been classified as malicious without user invention.	This intended to cut down unintentional direction to malicious URLs, caused when on platforms such as email
Presenting the information on each URL to the user in an understandable way, at appropriate points of intervention	To encourage the user to learn through embedded training: allowing users to catch URLs themselves and reduce their own danger
That the end built system includes best practice of software engineering: maximising efficiency and accessibility	That the system works effectively as and when required, such that users are not do not remove it due to efficiency concerns

Table 1.1: Justification of goals

1.2 PROPOSAL OUTLINE

To achieve these design goals, this proposal outlines a heuristic-based algorithm for classifying a URL into a one of three states: Safe (High likelihood of safety), Warn (Possible cause for concern) and Alert (High risk of danger with URL). Each of these form the basis of a traffic light system (Green, Amber and Red respectively) which informs the user about the relative safety of each URL. In this proposal document, each of these states will be referred to by their respective colour.

Additionally, the proposal also covers the a breakdown of each possible issue with a URL. For each of these, an explanation of the issue and a proposed method is outlined to deal with that issue. The proposal also considers how the data will be sourced to solve each possible

issue.

The front-end of this tool, which is not outlined in this proposal, includes the following features:

- active intervention when a user clicks a red URL (in the form of a site warning middle man)
- annotating links on each site with a badge representing the safety colour of each URL
- a breakdown of each URL using the a research based URL report[1]

The classification of each URL is used to create the appropriate User Interface display for each URL. One of the secondary goals of the tool is to achieve as low a false positive rate as possible to maintain user engagement with the tool.

Feedback on the plans outlined in this report would be much appreciated.

2 Overall Algorithm

2.1 Decision Making Algorithm

The algorithms input will be the given URL and all required information needed to consider the classification of the URL.

The algorithm begins by considering each case where a URL might be considered purely green (undoubtedly safe). If any of these indicators, such as inclusion on a reputable whitelist, are true then the URL is classified as safe and the algorithm returns that it is a green state. Otherwise the algorithm will analyse the URL and classify the given URL as having either a yellow or red state.

To achieve this, the algorithm analyses each set of features using the metrics outlined in the subsequent chapters. Each set of features is a useful indicator of a URL's safety, which can be used to classify how likely a given URL might be phishing. These indicators are referred to as issues in this proposal, and each of these issues is categorised into one of three groups: these being known, possible and no issue (which each represent a decreasing priority level).

The features that are the focus of this proposal are:

- URL parsing features
- Domain features
- Page features

Content Features could be used in addition to this but are not favoured in this proposal due to the associated high risk of parsing the content of malicious sites.
To classify the URL a count is kept of each issue that arises from the data parsing. This is matched with threshold values to classify the URL.

Known Issues	Possible Issues	Output
>=1	>=0	Red
0	>=5	Red
0	<5	Warn
0	0	Green

Table 2.1: Thresholds for Output

The resulting output is then feed back into the system for use by the User Interface. For each result, the state of the URL is returned along with the information used to calculate that state. This information is displayed to the user in the breakdown of the URL. For those in a green state, the reason indicating its safety is returned and if it is red or yellow, the issues included in the URL are also returned.

2.2 LIMITATIONS

This approach favours limiting the number of URL's classified as yellow as much as possible. This is because one of the major benefits of the tool is the tool's ability to actively intervene to prevent the user from visiting a malicious site. This only occurs for URLs classified as the red state. At this point users benefit from embedded training as they have the break-down of the URL information displayed to them, in a real-time scenario.

To limit the number of URLs in the yellow state, the algorithm has lower thresholds for classifying the URLs as a red state. One of the key limitations of this approach is that it potentially creates a higher false positive rate for the user. Active intervention with a high false positive rate has been shown to have a reduced effectiveness over-time[3].

To resolve this problem, the intention is to implement a personal whitelist for users to maintain. Users will be able to update this whitelist with URLs when they are presented as part of this tool. This is a constant element of the UI and is particularly useful for false positive URLs classifies as a red state. After the user whitelist a site it will not be included in the red state, instead it will be given a amber state (where active intervention does not occur but the user will still be made aware they are not entirely safe). This should reduce the false positive rate over time. In user trials, the ideal outcome would be that the tool tailors itself to users common sites over time, only showing sites that are concerning and the user has never visited before. This would therefore reduce the false positive rate over time.

