



UNDERSTANDING INFORMATION LEAKAGE IN E-MAIL HEADERS

Joshua Clark

Fourth Year Project Report for the Final Honour
School of Computer Science

May 2016

Abstract

After extensive public education, fewer people are now clicking on links in e-mails that are disguised as phishing attacks, though the threat still remains, and a considerable amount of work has gone into exploring the demographics most likely to be targeted. As the number of technically literate people grows, this sort of attack is less frequently successful. Therefore, malicious entities are more likely to attempt to attack people based on the information leaked in their emails, and more specifically, the header, which most people will not have a degree of control over.

The risks are not just limited to individual users, and at a corporate level, the risks posed by leaking information through e-mails could be even greater: e-mail headers can reveal the internal network structure of a company's computer systems as well as the different pieces of software that are running inside the system. Extracting the social information could be of great value for executing a phishing attack, however, there is also value in determining the specific weaknesses in a system. This can be aided through the use of vulnerability databases.

This report discusses the existing research into the information leaked by e-mail headers and presents a tool to extract such information. The tool's design and implementation is discussed, followed by the results of its testing.

Acknowledgements

I want to thank my supervisor, Dr Jason R.C. Nurse for his assistance throughout the year; my tutor, Professor Peter Jeavons, for his unfailing help and support throughout my time at Oxford. I would like to thank the residents and chaplains of the Oxford University Catholic Chaplaincy. Finally, I would like to acknowledge the support from my family and partner, Agata Borkowska, for encouraging me.

Contents

1. Introduction	1
1.1. Motivation	1
1.2. Aims and Objectives	1
1.3. Structure	1
2. Literature Review	3
2.1. General Data Leakage	3
2.1.1. Personal Data Leakage	3
2.1.2. Corporate Data Leakage	3
2.2. Data Leakage from E-Mails	4
2.2.1. E-Mail Headers	4
2.2.2. Example Header and Pertinent Data	4
2.2.3. Existing Research	6
2.3. Existing Tools	6
2.3.1. Google	7
2.3.2. Microsoft	7
2.4. Vulnerabilities	7
2.4.1. CVE Contents	8
2.4.2. MITRE CVE Lookup	9
2.4.3. Norton Vulnerability Protection	10
2.5. Summary	10
3. An Approach to Support Understanding of Data Leakage in E-Mail Headers	13
3.1. Overview	13
3.2. Program Specification	13
3.3. Program Overview	13
3.3.1. Parsing	14
3.3.2. Analysis	14
3.3.3. Visualisation	14
3.4. Comparisons to Existing Software	15
3.5. Typical Use	15
4. Implementation	17
4.1. Overview	17
4.2. Definitions	17
4.2.1. Parsing	17
4.2.2. Database Queries	19
4.2.3. Data Structures	19
4.3. Data Extraction and Parsing	20
4.3.1. Received fields	20

4.3.2.	Other fields	21
4.3.3.	Input Data Structures	21
4.3.4.	Output Data Structures	21
4.4.	Analysis	21
4.4.1.	Text-Based	22
4.4.2.	Client Inference	22
4.4.3.	Database Queries	24
4.4.4.	Analysis Modules and Data Flow	24
4.4.5.	Output Data Structures	26
4.5.	Visualising the Results	26
5.	Evaluation	29
5.1.	Methodology	29
5.2.	Sample Output	30
5.3.	Results	33
6.	Conclusions	37
6.1.	Summary	37
6.2.	Future Work	37
A.	Code Listings	41

List of Tables

1.	Format of presented data found in e-mail header	15
2.	M , N , R , F and score values for chosen headers	34

List of Figures

1.	Google Apps Toolbox E-mail header output	7
2.	Microsoft E-mail header output	8
3.	CVE Search Results	9
4.	Norton Vulnerability Protection Results	10
5.	Simplified Control Flow of Application	14
6.	Header Data Structure Format	21
7.	Information flow between analysis modules	25
8.	Found Information Data Structure Format	26
9.	Distribution of scores from e-mails	33

List of Algorithms

1.	Lookup based on a known key	22
2.	Lookup based on a key property	22
3.	Client Inference Technique	23
4.	Extracting CVE entries	24
5.	Exporting the Found Information to Visualisations	26
6.	Rendering the Visualisation	27

1. Introduction

1.1. Motivation

E-mail systems are now so integrated into our modern lives that we struggle to cope without them. E-mails ubiquity is also one of its largest weaknesses, a fact recognised very early on. The first spam email was sent in 1978, as documented by Templeton n.d. After spam came phishing, first described by Felix and Hauck 1987, with the first-real world use being against the customers of America Online, an ISP. However, this still relies on the targets providing their data for malicious purposes. One of the first e-mail viruses to spread was the Happy99 virus, which, other than propagating itself, had no effect on infected systems. Later viruses would target credit-card and banking information. However, all of these techniques rely on the malicious email being received and its contents being opened. There are fewer instances recorded, however, of the information flow being sent the other way. A more subtle attack will focus on the information being sent from a legitimate user to an attacker. It is easy enough for an individual to read an e-mail header and identify interesting elements, however, on a large scale, this quickly becomes more difficult.

1.2. Aims and Objectives

This project aims to support a better understanding of the data that may be leaked when e-mails are sent, both from a personal perspective, as well as the corporate data that is disclosed concerning network configurations and software installations.

To support this, I will develop a tool that can be used as described above to automatically extract information from e-mail headers and analyse its results to display the personal information contained within an e-mail's header, as well as information about the software configurations that may be found on a user's computer, or the servers used to send their e-mail.

The tool's objectives will be to parse, analyse and visualise the contents of any e-mail header that it is given. The parsing process should correctly and efficiently convert the plaintext of an e-mail header into an abstract representation. In the analysis section, the representation of the header should be searched to find information out about the sender, their software and device, and information for any servers that the e-mail message passes through. Finally, the visualisation produced should clearly show the information that is available about the sender of an e-mail and the path the e-mail took in order to arrive at its final destination.

1.3. Structure

Chapter 2 begins by discussing the existing research on the subject as well as existing publicly-available tools to analyse headers. I then use these as a basis to discuss features

that would be expected to appear in a header analyser looking for leaked information and vulnerabilities.

The specification and design of a program to support the understanding of the information leaked in e-mail headers is discussed in Chapter 3. The implementation's high-level structure and details will be discussed in Chapter 4, and algorithms presented in pseudo-code where necessary. A full listing will be presented at the end of this document in an appendix. The results of the analysis of the headers will be discussed in Chapter 5, beginning with the methodology used, and presenting a number of results, finishing with conclusions and areas for improvement.

2. Literature Review

In this chapter, we will discuss the nature of existing threats to data, and the ongoing research in this area. We will then consider the specific threats posed by e-mail.

2.1. General Data Leakage

The importance of data leakage is gaining more importance as the amount of information stored about entities increases, and the risks are being considered more carefully. From the obvious ramifications for businesses discussed in Papadimitriou and Garcia-Molina 2011: the loss of trust and legal action resulting from the discovery of leaking data, to the more personal issues discussed in Irani et al. 2011: the possibility of using the discovered data to discover passwords or to physically identify them. 53% of Americans can be uniquely identified by their birth date, gender and location (city/town), with the number jumping 87% when using birth date, gender and zip code.

2.1.1. Personal Data Leakage

From a personal perspective, there are a number of risks. There are a significant number of social networks available, with an estimated 1.65 billion monthly active users, with a significantly higher proportion used in developed countries. Irani et al. 2011 showed the rate at which the information gathered from social networks can be used to uniquely identify an individual by defining the aggregate normalised attribute leakage as

$$\Psi(F_a, P) = \frac{\sum_{f_a \in F_a} \phi(f_a, P)}{|F_a|} \text{ where } \phi(f_a, P) = [f_a \in P]$$

for a user's social footprint P , and attributes are referenced as f_a which are members of F_a .

Only 9 sites are required before there is approximately a 0.7 attribute leakage, corresponding to a 70% probability that both a person's hometown and name could be recovered. A similar number of sites can give an aggregate normalised attribute leakage of 1, where it becomes almost certain that an individual may be uniquely identified.

2.1.2. Corporate Data Leakage

When companies receive user data, they often have a legal obligation to ensure that the data is protected and treated as confidential and sensitive. When this trust is broken, there are often severe consequences, both from regulators and consumers moving their business to competitors.

In addition to ensuring that human procedures are present to ensure the integrity and confidentiality of data, it is also necessary to ensure that robust technical measures are in place to prevent data breaches. The Identity Theft Resource Center 2016 lists a total of 12 million data records from 399 breaches as having been illegitimately accessed in 2016 to date.

Two fifths of these data records are connected to government or military data breaches, with a third linked to medical and healthcare data, and a further fifth connected to business data.

Squicciarini, Sundareswaran, and Lin 2010 considers one way that data may be leaked, despite care being taken to ensure that it is properly encrypted and stored: by failing to protect against the indices for databases being stored insecurely, customer data may be leaked. Order-preserving encryption schemes, as described in Agrawal et al. 2004, is one way of solving this problem, to an extent.

2.2. Data Leakage from E-Mails

2.2.1. E-Mail Headers

All e-mails include additional information about the sender and receiver, some of which is used by an e-mail client in order to display more information about the message that is currently being viewed, such as its original sender, reply-to addresses and the time it was sent. Additional fields allow senders to authenticate themselves using public-key methods.

The format of e-mail headers was first defined in RFC822, and further refined in subsequent RFCs. The standard for e-mails was then formalised precisely in RFC5322.