Known Issues	Possible Issues	Personal Whitelist	Output
>=1	>=0	False	Red
>=1	>=0	True	Warn
0	>=5	False	Red
0	>=5	True	Warn
0	<5	False	Warn
0	0	False	Green

Table 2.2: Thresholds for Output

Another suggestion to improve the false positive rate, is that the users could be allowed to toggle the threshold for possible issues being classified in the red state. In either case, to ensure the needless use of the blacklist, particularly in cases where the URL has been blacklisted, the user will be asked for additional approval. In the User Interface this may involve presenting an alert pop-up box being displayed asking them if they are sure they wish to continue.

3 Safety Metrics

It is very hard to come up with a full proof set of safety metrics. Comparatively it is easier to verifiably say a site is malicious rather than safe. There are however some indicators that are useful to suggest a site is safe.

In the proposed algorithm, there are currently two ways that a site can be designated as safe. The first is that the site has no known or possible issues: this mean it has not bee n flagged by any of the heuristics. The second is that it matches against one or more safety metrics.

3.1 Metrics

The first metric is a URL's inclusion in the list of Alexa Top sites (7.2) and is not known to be a hosting service for other sites. The Alexa Top sites is a strong indicator of the most popular sites and popularity is an indicator of safety. This is because users do not regularly visit or return to sites known to be malicious. However, some of the most popular internet sites are known to host other sites such as *wordpress.com*. Wordpress, for instance, can be used to host other sites which contain malicious content so classifying these sites as safe would be inaccurate. Therefore the Alexa Top Sites results will be filtered using known content hosting sites and other potentially malicious sites by checking their occurrences on blacklists such as hpHosts (7.1).

An additional metric is the PageRank of the website. This is based on backlinks, which are incoming links to a webpage. By taking a measure of how many quality backlinks a webpage has it indicates how popular that webpage is. This is a metric which has been used in major search engine providers in the past to indicate how high a web page should be listed in its search results.

Looking at how often the site is shared on social media is another useful metric. Sites such as Facebook and Twitter regularly include links to external sites which are shared between its users. This is an indicator of a sites popularity, as multiple shares between users suggests it is safe since they are encouraging others to visit it.

3.2 Approach

The idea is to combine each of these metrics and use appropriate thresholds to determine if a site is popular enough to be classified as safe. By applying these metrics to each website, and ensuring each page's URL is safe in the site, we can mark sites containing multiple webpages as safe without analysing each URL, provided they do not leave that domain. The idea behind this is to increase efficiency in each site. This should not pose many problems providing the metrics are applied correctly, as there is a much smaller chance of being attacked on a site when swapping between pages from distinct sites [insert reference].

4 URL PARSING FEATURES

This focuses on parsing the URL into its distinct components and use these to pick out indicators in the URL itself which might suggest it is malicious.

Processing each of these heuristics typically involves using natural language tools such as regex to match the contents of each URL against any concerning flags. These often involve the request of some data in order to to complete each check accurately; the limited amount of required libraries means these checks can be preformed locally on a users machine. Each metric relates to a issue level based on how much of an indicator that metric is.

4.1 MANIPULATION TRICKS

Trick	Explanation	Check	Data Needed	Issue
Too many subdomains	Can redirect user to alternative site	regex search all '.' characters and take a count of resulting array	Amount of common URL domains; The URL domain	Possible
Typosquatting Popular Domains	Targets users who incorrectly type a web address into their browser	Similarity measure between host name and popular domains	Appropriate similarity measure (7.3); list of most popular domains	Known
Unusual top level domain (Camouflage)	Top level domain is not one of the most commonly used ones	Match top level domain with a list of most popular domains	List of most popular top level domains	Possible
Digit replacement of letters	Masking the direction of the URL by replacing characters such as 'o' with similar digits such as '0'	Use regex to count the amount of digits in the hostname, and compare to the average or typical	Typical amount of numbers in a URL hostname	Possible