2.2.2. Example Header and Pertinent Data

In the example below, and text highlighted with red, like so is information about the receiver. Information about the sender, their hardware or software is highlighted in green, like so; and information gathered about intervening devices is highlighted in blue, like so.

```
Delivered-To: joshuaclark94@gmail.com
Received: by 10.25.150.146 with SMTP id y140csp5431371fd;
    Sat, 6 Feb 2016 08:49:56 -0800 (PST)
X-Received: by 10.112.12.2 with SMTP id u2mr8302831lbb.145.1454777396580;
    Sat, 06 Feb 2016 08:49:56 -0800 (PST)
Return-Path: <agatabor@poczta.onet.pl>
Received: from smtpo75.poczta.onet.pl (smtpo75.poczta.onet.pl. [141.105.16.25])
    by mx.google.com with ESMTPS id o199si122556361fb.94.2016.02.06.08.49.56
    for <joshuaclark94@gmail.com>
    (version=TLS1_2 cipher=ECDHE-RSA-AES128-GCM-SHA256 bits=128/128);
    Sat, 06 Feb 2016 08:49:56 -0800 (PST)
Received-SPF: pass (google.com: domain of *****@poczta.onet.pl designates
    141.105.16.25 as permitted sender) client-ip=141.105.16.25;
Authentication-Results: mx.google.com;
spf=pass (google.com: domain of *****@poczta.onet.pl designates 141.105.16.25
    as permitted sender) smtp.mailfrom=agatabor@poczta.onet.pl
Received: from [10.26.196.156] (client-8-32.eduroam.oxuni.org.uk [192.76.8.32])
    (Authenticated sender: *****@poczta.onet.pl)
    by smtp.poczta.onet.pl (Onet) with ESMTPA id 3pyKNH4ffyzT6tkv8
    for <joshuaclark94@gmail.com>; Sat, 6 Feb 2016 17:49:50 +0100 (CET)
Date: Sat, 06 Feb 2016 16:49:07 +0000
Subject: Test e-mail
Message-ID: <j66i9tkyhy3l4v77erlw1gne.1454777347191@email.android.com>
Importance: normal
```

```

From: ***** <*****@poczta.onet.pl>
To: Joshua Clark <joshuaclark94@gmail.com>
MIME-Version: 1.0
Content-Type: multipart/alternative;
        boundary="--_com.android.email_1892258509098440"

----_com.android.email_1892258509098440
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: base64

CgoKC1NlbnQgZnJvbSBteSBTYW1zdW5nIEdhbGF4eSBzbWYdHBob25lLg==

----_com.android.email_1892258509098440
Content-Type: text/html; charset=utf-8
Content-Transfer-Encoding: base64

PGh0bWw+PGhlYWQ+PG1ldGEgaHR0cC1lcXVpdj0iQ29udGVudC1UeXB1IiBjb250ZW50PSJ0ZXh0
L2h0bWw7IGNoYXJzZXQ9VVRGLTgiPjwvaGVhZD48Ym9keT48ZG12IHNOeWxlPSJ3b3JkLWJyZWFr
O2t1ZXAtYWxsOyI+PGJyPjxicj48YnI+PGJyPlNlbnQgZnJvbSBteSBTYW1zdW5nIEdhbGF4eSBz
bWYdHBob25lLjxicj48L2Rpdj48L2JvZHK+PC9odG1sPg==

----_com.android.email_1892258509098440--

```

E-Mail Header Fragment 2.1: Sample E-Mail

The particularly interesting portions of the e-mail header include the IP addresses of the various servers the message has travelled through, allowing their approximate location to be determined. Additionally, the information on the protocol being used and the software being run allows for anyone with access to mail headers to find more information about the attacks a device and its software may be vulnerable to.

Example 2.2 show a server with an internal IP address of 10.25.150.146 in the Pacific Seaboard Timezone received the e-mail using the SMTP protocol.

```

Received: by 10.25.150.146 with SMTP id y140csp543137lfd;
        Sat, 6 Feb 2016 08:49:56 -0800 (PST)

```

E-Mail Header Fragment 2.2: E-Mail Server configuration information

In Example 2.3, a number of pieces of information can be extracted: the hostname of the sending device is `client-8-32.eduroam.oxuni.org.uk` with associated 192.76.8.32 is on the eduroam network, with a local IP address of 10.26.196.156. By performing a GeoIP lookup¹ on this address, we find that it has a location of (51.75, -1.25) positioning it somewhere within Oxford.

```

Received: from [10.26.196.156] (client-8-32.eduroam.oxuni.org.uk [192.76.8.32])

```

E-Mail Header Fragment 2.3: Information revealed in Received field

Further samples from headers that may be particularly interesting include the following examples.

Many Apple iOS devices will include hardware and version names in the **X-Mailer** field.

¹See <https://geoiptool.com/en/?ip=192.76.8.32>

X-Mailer: Zimbra 8.6.0_GA_1153 (MobileSync - Apple-iPhone7C2/1305.238)

E-Mail Header Fragment 2.4: Apple iPhone version

Alternatively, some versions of Mozilla Thunderbird include the software details in the **User-Agent** field, with the following example showing the currently running operating system type as well (64-bit Linux).

User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101
Thunderbird/45.1.0

E-Mail Header Fragment 2.5: Thunderbird Information

2.2.3. Existing Research

In Nurse et al. 2015, the idea of using the information available in an email header was mooted, turning the previously standard threat of malware and phishing contained in received e-mails on its head, and instead presenting the threat in outgoing emails, and the personally identifying information (PII) contained therein. Many emails leaked information about employers, e-mail services and applications used, and IP address. Initial examination of a variety of e-mail headers found within my own inbox also revealed a plethora of information, including phone carriers, preferred languages, and system usernames. It is conceivable therefore, that it is possible to automate at least part of this, and present the information that can be extracted, in a white-hat tool to allow people to audit the information that they are revealing. The obvious malicious use-case involves using such information as part of a spear-phishing exercise.

An alternative vulnerability presents itself in the information about systems that may be revealed. Many email clients embed identifying information, and there are multiple databases available to allow specific threats to be identified. This could allow a malicious entity to compromise the security of a target machine, and gain access to the data stored on that machine and available on any connected network devices. Work started in Joshi, Lal, and Finin 2013 discusses the need to aggregate data about vulnerabilities from multiple sources to present a more complete and coherent picture, which is also likely to then contain more accurate data.

Al-zarouni 2004 presents an alternative set of results, describing how an individual can seek to protect themselves against malicious e-mails, using the contents of e-mail headers. Various discrepancies between forged e-mail addresses and legitimate messages are described.

2.3. Existing Tools

Several tools already exist online to display the information that is found in e-mail headers, with tools from Microsoft and Google existing to analyse the contents of e-mail headers. Additionally, many e-mail clients will display some of the information found within headers to provide additional context for a message, however, this is rarely additional information about the parties involved in an exchange.

2.3.1. Google

The Google Apps Toolbox features an e-mail header analyser². An example of the output of the utility is found in Figure 1.

One of the most useful features from the Google Apps Toolbox is the information provided about the servers the message travelled through. This tool shows the details of the time taken for each hop, and the protocol used.

MessageId:		<201107031502.p63F2i2m001182@nyork.iii.com>				
Created at:		Sun Jul 03 2011 08:02:18 GMT-0700 (PDT) (Delivered after 19 mins)				
From:		New York Public Library				
To:		some_random_user@gmail.com				
Subject:		New York Library items due soon.212-555-2329				

#	Delay	From		To	Protocol	Time received
0		localhost.localdomain	→	nyork.iii.com	ESMTP	Sun Jul 03 2011 08:02:18 GMT-0700 (PDT)
1	19 mins	nyork.iii.com	→	sienna.pobox.com	ESMTP	Sun Jul 03 2011 08:21:05 GMT-0700 (PDT)
2	1 sec	localhost	→	sienna.pobox.com	ESMTP	Sun Jul 03 2011 08:21:06 GMT-0700 (PDT)
3		sienna.pobox.com	→	[Google] mx.google.com	ESMTP	Sun Jul 03 2011 08:21:06 GMT-0700 (PDT)
4	1 sec		→	[Google] 10.52.65.169	SMTP	Sun Jul 03 2011 08:21:07 GMT-0700 (PDT)
5	3 sec		→	[Google] 10.229.234.71	SMTP	Sun Jul 03 2011 08:21:10 GMT-0700 (PDT)

Show Raw header

Figure 1.: Google Apps Toolbox E-mail header output

2.3.2. Microsoft

Microsoft has released its own e-mail message header analyser, the Microsoft Message Header Analyser³ and showing sample results in Figure 2 is of a similar nature to the Google tool, discussed in Subsection 2.3.1. In addition to the information presented by Google, this tool also produces a set of “Other Headers”, highlighting fields that may be of interest. However, little additional context is provided as to their relevance.

2.4. Vulnerabilities

Beware of bugs in the above code; I
have only proved it correct, not tried it.

Donald Knuth

Problems in software are nothing new, and seeking to exploit these issues is almost as old. As the security implications behind flawed software became more widely recognised, reducing their impact wherever possible became the next most important step. The MITRE Corporation operates the National Cybersecurity Federally Funded Research and

²Found at <https://toolbox.googleapps.com/apps/messageheader/>
³Found at <https://testconnectivity.microsoft.com/MHA/Pages/mha.aspx>

Message Header Analyzer

Insert the message header you would like to analyze

--001a11c075a2ebbc9c052e50c47d
Content-Type: text/html; charset=UTF-8
Content-Transfer-Encoding: quoted-printable

<div dir=3D"ltr">Hi Joshua!<div>
</div><div>CONGRATULATIONS!!!=C2=A0</div>
><div>I am super glad you are going to join us.=C2=A0</div><div>Do you know=
when you want to start?</div><div>
</div><div>Ewa</div></div>

--001a11c075a2ebbc9c052e50c47d--

Analyze headers

Clear

Get the Message Header Analyzer App for Office

Summary

Subject

Welcome to Google!