Table 4.1: Manipulation Tricks

Trick	Explanation	Check	Data Needed	Issue
Shortened URLs	URLs which have been shortened means users are unable to pick out key characteristics from a URL	Use the shortener services to follow or expand the URL and express this to user; express to the user and run analytics on expanded URL (if the shortened URL is safe)	Access to shortener APIs	Possible
UTF8 encoding substitution	Substituting identical looking characters from different alphabets such as English and Cyrillic	Use a language detector to check if the detected URL language matches with the user's local language	Language detector; Access to default local language of browser	Known
Use of I.P. address, hex or decimals	Use of non-standard information makes the URL harder to understand and can be used to confuse the user	Use specific regex for checking the hostname for each of these elements	IP address regex (7.3.1); Hex regex (7.3.1); Decimal Regex (7.3.1)	Known
Substituting normal characters	Characters such as 'w' can be replaced with 'vv'	Check the work similarity with any suggested search engine search replacement	API to do search engine search (7.5) and/or get replacement text	Known
Mislead	Expected company name is embedded somewhere in the URL but not destination	Search the URL for a list of the most popular company names	List of most popular company names	Known

Table 4.2: Manipulation Tricks

Trick	Explanation	Check	Data Needed	Issue
Embedded URL in query string	Open redirection is detected based on the existence of a URL in the query string	Use regex to match URL any possible the query string	Access to URL query string; URL match regex	Possible
Number of http/https	Some attackers add additional http to trick the user about the start of the URL	Search the URL using regex to get a count of each occurrence	Regex to match protocols across the URL	Possible
Suggestive word tokens	Some words only exist in phishing URLs such as "confirm", "banking", "account", and "signin"	Search using regex for any words in a list of known suggestive word tokens	List of suggestive word tokens	Known
Suspicious characters in the URL	The existence of 'at' (@) means criminals use this to mislead users or characters such as hyphen (-)	Use of regex to check for any suspicious characters in the URL	list of suspicious characters and their occurrences [4]; regex to check whole URL	Known
Encryption	HTTPS connection is more secure. Incompetent feature but useful for safety	Check the protocol is https or http through string comparison	access to the URL protocol;	Known
Non-standard port	Whether the port belongs to a standard HTTP ports: 80, 8080, 21, 4143, 70 and 1080	Match the port type to a given port in the URL and check the protocol and the port match	list of common ports; access the URL port; access the URL protocol	Known

Table 4.3: Manipulation Tricks

Trick	Explanation	Check	Data Needed	Issue
Use of atypical deliminator character	'.' is the typical deliminator character, example is a 'home-depot.com' rather than 'homede-pot.com'	Search for deliminator characters other than '.' using regex	list of deliminator characters; regex to search for these	Possible
Unusually long URL hostname	URL hostnames that are overly long can be used to mask the true destination of the URL	Count the amount of characters in the URL, and compare to the typical URL length	Typical length of a URL hostname[2]	Possible
Different Top Level Domain (TLD)	The URL has the same domain as a popular site but a different top level domain	Compare both the popular TLD with the URL	Popular domain list; hostname of popular domain; top level domain of each	Known
TLD out of position	The appearance of a popular TLD such as "com" in the subdomain deludes users into believing this is the end of the hostname.	Check if the URL subdomain includes any popular subdomains using regex	List of popular TLDs; regex to match TLDs in subdomain; access to the URL subdomain	Known

Table 4.4: Manipulation Tricks

5 Domain Features

Domain features focus on detecting phishing domain names, such as by checking the details of the domain's registration status. In doing passive queries related to the domain name, we can detect indicators based on the known trends of malicious URLs. To handle each of these indicators, external data using API's must be requested to reason what the status of a given URL is.

Fact	Explanation	Check	Data Needed	Issue
Domain/I.P. Blacklisted	If info is blacklisted it is unlikely to be safe	Store blacklist as part of local database: query blacklist database tables for URL presence	Stored blacklists; Chrome Database APIs	Known
Days of domain registration	Phishing sites tend to be newly created	Store WHOIS information as a database: query the database to get the creation data of the website	Current date; Threshold amount of days to check URL uptime; Database or API with WHOIS Info	Possible
Registrant name hidden	This can indicate the individual doe not wish to be found and a crime is afoot	Query the WHOIS database for the registrant name: compare the registrant name to check if it has been hidden	Get an idea of what a hidden registrant looks like	Possible
Domain match	The domain exists in the WHOIS record or the domain is in the URL matches the domain in WHOIS	Check if the domain exists in the WHOIS database by querying it	WHOIS database API	Known

Table 5.1: Domain Facts

6 Page Features

Page features use information about pages which are calculated using reputation ranking services. These are some of the most useful indicators of a URL's safety if they show a site is popular, but can equally be an indicator of phishing where a site is shown to be unpopular. In this sense, they give information about how reliable a site is.

Each of these heuristics also requires the use of a number of external data sources to be able to accurately function as a heuristic.