Message Id

<CAMDwjz=HOBPVg1qpU1caMfK0y1VNOEPBvNw6EBORuEzN-+Y5fg@mail.gmail.com>

Creation time

18/03/2016, 11:08:20 (Delivered after 1 minute 6 seconds)

From

Ewa_Macias <emacias@google.com>

To

joshua@clark.io

Received headers

Hop #	Submitting host	Receiving host	Time	Delay	Type
1		10.31.65.194	18/03/2016, 11:08:20		HTTP
2		mail-vk0-f51.google.com	18/03/2016, 11:09:00	40 seconds	SMTP
3	mail-vk0-f51.google.com ([209.85.213.51]:32795)	server135.web-hosting.com	18/03/2016, 11:09:26	26 seconds	esmtps (TLSv1.2:AE5128-GCM-SHA256:128) (Exim 4.86_1) (envelope-from <emacias@google.com>)

Other headers

#	Header	Value
1	Return-path	<emacias@google.com>

Figure 2.: Microsoft E-mail header output

Development Centre, which exists to maintain a database of these vulnerabilities, which are referred to as Common Vulnerabilities and Exposures (CVE).

2.4.1. CVE Contents

This section uses CVE-2016-4065 as a reference, with the original version available at <https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-4065>.

Overview — “The ConvertToPDF plugin in Foxit Reader and PhantomPDF before 7.3.4 on Windows, when the gflags app is enabled, allows remote attackers to cause a denial of service (out-of-bounds read and application crash) via a crafted (1) JPEG, (2) GIF, or (3) BMP image.”

This is a free-text field typically containing the requirements and methodology for an attack.

Impact — This is divided into two sections, the Common Vulnerability Scoring System Version 2 and 3 results. These give a numerical value of the severity of a vulnerability, based on a number of factors, including how easy it is to take advantage of the vulnerability, and the level of impact the attack would have on a given system. Both systems represent this as a vector.

The example entry has the following two vectors:

CVSS:3.0/AV:L/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H

and

(AV:N/AC:M/Au:N/C:P/I:P/A:P)

The first corresponds to a locally available attack, with low complexity not needing privileges but relying on user interaction. Successfully completing this attack gives

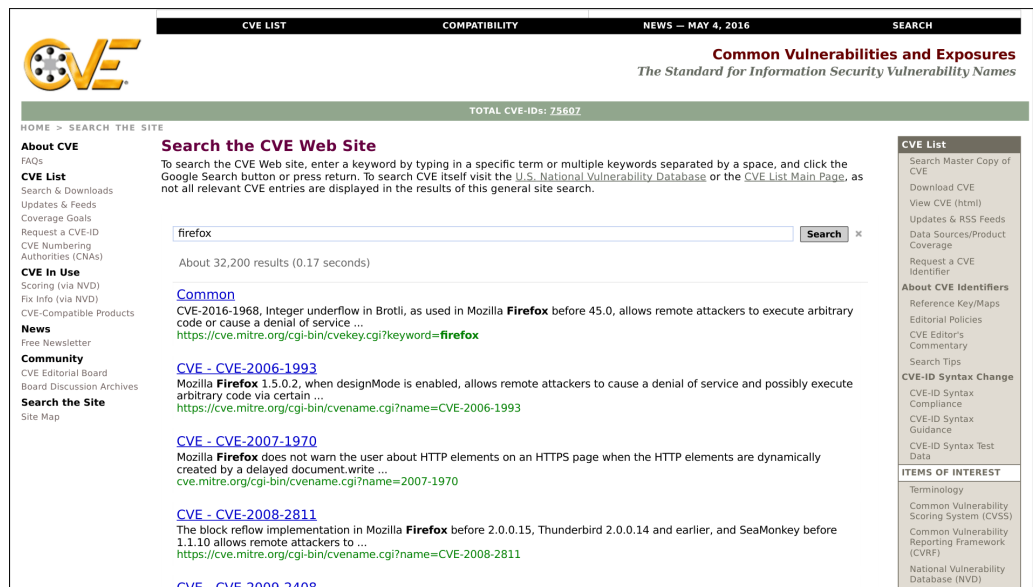


Figure 3.: CVE Search Results

no additional access outside of the scope of the attacked application. Within the application, it is possible to access any information, modify it at will, or prevent access to the application, all of which represent a high threat level.

The second attack vector represents an attack that can be completed via the network, with medium complexity, requiring no authentication. The confidentiality, integrity and availability metrics are all valued as partial, however, no scope context is registered.

Vulnerable Software and Versions — “cpe:/a:foxitsoftware:reader:7.3.0.118::~~~~windows~~~ and previous versions; cpe:/a:foxitsoftware:phantompdf:7.3.0.118::~~~~windows~~~ and previous versions”

Not shown in this example, but a tree structure using conjunctions and disjunctions is used to represent the software that is required in order for the vulnerability to be exploited.

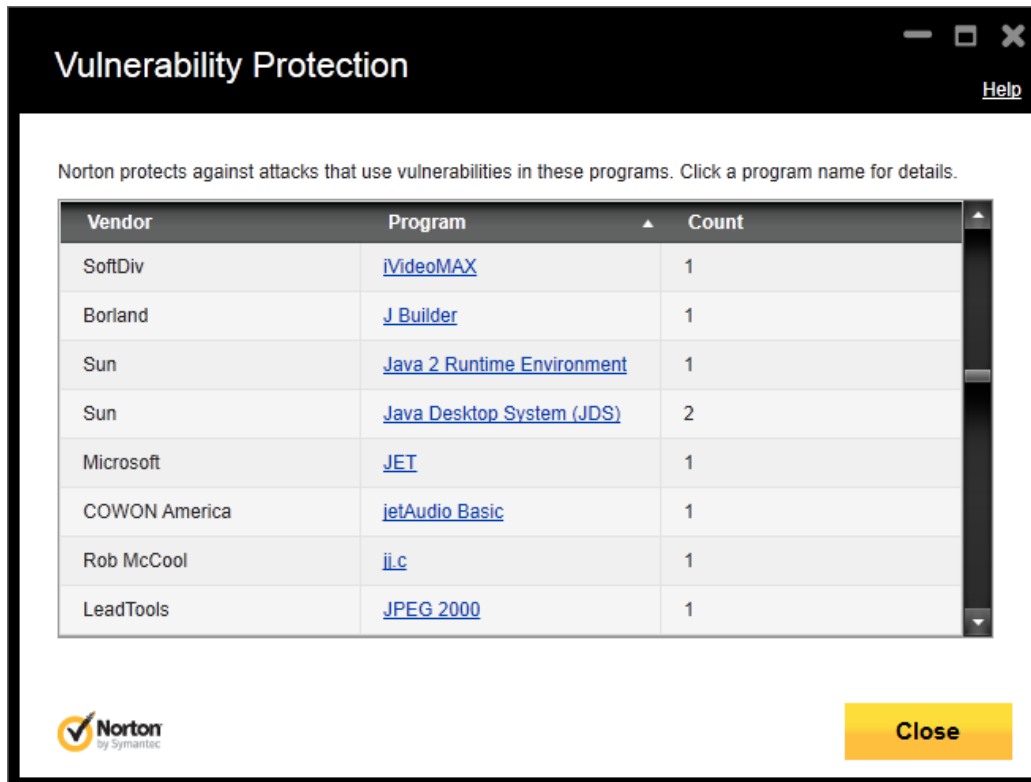
2.4.2. MITRE CVE Lookup

There are a number of tools to look up CVEs⁴ and showing sample results in Figure 3. There are a number of limitations to the results returned by the CVE Mitre tool. Firstly, little context is returned: information about scores, the impact and access information are omitted, for example. Additionally, the process of finding relevant vulnerabilities is further slowed down by the necessity to search for specific terms one at a time. There are automated tools exist at a consumer and enterprise level that will automatically scan a computer or network to detect installed software configurations and show the results.

⁴Found at <https://www.cve.mitre.org/find/index.html>

2.4.3. Norton Vulnerability Protection

For example, the now deprecated Norton Vulnerability Protection tool, as shown in Figure 4⁵ lists the programs and the total number of vulnerabilities found, providing more information on each program. This method has the advantage of indicating the specific programs that have vulnerabilities, with the aim of allowing a user to update their vulnerable applications, however it does not allow for more fine-grained information.



Vendor	Program	Count
SoftDiv	iVideoMAX	1
Borland	J Builder	1
Sun	Java 2 Runtime Environment	1
Sun	Java Desktop System (JDS)	2
Microsoft	JET	1
COWON America	jetAudio Basic	1
Rob McCool	jlc	1
LeadTools	JPEG 2000	1

Figure 4.: Norton Vulnerability Protection Results

2.5. Summary

There breadth of research available indicates the importance that is placed on maintaining data security, as well as the attempts to track and report on breaches, allowing individuals and companies to track when their data may be exposed. There is also a research available on the data that is willingly disclosed by individuals, and this analyses the risk to individuals that a malicious actor accumulating this data can present.

Less research is available on the nature of the data leaked through e-mails, though it is becoming more common, with information also being provided on ways to protect against malicious or spoofed e-mails, as well as significant amounts of work on spam-detection.

There are a number of specific tools available designed to analyse e-mail headers, however, their main focus is usually on presenting the trace fields, rather than the information that

⁵Available at community.norton.com

may be extracted from the rest of the e-mail.

Finally, available tools for analysis of CVEs tend to have divergent aims, with the tool either being aimed at experienced professionals, offering a lot of data, but only if the user is familiar with the system of CVEs, or offering just enough information to allow a system to be kept updated as vulnerabilities are discovered.

3. An Approach to Support Understanding of Data Leakage in E-Mail Headers

3.1. Overview

In order to satisfy the aims of this project, and building on the presented objectives, this chapter discusses the requirements of the software intended to analyse e-mails and the design of the program and its specifications.

3.2. Program Specification

The software should automatically extract information from e-mail headers and analyse its results to display the personal information contained within an e-mail's header, as well as information about the software configurations that may be found on a user's computer, or the servers used to send their e-mail.