Fact	Explanation	Check	Data Needed	Issue
Search Results	Phishing website have short life therefore, usually they are not in the result, besides, not frequent in the results	Use an api to get the first x(20) [reference needed to indicate amount] and check if the result is there	API to do a search from command line; A means to parse search results	Known
Number of redirections	Phishing sites tend to be newly created	Use an api to check the amount of redirects in a given URL	API to do this redirect search; Build understanding of what output will look like	Known
Location	The physical location of the registrant usually differs from the physical ones	Use a web service to look-up the location of the corresponding service to the website; Check this location with locations that are known to host a high amount of malicious contents	List of places that are known to host more malicious content; Loo-up web service	Possible

Table 6.1: Page Facts

Fact	Explanation	Check	Data Needed	Issue
Global Popularity	The rank that Alexa assigns to domains	Get the global popularity of the site of using the Alexa Api; Check threshold to see if this is low	Alexa Global popularity api	Known
Page Rank	The relative importance of a page within other web pages	Use page rank api; Check if the page rank is high or above threshold	PageRank api	Possible
Social Reputation	The link popularity among social media users such as Twitter and Facebook	Use social reputation api; Check if social reputation is low	Social reputation api	Possible

Table 6.2: Page Facts

7 Data Sources

There are two specific goals when it comes to requesting data from various sources. The first is security and privacy: each API transmission of a plain text URL, could expose their pattern of website behaviour to potential malicious actors. Therefore one goal in collating of requisite data, is to limit this by focusing on data which can be collected and stored internally within the tool. The second goal is efficiency: if the time taken to consider each URL is too great then the response time of the tool might be too slow for the user. This can be handled in multiple ways such as asynchronous URL prepossessing. However, requesting, where possible, data such as blacklists in bulk and storing them locally in a tool accessible database, is a means to handle this from a data perspective. The advantage of this is to prevent outside awareness of user requests. This also allows the developers to store visited user URLs more securely i.e. hashed and salted.

Alongside this, using multiple sources of data is also key to prevent an over-reliance on any one particular data source and a corruption of the tool's accuracy. This could lead to information being skewed, in turn compromising user safety.

7.1 Blacklists

Google Transparency Report:
https://transparencyreport.google.com/safe-browsing/search?hl=en$_GB$
$Database - D/L$: *https://developers.google.com/safe-browsing/v4/*

PhishTank:
https://www.phishtank.com/

hpHosts:
https://hosts-file.net/
$Database - D/L$: *https://hosts-file.net/?s=Download*

7.2 Reputation Ranking

Alexa:
https://www.alexa.com/topsites

The Moz Top 500:
https://moz.com/top500

7.3 URL Information

Levenshtein Distance:
https://en.wikipedia.org/wiki/Levenshtein$_Distance$

7.3.1 Required regex

***Hex regex**:*
https://stackoverflow.com/questions/9221362/regular-expression-for-a-hexadecimal-number

***Decimal regex**:*
https://stackoverflow.com/questions/11500482/regex-to-find-integers-and-decimals-in-string

***I.P. address regex**:*
https://www.regular-expressions.info/ip.html

7.4 Domain Information

7.4.1 Top-level domains

***W3Techs**:*
https://w3techs.com/technologies/overview/top$_{l}evel_{d}omain/all$

Lifewire:
$https://www.lifewire.com/most-common-tlds-internet-domain-extensions-817511$

7.4.2 WHOIS Database

***WHOIS Database API**:*
https://www.whoisxmlapi.com/

7.5 Page Information

***Search Engine Spell-check**:*
https://azure.microsoft.com/en-gb/services/cognitive-services/spell-check/

***Google search command line**:*
https://www.npmjs.com/package/node-googler

***URL Redirect Checker**:*
https://httpstatus.io/help

***Check Host Net**:*
https://check-host.net/ip-info?host=www.google.com

7.5.1 Certificate Validation

***Certificate Info**:*
https://chrome.google.com/webstore/detail/certificate-info/jhldepncoippkjgjkmambfglddmjdmaj
The API behind this extension is possible to use. It only uses basic certificate validation however.

REFERENCES

[1] *K. Althobaiti and K. Vaniea. Is this url safe to click on? supporting users' comprehension of phishing features. "[Unpublished]", 2018.*

[2] *S. Garera, N. Provos, M. Chew, and A. D. Rubin. A framework for detection and measurement of phishing attacks. In* Proceedings of the 2007 ACM Workshop on Recurring Malcode, *WORM '07, pages 1–8, New York, NY, USA, 2007. ACM.*

[3] *M. Wu, R. C. Miller, and S. L. Garfinkel. Do security toolbars actually prevent phishing attacks? In* Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, *CHI '06, pages 601–610, New York, NY, USA, 2006. ACM.*

[4] *G. Xiang, J. Hong, C. P. Rose, and L. Cranor. Cantina+: A feature-rich machine learning framework for detecting phishing web sites.* ACM Trans. Inf. Syst. Secur., *14(2):21:1–21:28, Sept. 2011.*

Appendix E

System Usability Scale Survey

Feedback Survey

		Strongly disagree				Strongly agree
1.	I think that the target audience would like to use this system	1	2	3	4	5
2.	I found the system unnecessarily complex	1	2	3	4	5
3.	I thought the system was easy to use	1	2	3	4	5
4.	I think that I would need the support of a technical person to be able to use this system	1	2	3	4	5
5.	I found the various functions in this system were well integrated	1	2	3	4	5
6.	I thought there was too much inconsistency in this system	1	2	3	4	5
7.	I would imagine that most people would learn to use this system very quickly	1	2	3	4	5
8.	I found the system very cumbersome to use	1	2	3	4	5
9.	I felt very confident using the system	1	2	3	4	5
10.	I needed to learn a lot of things before I could get going with this system	1	2	3	4	5

What is your gender?

☐ Male
☐ Female
☐ Prefer not to answer
☐ Other .....

What year are you?

☐ UG1
☐ UG2
☐ UG3
☐ UG4 / MInf
☐ MSc
☐ Other .....

Appendix F

Think Aloud Materials

‘Phishing Detection and Learning Tool’ Consent Form

This project aims to gather information about the usability of a developed Phishing Learning and Detection tool through user evaluation.

Today we will be doing a think aloud study to understand your thoughts on the usability of the given tool. During this study, I will be asking you to interact with both the tool and an example webpage, and talk about how you interact with them. At the end of the think alouds, there will be an opportunity to discuss your general thoughts and opinions about the tool based on your experiences with it.

We will be audio recording during the study. If you feel uncomfortable about this at any time, you may stop the recording or tell us that the next bit should not be quoted. There will not be any compensation for participation in this study beyond the knowledge that you will have advanced science.

This study will be used to gather usability information about the tool, this information will be later used in as part of my dissertation. The audio and transcripts will be kept for a maximum of one year and then destroyed. Anonymized quotes or short audio clips may be retained longer for use by future students on this project.

The project is supervised by Dr Kami Vaniea (kvaniea@inf.ed.ac.uk) and conducted by myself, Stephen Waddell (s1346249@sms.ed.ac.uk).

[ ] I understand that I am participating in a study as part of the “Evaluate the usability of a security or privacy tool” project.

[ ] I am willing for the audio to be digitally recorded and transcribed as part of this research project

[ ] The researcher may use **audio/ literal quotes** from the study in future works provided that the quotes are anonymised and cannot be connected back to me.

Participant: _____ Date: _____
Researcher: Stephen Waddell_____ Date: _____

Phishing Think Aloud

Hello my name is Stephen. Today you will be interacting with a phishing learning and detection tool that I have developed as a chrome browser extension, working on an example webpage. Thank you very much for signing the consent form. Try your best to complete all the tasks, but your participation today is purely voluntary, you may stop at any time. The purpose of this tool is to detect malicious phishing and protect and inform users of this. Please remember we are testing this tool's capabilities, not your own ability.

In this observation, we are interested in what you talk about as you perform the tasks we are asking you to do. In order to do this, I am going to ask you to talk aloud as you work on the tasks. What I mean by "talk aloud" is that I want you to tell me everything you are thinking from the first time you see the statement of the task till you finish the task. I would like you to talk aloud constantly from the time I give you the task till you have completed it. I do not want you to try and plan out what you say or try to explain to me what you are saying. Just act as if you were alone, speaking to yourself. It is most important that you keep talking. If you are silent for a long period of time, I will ask you to talk. Do you understand what I want you to do?

Wait for a response

Good. Now we will begin with some practice problems. First, I will demonstrate by talking aloud while I solve a simple problem: "How many windows are there in my house?"