The program would be expected to satisfy the following minimal requirements in order for it to be considered successful:

Accuracy — any information produced by the parser should be reflective of the input e-mail

Representation — the produced visualisation should be intuitive to read: each element should be presented separately from the others, and clearly labelled.

Portability — the visual output produced by the program should be available to the user in a variety of formats.

Interactivity — the program should produce sensible warnings when an e-mail that is not possible to parse has been entered.

3.3. Program Overview

The program is split up into three main stages: textual analysis and parsing; header contents analysis, and visualisation, as shown in Figure 5. The relevant data is often stored in a central `MainWindow` class, rather than passed as a parameter, as the composition would indicate, allowing clear references to be maintained.

The analysis is implemented as a series of stages, firstly, the e-mail header is parsed, to extract important information to a predefined set of Java objects. This is followed by the analysis phase, where the resultant data is passed to a set of analyser modules, each running separately. Finally, this information is presented to the user. After discussing an overview of each module, this chapter presents each of these stages in detail.

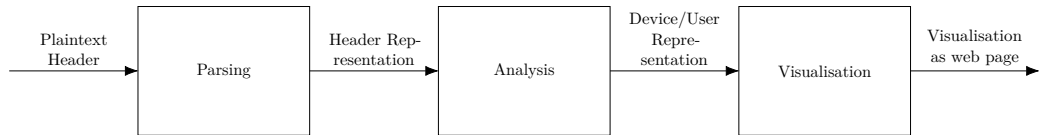


Figure 5.: Simplified Control Flow of Application

3.3.1. Parsing

This module receives the plain-text of the e-mail as an input, splitting it into two sections, fields with the “Received” tag, and all other fields. These are then parsed separately. The other fields are easier to parse, as they can be loaded into a hash-map, split by the colon. The trace fields require a more complex parsing strategy, fully described in Section 4.3.1. This information is then extracted to more abstract Java objects, allowing the relevant information to be queried on a device-by-device basis, rather than constantly referring back to the source text.

3.3.2. Analysis

The analysis of the e-mail headers is handled independently by a number of small classes, each running asynchronously. The decision was made to structure the program in such a way that concurrent operation was possible in order to prevent blocking operations from limiting the progress of other operations. This also required care to ensure the separation between the model of the header and the model of the available information was maintained, as no guarantees could be placed on which thread was modifying data.

It is in this part of the program that the automated discovery of vulnerabilities takes place. An example software configuration string is `cpe:/a:cloudbees:jenkins:2.2`, so it is necessary to attempt to convert found software information. For example, **Apple Mail** would become `apple:mail` and a search would be performed over the database for configurations containing that string.

3.3.3. Visualisation

Once the header has been analysed, it is then necessary to present the information that has been found in a useful and informative manner. Firstly, the information about the individual sender, such as their name, organisation, software and usernames, should be presented (where it exists). The information about the servers that the e-mail has passed through should then be detailed, their address, location and software, as well as relevant vulnerabilities.

Once this has been listed, information about the scores of the vulnerabilities for each piece of found software, and a listing of the available vulnerabilities should be made. By providing a visualisation of the distribution of the scores assigned to each piece of software conclusions can be drawn about the expected severity of future vulnerabilities.

3.4. Comparisons to Existing Software

Google and Microsoft Header Analysers Both of the tools discussed in Sections 2.3.1 and 2.3.2 produced detailed information on the servers that are being used to send and received messages, with Google’s tool also clearly reporting when its own servers were used to send a message. The software should mimic this by providing a similar level of detail on the devices being used to send information, and extend this by looking up information on the device’s owning organisation. Additionally, a more exhaustive search of the other fields should be conducted, so more information about the sender can be provided than the immediately available details such as time, sender’s e-mail and recipient.

MITRE CVE Lookup and Norton Vulnerability Protection The tools discussed in Sections 2.4.2 and 2.4.3 are both targeted at very different demographics. The MITRE CVE Lookup is designed for IT professionals and system administrators wishing to gather more information about specific vulnerabilities and software, requiring knowledge of the software present on a network or device. The information presented is also not structured or sorted in a clear fashion. Norton’s tool, on the other hand, is focused on individual users who administer their own systems. The information is presented in such a way as to warn them as to which software needs updating or patching against a vulnerability, but provides few details on the nature of the vulnerabilities.

The tool that is described in this report ought to be able to bridge the gap between these two tools, providing a useful and relevant list of vulnerabilities, without overloading the information provided, or requiring complex search terms to be crafted.

3.5. Typical Use

On starting the application, the user will provide an e-mail that they wish to have analysed. This will then be parsed, and some relevant information presented in a table.

Lastly, an option is available to view the information about security vulnerabilities in a separate webpage, forming the main output of the program.

The resultant webpage will be structured as in Table 1. It will then be possible for the user to click on the representations of the devices to find out more information. It will also be possible to search within the vulnerability list to find more information, as well as filter by impact and availability details.

Email Header Information		
Sender Information (Name, originating domain)	Sender Software	Sender Usernames (Presented as a list with likely organisation)
Graphical representation of devices used to deliver the e-mail		
List of derived information including found software and similar information		
Histograms for vulnerability scores, separated by product		
Table of discovered CVEs that can be searched and filtered		

Table 1.: Format of presented data found in e-mail header

4. Implementation

4.1. Overview

In order to satisfy the aims of this project, and building on the presented objectives, this chapter discusses the implementation of the software intended to analyse e-mails, starting with the required definitions before presenting the algorithms and data structures that are needed.

4.2. Definitions

The following covers the essential definitions required for the notation and concepts that will be discussed in this document.

4.2.1. Parsing

In order to aid the parsing of the e-mail header, a combination of regular expressions and context-free grammars are needed, and defined as follows.

Alphabets and Languages A set of symbols, usually denoted as Σ . A language is a subset of $\mathcal{P}(\Sigma)$.

The following special classes are provided as part of the Perl-Compatible Regular Expression library, and are subsets of the alphabet of Unicode characters, defined in PHP Group et al. n.d.

alnum — letters and digits

alpha — letters

ascii — the set of ASCII characters (character codes 0 — 127)

blank — tabs or blank spaces

cntrl — control characters

digit — decimal digits

graph — printing characters (excluding spaces)

lower — lower-case letters

print — printing characters (including spaces)

punct — punctuation marks (printing characters excluding letters and spaces)

space — white space

upper — upper case letters

word — “word” characters (same

xdigit — hexadecimal digits

Regular Languages

Regular languages are defined as follows:

- \emptyset and $\{\epsilon\}$ are regular languages
 - for each $a \in \Sigma$, $\{a\}$ is a regular language
 - if A and B are both regular, $A \cup B$, $A \cdot B$ and A^* are regular languages.
 - $A \cup B$ is the union of two languages.
 $A \cup B = \{s : s \in A \vee s \in B\}$
 - $A \cdot B$ is the concatenation of two languages. $A \cdot B = \{ab : a \in A, b \in B\}$
 - A^* is the Kleene star of a language.
- $$A_0 = \{\epsilon\}$$
- $$A_1 = A$$
- $$A_{i+1} = \{aa' : a \in A_i, a' \in A\}$$
- $$A^* = \bigcup_{i \in \mathbb{N}} A_i$$

Context-Free Grammars

A context-free grammar G is defined as $G = (V, \Sigma, R, S)$ where:

- V is a variable.
- R is a relation defined over $V \rightarrow (V \cup \Sigma)^*$
- Σ is the alphabet of symbols.
- S is the start symbol

For example, $\langle S \rangle$ is the field name with the associated productions $\langle T \rangle \langle U \rangle$, where T and U are productions.

$$\langle S \rangle \models \langle T \rangle \langle U \rangle$$

For example, $\langle S \rangle$ is the field name with the associated productions $a \langle U \rangle$, where a is a terminal symbol.

$$\langle S \rangle \models a \langle U \rangle$$

This is then extended in the following ways used in the RFC syntax.

The square brackets are used to indicate an optional element.

$$\langle \text{field} \rangle \models \langle \text{field-name} \rangle : [\langle \text{field-body} \rangle] \text{ CRLF}$$

The asterisk is used to indicate an element that appears 0 or more times. n^* is used to indicate a component that repeats n or more times.

$$\langle \text{fields} \rangle \models \langle \text{dates} \rangle \langle \text{source} \rangle 1^* \langle \text{destination} \rangle * \langle \text{optional-fields} \rangle$$

The hash-symbol is used to indicate an element that appears a certain number of times. $m * n$ is used to indicate a component that repeats at least m times and at most n times.

$$\langle \text{fields} \rangle \models \langle \text{dates} \rangle \langle \text{source} \rangle 1\# \langle \text{destination} \rangle * \langle \text{optional-fields} \rangle$$

The $|$ is used to indicate a selection between a pair of elements.

$$\langle \text{fields} \rangle \models a \mid b$$

4.2.2. Database Queries

The following notations will be used for the CVE database queries.

Set-Theoretic Operators The operators $F \cup G$, $F \cap G$, $F \setminus G$ behave as is expected for these operators, resulting in the union, intersection and difference of the sets. The only proviso being that the attribute names must match.

Selection

$$\sigma_{\text{product}=\text{thunderbird}} D$$

The above notation is used to indicate a search over the attribute named “product” for the string “thunderbird” in the database table D . As a single database is only being used, this may be occasionally elided. The output of this function is another object of the same type as D .

Projection

$$\pi_{\text{product}} D$$

The above notation is used to indicate a projection on the attribute named “product” in the database table D . The output of this function is another object of the same type as D .

Composition The above functions results can be composed repeatedly to produce more specific search queries.

4.2.3. Data Structures

These will be drawn using square boxes to represent single objects that are encapsulated within an object. Square boxes with an inner square box indicate some collection of objects.

Thin arrows will be used to denote the encapsulation relation, with thicker arrows being used to list relevant public methods.

Where the type of an object can be expressed simply, the following conventions are used:

Functions — $\alpha \rightarrow \beta$ **Str** is a function from α to β stored using a **Str** object in Java.

Tuples — (α, β) represents a two-tuple containing objects of type α and β , respectively. This extends naturally for arbitrarily many objects.

Lists — $[\alpha]$ represents a list or array of objects, all with type α .

4.3. Data Extraction and Parsing

The parser's operation completes in a number of stages, following RFC822 (Crocker 1982). The header is divided up into two disjoint sections, the routing information (**Received from...**) and the key-value map of other pertinent information.

4.3.1. Received fields

The received fields are the most complicated part of the e-mail header to parse, as they are described by a non-trivial grammar, presented below.

$\langle \text{message} \rangle$	\models	$\langle \text{fields} \rangle * (\text{CRLF} * \text{text})$
$\langle \text{fields} \rangle$	\models	$\langle \text{dates} \rangle \langle \text{source} \rangle 1* \langle \text{destination} \rangle * \langle \text{optional-fields} \rangle$
$\langle \text{field} \rangle$	\models	$\langle \text{field-name} \rangle : [\langle \text{field-body} \rangle] \text{CRLF}$
$\langle \text{field-name} \rangle$	\models	<i>any word consisting of CHAR, excluding CTLs, SPACE, and "":</i>
$\langle \text{field-body} \rangle$	\models	$\langle \text{field-body-contents} \rangle [\text{CRLF LWSP-char} \langle \text{field-body} \rangle]$
$\langle \text{field-body-contents} \rangle$	\models	<i>ASCII characters</i>
$\langle \text{source} \rangle$	\models	$[\langle \text{trace} \rangle] \langle \text{originator} \rangle [\langle \text{resent} \rangle]$
$\langle \text{trace} \rangle$	\models	$\langle \text{return} \rangle 1* \langle \text{received} \rangle$
$\langle \text{return} \rangle$	\models	Return-path: $\langle \text{route-addr} \rangle$
$\langle \text{received} \rangle$	\models	Received:
$\langle \text{cont.} \rangle$	\models	[from $\langle \text{domain} \rangle$]
$\langle \text{cont.} \rangle$	\models	[by $\langle \text{domain} \rangle$]
$\langle \text{cont.} \rangle$	\models	[via $\langle \text{atom} \rangle$]
$\langle \text{cont.} \rangle$	\models	*(with $\langle \text{atom} \rangle$)
$\langle \text{cont.} \rangle$	\models	[id $\langle \text{msg-id} \rangle$]
$\langle \text{cont.} \rangle$	\models	[for $\langle \text{addr-spec} \rangle$]
$\langle \text{cont.} \rangle$	\models	; $\langle \text{date-time} \rangle$
$\langle \text{msg-id} \rangle$	\models	$<\langle \text{addr-spec} \rangle>$
$\langle \text{addr-spec} \rangle$	\models	$\langle \text{local-part} \rangle @ \langle \text{domain} \rangle$
$\langle \text{local-part} \rangle$	\models	$\langle \text{word} \rangle * (. \langle \text{word} \rangle)$
$\langle \text{word} \rangle$	\models	$\langle \text{atom} \rangle \mid \langle \text{quoted-string} \rangle$
$\langle \text{domain} \rangle$	\models	$\langle \text{sub-domain} \rangle * (. \langle \text{sub-domain} \rangle)$
$\langle \text{sub-domain} \rangle$	\models	$\langle \text{domain-ref} \rangle \mid \langle \text{domain-literal} \rangle$
$\langle \text{domain-ref} \rangle$	\models	$\langle \text{atom} \rangle$
$\langle \text{date-time} \rangle$	\models	[day,] <i>date time</i>
$\langle \text{atom} \rangle$	\models	$1* \text{any character excluding specials, SPACE and CTLs}$

An example field is as follows:

Received: from relay12.mail.ox.ac.uk (129.67.1.163)

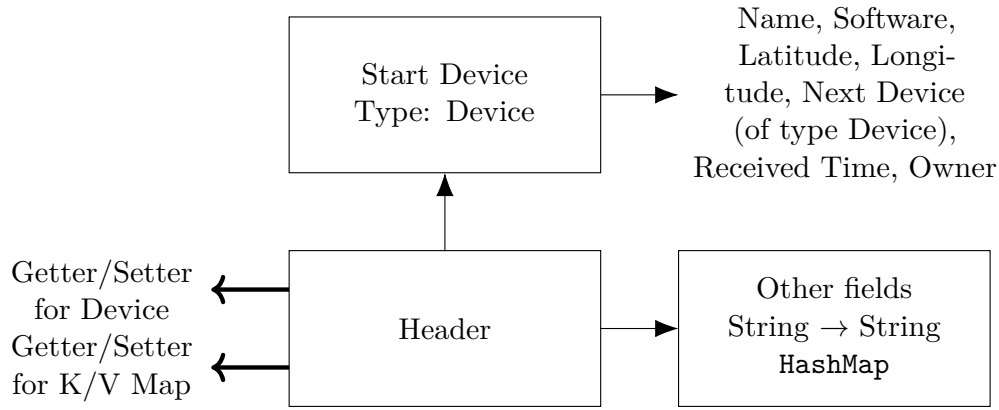


Figure 6.: Header Data Structure Format

```

by HUB05.ad.oak.ox.ac.uk (163.1.154.231)
with Microsoft SMTP Server id 14.3.169.1;
Sat, 14 Nov 2015 10:55:35 +0000

```

Of particular interest is the pair of hostnames (both are needed as each line is analysed independently, therefore it is not necessary to treat the last line as a special case) and their associated IP addresses. The hostname is of more interest, as performing an IP address lookup is less complicated than determining a hostname. After determining the IP address, it is then often possible to perform a GeoIP lookup to determine an approximate location for the device. Sometimes, by determining the hostname, it is then possible to perform a WhoIs search for information about the owner of the domain. The software used is identified by the `with` field, and is also of interest, however, is insufficient in most cases to produce meaningful CVE data.

4.3.2. Other fields

These are read by a Python script and output to `STDOUT` to be read by the Java parser in a consistent format. These are then loaded into a hash-map to allow quick lookup.

4.3.3. Input Data Structures

The raw string of the message header is the only input to this module.

4.3.4. Output Data Structures

The data structure presented in Figure 6 shows the output of the parsing and textual analysis module, which is then provided as an input to the analysis modules.

4.4. Analysis

After completing the parsing of the fields, it is then ready to be analysed for different features. All of the analysers implement the `HeaderAnalyser` interface, requiring information about

the header to be analysed, and the currently running application. All of these then implement the `Runnable` interface, allowing the class to be run asynchronously.

4.4.1. Text-Based

The fields from the header are analysed in different modules, with searches being performed for specific strings. Of particular interest to Oxford Nexus users is the “X-Oxford-Username” string, containing the username of the individual that sent the message. As confirming the username is a fairly standard security procedure for an IT support technician, having access to this information could allow a phisher in a later stage of an attack to increase their credibility.

In some cases, the likely keys that are being searched for are known in advance, and can then be checked against the hash-map of entries.

An example of this approach is for the specific check for an Oxford username, as shown in Algorithm 1.

```

Input: Header
Output: Any username that is found
 $kws \leftarrow \{X-Oxford-Username, X-Username, X-Authenticated-User\};$ 
foreach  $kw \in kws$  do
    if  $kw \in Header.KvMap$  then
        return  $Header.KvMap(kw);$ 
    end
end

```

Algorithm 1: Lookup based on a known key

Alternatively, we may be interested in properties of the keys, necessitating a search over the keys, as shown in Algorithm 2.

```

Input: Header
Output: Any information relating to Microsoft Exchange that is found
foreach  $Key\ k \in Header.KvMap$  do
    if  $k$  starts-with  $X-MS-Exchange$  then
        return  $Header.KvMap(k);$ 
    end
end

```

Algorithm 2: Lookup based on a key property

4.4.2. Client Inference

In Nurse et al. 2015, a number of different e-mail clients were identified based on the header tags that were present. By identifying these pieces of software likely to be found on a user’s machine, we gain a significant amount of information from them, and should therefore devote some effort to correctly identifying them. Using a number of e-mail samples provided, I have been able to extract examples for a number of different e-mail clients, and use them to infer the client being used. Using a sample of e-mails, it is possible to find information for

additional e-mail clients and senders, such as (but not limited to) PHPMailer and Foxmail. A number of these approaches are shown in Algorithm 3.

Input: Header

Output: The name of the software that is likely being used, and necessary CVE
Product name (elided for brevity)

OutlookKeywords

$\leftarrow \{\text{Accept-Language, Content-Language, Threat-Index, Threat-Topic}\};$

if “Message-ID” \in Header.KvMap **then**

if Header.KvMap(“Message-ID”) contains “email.android.com” **then**

return “Android Device”;

end

else if “X-Mailer” \in Header.KvMap **then**

switch Header.KvMap(“X-Mailer”) contains **do**

case “iPhone” **do**

return “iPhone”

end

case “Outlook Express” **do**

return “Microsoft Outlook Express”

end

 ...

end

else if “User-Agent” \in Header.KvMap **then**

return “Thunderbird”;

else if Header.KvMap \cap OutlookKeywords $\neq \emptyset$ **then**

if “X-Mailer” \in Header.KvMap **then**

return Apple Mail

else

return Outlook

end

end

Algorithm 3: Client Inference Technique

4.4.3. Database Queries

Using the results gathered from the text-based queries and analysis of the received fields, relevant software configurations are extracted and queried against results in the CVE database. These are then parsed and collated in preparation for displaying the outputs. Specifically, the queries are limited to those matching the product name, and vulnerabilities that can be remotely executed.

As more information is found, more details of products used will also become available. These are added asynchronously.