[Demonstrate thinking aloud]

Wait for response

Now it is your turn. Please talk aloud as you multiply 120 * 8.

[Let them finish]

Good. Now, those problems were solved all in our heads. However, when you are working on the computer you will also be looking for things and seeing things that catch your attention. These things that you are searching for and things that you see are as important for our observation as thoughts you are thinking from memory. So please verbalize these too. As you are doing the tasks, I won't be able to answer any questions. But if you do have questions, go ahead and ask them anyway so I can learn more about what kinds of questions the tool brings up that it hasn't explained. I will answer any questions after the session. Also, if you forget to talk aloud, I'll say, "please keep talking." Do you have any questions about the talk aloud?

[See if they ask a question]

Now I have some tasks printed out for you. I am going to go over them with you and see if you have any questions before we start.

[Hand them the tasks.]

Here are the tasks you will be working on. Please read them aloud so you can get comfortable with speaking your thoughts. Do you have any questions about the tasks?

[See if they ask a question]

You may begin.

Tasks

Task 1: Find a malicious url on the page, and proceed to the page after viewing the details of the url

Task 2: Find the settings page and turn the annotate link feature off

Task 3: Use the tool help menu to read more about the tool's web redirection features

Task 4: Analyse the details of a url on the webpage

Task 5: Add a url to your whitelist

Task 6: Analyse any url on the page and report it

Task 7: Use the tool to report an issue with the webpage

Thank you for completing the Think Aloud's. I now have some general questions about your thoughts on the ease of use of the tool.

Questions

How easy did you find the tool to use?

- Not at all easy
- Not very easy
- Somewhat easy
- Fairly easy
- Extremely easy

Where there any features of the tool that you thought were particularly easy to use?

Where there any features of the tool that you thought were particularly difficult to use?

Is there anything about the tool that you think could be improved?

How likely would you be to use the tool in the future?

- Not at all likely
- Not very likely
- Somewhat likely
- Fairly Likely
- Extremely Likely

For what reason, did you select *insert user's answer*?

Thank you very much for your participation. It is very much appreciated. Please feel free to take a cookie.

Appendix G

Think Aloud Data Feature Codes

Open Codes	Themes	
More user confirmation; would like bigger badges to see better; make size of urls bigger and more clear; whitelist tooltip should be changed to feature name; prefer less tutorial content details; useful to get count of malicious urls on page;	Suggested Improvements	Potential Improvements
Report issue is complex; not able to report url as expected on alert url; reported url on page rather than issue; unsure how to report whole site; not able to report url as expected on alert url; confused about difference between report a url and an issue;	Report issues	
Thinks whitelist tooltip might be confusing for others; confused by whitelist location; thought the whitelist icon was confusing; doesn't know what a whitelist is; whitelist feature is clear; not able to find whitelist feature easily;	Whitelist comments	
Right-click inspection is not a clear feature; confused by help icon; confused about main popup site purpose; difficult finding a malicious link on the page;	Difficulties with tool	
Url analysis is clear; clear warning page; likes ability to undo action; likes tutorial more info link; likes link annotation; likes design of popup; likes ability to only see details for anchor tag elements; likes web redirection feature; likes reporting feature; likes the demo site; likes help icon; whitelist feature is useful;	Features people like	UI Positives
Understand popup explanation; understands the badge system; understands top bar url relativity;	Well understood features	
Easy to navigate tutorial; easy to find settings; easy to navigate through tool; easy to find features; happy with report feature location;	Navigating ease of use	
Satisfied with report issue layout; likes location of report and whitelist; too much content in help;	Tool layout	
Impressed by quality; very user friendly; fast to use after learning how to; right-click on url very easy when found;	User impressions	General Impressions
Thinks tool audience is low technical	Confusion of target audience	
Right-click on url; clicks on all the urls; goes to page to view main popup rather than right-click; does not check further details; does check further details; trys to click on badge for details;	How people are using tool	

Appendix H

Think Aloud Qualitative Data

Think Aloud Participant Information/Statistics

Participant	Date	Easy to Use	Likely to Use			
		{1,5}	*{1,5}*			
1	08/03/2019	4	5			
2	08/03/2019	5	3		Average Ease of Use	**4**
3	08/03/2019	5	5		Average Likely to Use	**4**
4	08/03/2019	4	3			
5	08/03/2019	5	5			
6	08/03/2019	5	5			
7	08/03/2019	4	3			
8	08/03/2019	3	4			