4.4.4. Analysis Modules and Data Flow

The following modules are used in the analysis of e-mail headers. Via HeaderAnalyser, they all subclass Callable<Object>, allowing them to return values to calling classes when

Input: Header product name p
Output: CVE Entries
 $cve_list \leftarrow \emptyset$;
foreach $s \in \sigma_{vector \neq LOCAL} \sigma_{product=p} D$ **do**
 $cve_builder \leftarrow \text{blank } cve$;
 $cve_builder.id \leftarrow \pi_{CVE-ID} s$;
 ... – extract other features;
 $cve_list \leftarrow cve_list \cup \text{make}(cve_builder)$;
end
return cve_list ;

Algorithm 4: Extracting CVE entries

side-effects are undesirable.

HeaderAnalyser — the base interface for all analysis modules.

ClientInferer — as described in Algorithm 3, this analyses the entire header, looking for specific indicators relating to e-mail clients.

DeviceAnalyser — Extracts the hostname, and then IP address, or IP address for each device, allowing a lookup of its co-ordinates.

ExchangeHeaderAnalyser — determines if a Microsoft Exchange server has handled the message.

GeoIPAnalyser — Given an IP address, this module looks up the latitude and longitude for said IP address. This search will fail for local IP addresses (commonly found in the 192.168.0.0/16 subnet).

UsernameHeaderAnalyser — This module searches for the specific `X-Oxford-Username` field, one of the most common fields in my collection of e-mail headers, as well as a number of more generic username fields.

SenderInformationExtractor — This is used to lookup an individual’s name from the set of fields, if it is available.

VulnerabilityAnalyser — For each entry from the database as a `String`, this analyser returns a `VulnerabilityDisclosure` object.

VulnerabilityFinderManager — this is an interface for other classes to implement. A reference implementation is provided in `VulnerabilityFinderManagerImpl`, with a simple implementation found in `NoopVulnerabilityFinderManager` for systems that lack the CVE database.

WholsAnalyser — Using a hostname, this looks up the relevant owning organisation for a server, if it is available.

Figure 7 shows the direct interaction of the analysis modules. Unless noted, if an arrow goes from α to β , it is used to indicate that α calls β , passing data from α , and/or receiving data from β .

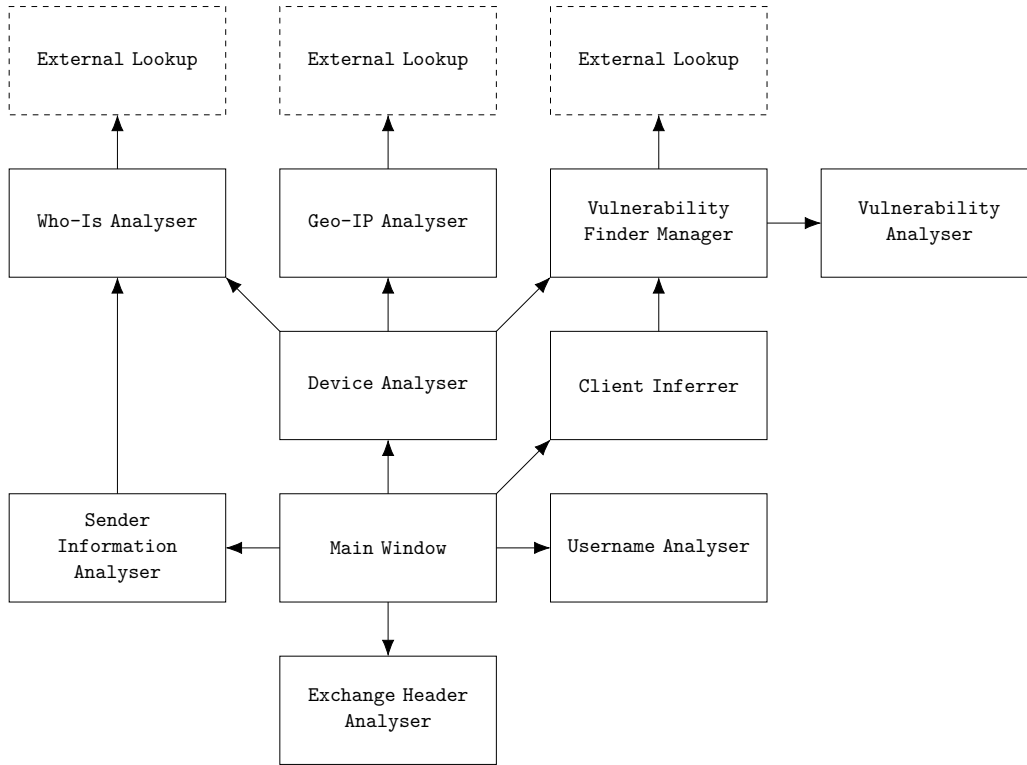


Figure 7.: Information flow between analysis modules

Where possible, these requests are non-blocking, that is run asynchronously, while waiting for other results to be computed. The biggest delay is caused by the CVE Database lookup, followed by the WhoIs lookup and then the GeoIP lookup. The CVE database lookups require the most time as they are often consuming large amounts of data, and searching over specific fields. The WhoIs lookup takes place over the network, causing its response time to be unpredictable, however, its relatively small response size reduces the time needed. Finally, the GeoIP lookup takes place locally, with a relatively small response, allowing it to complete quickly.

4.4.5. Output Data Structures

The output of the analysis modules is compiled into the `FoundInformation` class, which represents the list of facts, servers and other information that has been gathered from an e-mail.

4.5. Visualising the Results

Using a pre-existing template, the results from the e-mail analysis will be presented in a temporary webpage, which can then be saved independently. Other than the referenced JavaScript libraries, and a freely available set of country borders, the document requires no additional information or database access, allowing it to be quickly shared.

In order to export the results, the process described in Algorithm 5 are taken. The

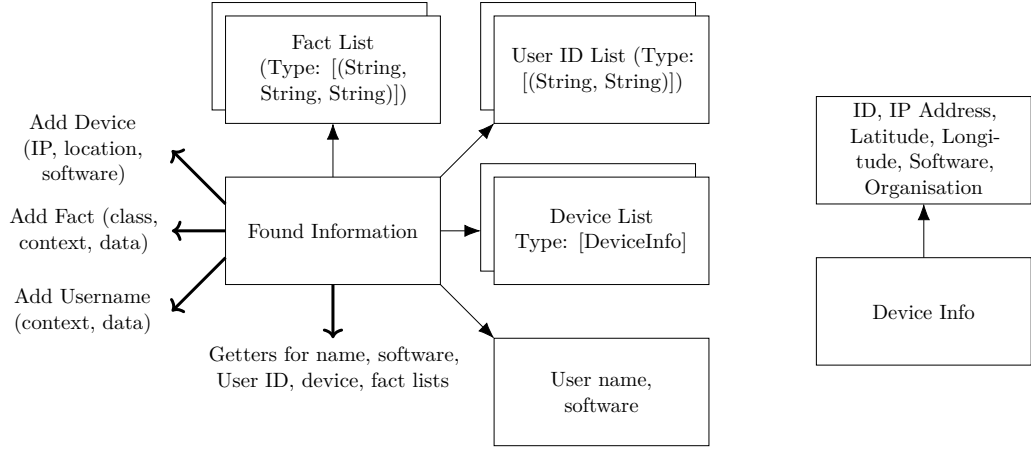


Figure 8.: Found Information Data Structure Format

keyword objects referred to are one of a number of specific strings representing one of the sender's name, organisation, client or usernames; CVE entries, fact entries, products that have been found, or servers that have been found.

Input: Found Information Object

Output: Webpage

```

lines ← read-lines(webpage-template);
foreach l ∈ lines do
  if l contains keyword then
    | webpage ←+ l[field from FoundInformation/keyword];
  else
    | webpage ←+ l;
  end
end
return webpage;

```

Algorithm 5: Exporting the Found Information to Visualisations

While most of the data processing has been completed by the time the webpage is displayed, some computational steps are required to render the histograms and map, which are detailed in Algorithm 6

Input: Webpage text
Output: Visualisation of Webpage

```

foreach  $u \in \text{userentries}$  do find HTML list of usernames and append  $u$ ;
foreach  $f \in \text{factentries}$  do find HTML table of facts and append  $f$ ;
foreach  $c \in \text{cveentries}$  do
    if  $c \text{ matches search string} \vee \text{search string} \stackrel{?}{=} \epsilon$  then
        if  $\langle c_{\text{impact}}, c_{\text{access}} \rangle \text{ matches filters}$  then
            find HTML table of facts and append  $f$ ;
        end
    end
end
foreach  $c \in \text{cveentries}$  do distributions  $[c_{\text{software}}] \leftarrow \cup c_{\text{score}}$  ;
foreach  $d \in \text{distributions}$  do draw histogram for  $d$ ;
foreach  $s_i \in \text{servers}$  do
    project  $s_i$  onto map;
    add  $s_i$  to server table;
end
foreach  $s_i \in \text{servers}$  do draw link between  $s_i$  and  $s_{i+1}$  onto map;
return  $\text{webpage}$ ;

```

Algorithm 6: Rendering the Visualisation

5. Evaluation

After producing the application described in Chapter 4, referring back to the design specifications to determine its performance against the stated criteria was the next step.

5.1. Methodology

Using a sample of e-mails provided by my supervisor, each of them was run through the final version of the program, and scored based on the following attributes:

- Let M be the total number of received fields.
- Let N be the total number of other fields.
- 1 point is added to the total R for each piece of information found in a Received field for the following:
 - One of device name *or* IP address
 - Software *or* protocol used
 - Vulnerabilities found for the relevant piece of software
 - Location Data
- 1 point is added to F for each piece of information found in other fields.

The final score for an e-mail is given as

$$\frac{1}{2} \left(\frac{R}{4M} + \frac{F}{N} \right)$$

to give a value between 0 and 1 for each e-mail.

The e-mails received have been numbered from 1 up to 70, and a random sample of size 30 was selected using the following code:

```
>>> random.seed()
>>> random.sample(list(range(1,70)), 30)
[64, 48, 11, 21, 63, 68, 27, 69, 29, 8, 28,
 34, 13, 57, 10, 3, 22, 32, 23, 49, 26, 45,
 19, 1, 36, 46, 41, 18, 20, 17]
```

Thus giving a sorted list of the following e-mails: 1, 3, 8, 10, 11, 13, 17, 18, 19, 20, 21, 22, 23, 26, 27, 28, 29, 32, 34, 36, 41, 45, 46, 48, 49, 57, 63, 64, 68, 69.

5.2. Sample Output

The following pages show the results of running the completed software on a sample e-mail sent from my own Oxford e-mail account to my G-Mail account, the contents of which is listed in Fragment 5.1. The full table of CVE entries is elided for brevity. The entries in this colour are associated with points scored for R , and the entries in this colour are associated with points scored for F .

```
Delivered-To: joshuaclark94@gmail.com
Received: by 10.28.1.200 with SMTP id 191csp811296wmb;
    Sun, 22 May 2016 09:55:48 -0700 (PDT)
X-Received: by 10.28.54.204 with SMTP id y73mr6213044wmh.59.1463936148370;
    Sun, 22 May 2016 09:55:48 -0700 (PDT)
Return-Path: <joshua.clark@st-annes.ox.ac.uk>
Received: from relay14.mail.ox.ac.uk (relay14.mail.ox.ac.uk. [163.1.2.162])
    by mx.google.com with ESMTPS id mw4si38680863wjb.85.2016.05.22.09.55.48
    for <joshuaclark94@gmail.com>
    (version=TLS1_2 cipher=AES128-SHA bits=128/128);
    Sun, 22 May 2016 09:55:48 -0700 (PDT)
Received-SPF: pass (google.com: best guess record for domain of joshua.clark@st-
    annes.ox.ac.uk designates 163.1.2.162 as permitted sender) client-ip
    =163.1.2.162;
Authentication-Results: mx.google.com;
    spf=pass (google.com: best guess record for domain of joshua.clark@st-annes.
    ox.ac.uk designates 163.1.2.162 as permitted sender) smtp.mailfrom=
    joshua.clark@st-annes.ox.ac.uk
Received: from smtp4.mail.ox.ac.uk ([129.67.1.207])
    by relay14.mail.ox.ac.uk with esmt (Exim 4.80)
    (envelope-from <joshua.clark@st-annes.ox.ac.uk>) id 1b4WfM-000315-jc
    for joshuaclark94@gmail.com; Sun, 22 May 2016 17:55:48 +0100
Received: from chap05.pmb.ox.ac.uk ([129.67.89.5])
    by smtp4.mail.ox.ac.uk with esmt (Exim 4.80)
    (envelope-from <joshua.clark@st-annes.ox.ac.uk>) id 1b4WfM-0008Mz-D6
    for joshuaclark94@gmail.com; Sun, 22 May 2016 17:55:48 +0100
X-Oxford-Username: sann4425
To: joshuaclark94@gmail.com
From: Joshua Clark <joshua.clark@st-annes.ox.ac.uk>
Subject: Test
Message-ID: <0d2bc153-4b20-8f0e-e137-804305c28527@st-annes.ox.ac.uk>
Date: Sun, 22 May 2016 18:55:16 +0100
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101
    Thunderbird/45.1.0
MIME-Version: 1.0
Content-Type: text/plain; charset=utf-8; format=flowed
Content-Transfer-Encoding: 7bit
```

Test

E-Mail Header Fragment 5.1: Sample E-Mail

E-Mail Header Information Results

Sender Information

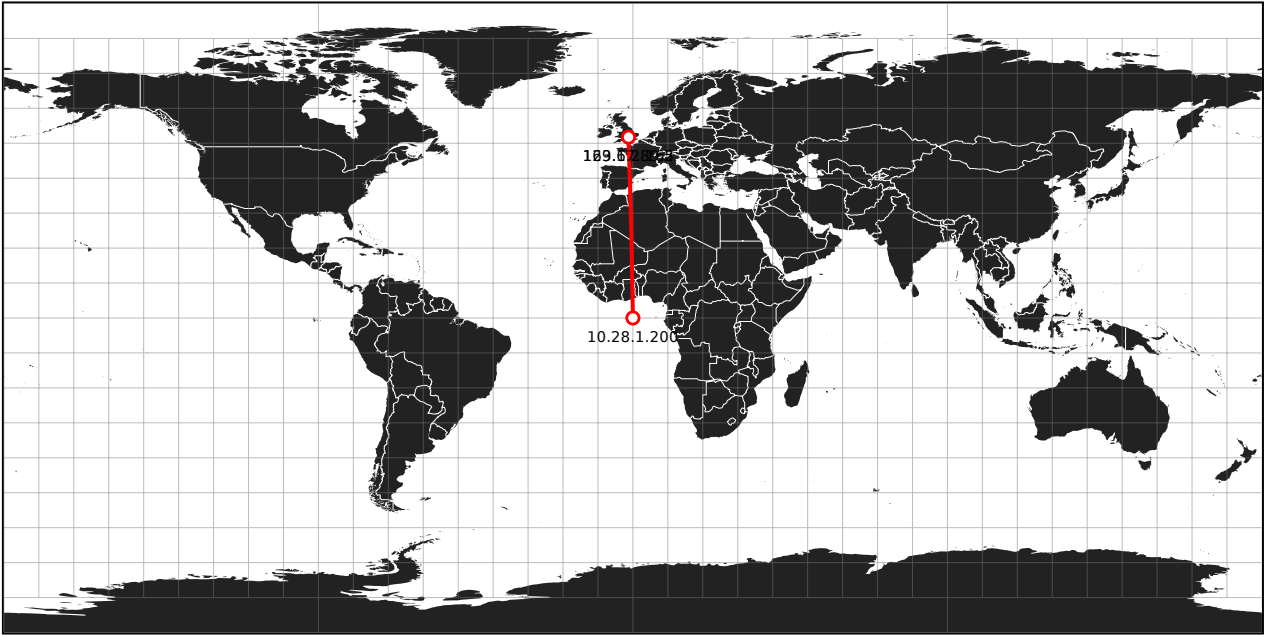
Name: Joshua Clark
Organisation: NOT FOUND

Sender Software

Software: Thunderbird

Found usernames

- Oxford
sann4425

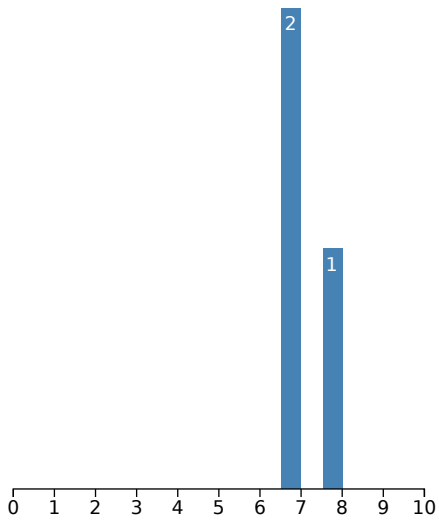


Class	Type	Details
Username	Username	sann4425
Application	Thunderbird	User-Agent
Personal	Sender Information	Joshua Clark

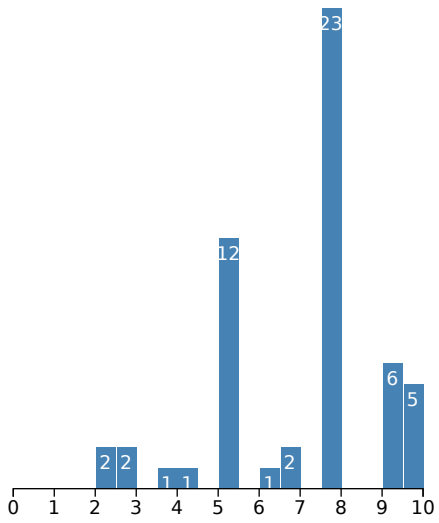
	Server	Time	Software	CVEs
0	10.28.1.200 (0,0)		SMTP	
1	(0,0)		ESMTPS	
2	163.1.2.162 (51.75,-1.25)		esmtplib (Exim 4.80) (envelope-from)	
3	129.67.89.5 (51.75,-1.25)		esmtplib (Exim 4.80) (envelope-from)	

Distribution of CVE Scores by Product

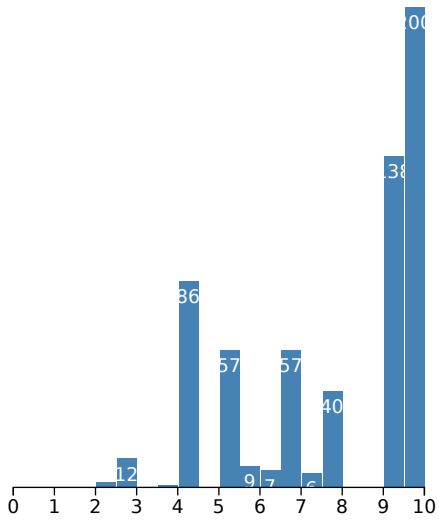
esmtplib



smtp



thunderbird



Availability: Confidentiality: Integrity: Vector: Complexity: Authentication:

CVE-ID	Summary
CVE-2002-	The JavaScript implementation in Mozilla Firefox before 4.0, Thunderbird before 3.3, and SeaMonkey before 2.1 does not properly restrict the set of values contained in the object returned by the getComputedStyle method, which allows remote attackers to obtain

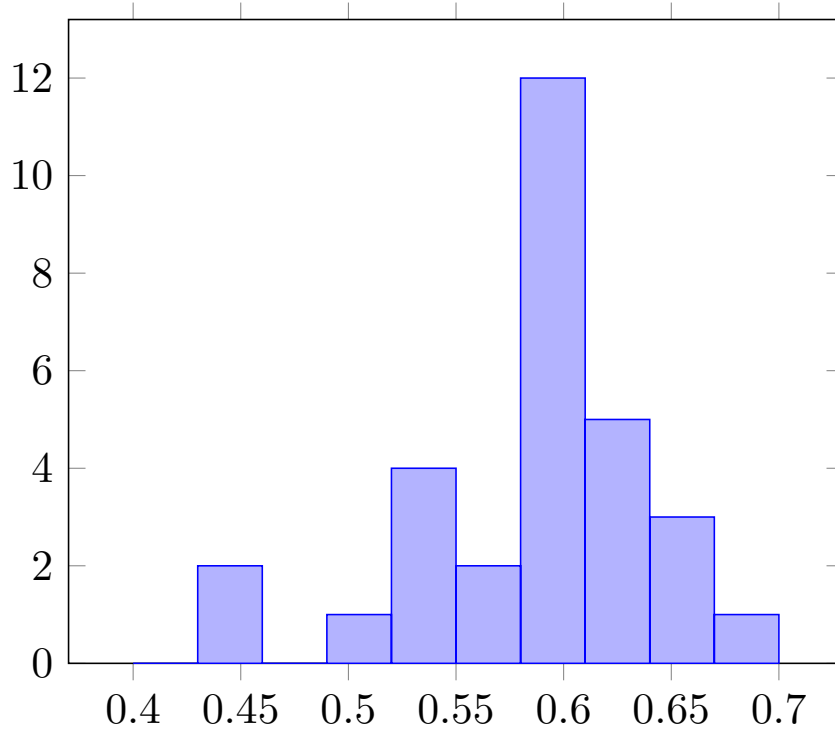


Figure 9.: Distribution of scores from e-mails

The servers highlighted in red, with the exception of the field labelled **X-Received** have all be parsed properly, and their details collected.

No IP address is available for the `mx.google.com` server entry, and therefore no location data can be found, with later locations then being internal addresses, which also have no available location data. However, all of the Oxford based entries have correctly been identified as shown on the map. Due to the limitations of the location data provided, it has not been possible to position them accurately relative to each other.

All the available software entries have been correctly detected (SMTP, ESMTP and Thunderbird), and CVE data has been found for these pieces of software.

Lastly, my Oxford username has been correctly identified within the e-mail.

5.3. Results

Table 2 gives the M and N values for the different sampled e-mails. These were sampled using standard Unix tools: `grep` for fields and counting the output. The results for R and F have been listed after completing the testing, after counting the number of entries in the final visualisation.

The final score for the e-mails gives a minimum value of 0.449 and a maximum value of 0.680, with an average of 0.583 and standard deviation of 0.0540. The histogram in Figure 9 shows the distribution of scores from the e-mails, showing a skew towards higher scores. This is a promising result, however, most of the score is contributed to by the trace-fields, with the value of $R/4N$ being consistently around 0.85. As the aim of this project has both

Header	Total Fields	M	N	R	F	S
1.txt	26	7	19	23	7	0.594924812
3.txt	32	6	26	16	6	0.4487179487
8.txt	34	8	26	30	6	0.5841346154
10.txt	22	3	19	9	7	0.5592105263
11.txt	29	6	23	23	6	0.6096014493
13.txt	17	4	13	13	5	0.5985576923
17.txt	27	6	21	22	6	0.6011904762
18.txt	20	6	14	23	5	0.6577380952
19.txt	20	6	14	21	4	0.5803571429
20.txt	22	7	15	25	7	0.6797619048
21.txt	22	6	16	23	5	0.6354166667
22.txt	22	6	16	22	6	0.6458333333
23.txt	26	6	20	20	7	0.5373563218
26.txt	19	6	13	21	5	0.6298076923
27.txt	21	1	20	2	8	0.45
28.txt	22	6	16	23	4	0.6041666667
29.txt	20	6	14	21	6	0.6517857143
32.txt	24	6	18	20	3	0.5
34.txt	23	5	18	19	4	0.586111111
36.txt	26	6	20	21	4	0.5375
41.txt	21	6	15	20	5	0.5833333333
45.txt	23	6	17	22	5	0.6053921569
46.txt	26	5	21	15	7	0.5416666667
48.txt	25	6	19	22	6	0.6162280702
49.txt	21	6	15	22	3	0.5583333333
57.txt	28	6	22	21	4	0.5284090909
63.txt	23	5	18	18	5	0.5888888889
64.txt	36	9	27	31	6	0.5416666667
68.txt	21	6	15	20	5	0.5833333333
69.txt	22	6	16	21	5	0.59375
<i>Average</i>	24.3	5.8	18.5	20.3	5.4	0.582916135

Table 2.: M , N , R , F and score values for chosen headers

focused on the disclosure of individual's data and of network vulnerabilities, the high score from the trace-fields is a positive result.

This would also imply that the other fields contributed a score of around 0.3, giving an estimated 3 pieces of information for every ten lines of header, also a promising result. Especially considering that the score for the information gathered from fields does not take into account the number of fields needed to determine or infer a piece of information. For example, the presence, or absence of multiple fields is required to determine a piece of information. Nor does it consider the relative value of a piece of information, failing to rate the presence of a username above the presence of a particular piece of software also giving false positives.

During the testing, the following trends were noticed. As many of the e-mails passed through the same set of servers, as they had been received by an Oxford e-mail address, the same set of servers were frequently seen, all of which had associated IP addresses, geolocation data and (except for one server running Microsoft SMTP Server) CVE data for the running software. For an e-mail sent within the University Nexus system, this gives an inflated score, as very few of the servers are missing information.

However, very few e-mails in the testing population contained fields relating to usernames (for example, `X-...-User`, `X-Oxford-Username`, `X-Authenticated-User`) compared to the result in Nurse et al. 2015, which found 14% of e-mails to contain usernames as opposed to the 8% found in the population.

The data that could be extracted from CVEs was notably frequently missing an entry for the Microsoft SMTP Server. Given its freely within Microsoft server packages, its absence would be surprising, however, vulnerabilities for this piece of software were later found to refer to Microsoft Windows Server editions, rather than the SMTP Server software itself, making extracting relevant information even more difficult.

6. Conclusions

6.1. Summary

As described in Chapter 1, the aim of this project is to support a better understanding of the data that may leaked when e-mails are sent, both from a personal perspective, as well as the corporate data that is leaked concern network configurations and software installations.

To support this, I have developed a tool that can be used to automatically extract information from e-mail headers and analyse its results to display the personal information contained within an e-mail's header, as well as information about the software configurations that may be found on a user's computer, or the servers used to send their e-mail.

This tool has performed well, particularly in extracting data from e-mail headers in connection to the trace-fields with a score of 0.85 for the trace-fields. The lower rate of information that has been extracted from the other fields, while less impressive, is still promising, given that all fields in an e-mail do not necessarily provide information about the sender themselves, for example, the subject, CC, and BCC fields.

However, as promising a result as this is, it does come with the caveat: more data is being leaked from e-mails than would have been previously believed, and given that limited awareness of the scope and sensitivity of the data that is released, it is not unreasonable to expect that attacks focusing on the data leaked in e-mails become more prevalent in the future. These attacks may focus on individuals, based on the personal data that is leaked by individuals, or on corporations, as details their internal software and network configurations become more accessible.

6.2. Future Work

During the late stages of development and testing, a number of missing features quickly became apparent. Due to the limited information available, and the differences in version numbering, a decision was made to search for all available vulnerabilities for an application, allowing the user to discern which were most relevant. Subsequent versions could focus on the different pieces of version data available. For example, `esmtplib` frequently references its version number in the "Received" field frequently.

Alternatively, a better picture may be presented by accumulating multiple e-mails. For example, using the information provided from multiple members of single organisation, a better picture may be built up of the software used by the servers, as well as the network configuration.

The application's response times may also be improved by caching some data in memory, such as WhoIs responses and GeoIP lookups, so that frequently accessed lookups can be completed more quickly.

Additionally, future testing should take place on a larger dataset, using e-mails from a wider variety of sources sent to a number of different recipients.

Finally, it should be possible for an updated version of this application to determine which header fields have been added by mail servers within one's own organisation. For example, e-mail header fields beginning with **X-MS-Exchange** are seen within almost all e-mail messages sent to recipients within the Oxford domain, adding more false positives to the test results. While it is possible that other preceding e-mail servers have added similar fields, in most cases, these entries yielded little useful data.

Bibliography

- [1] Rakesh Agrawal et al. “Order preserving encryption for numeric data”. In: *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. ACM. 2004, pp. 563–574.
- [2] D. Crocker. *STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES*. STD 11. RFC Editor, Aug. 1982.
- [3] Jerry Felix and Chris Hauck. “System security: a hacker’s perspective”. In: *Interex Proceedings* 1 (1987), pp. 6–6.
- [4] Danesh Irani et al. “Modeling unintended personal-information leakage from multiple online social networks”. In: *Internet Computing, IEEE* 15.3 (2011), pp. 13–19.
- [5] Akanksha Joshi, Ravendar Lal, and Tim Finin. “Extracting cybersecurity related linked data from text”. In: *Semantic Computing (ICSC), 2013 IEEE Seventh International Conference on*. IEEE. 2013, pp. 252–259.
- [6] Jason RC Nurse et al. “Investigating the leakage of sensitive personal and organisational information in email headers”. In: *Journal of Internet Services and Information Security (JISIS)* 5.1 (2015), pp. 70–84.
- [7] Panagiotis Papadimitriou and Hector Garcia-Molina. “Data leakage detection”. In: *Knowledge and Data Engineering, IEEE Transactions on* 23.1 (2011), pp. 51–63.
- [8] PHP Group et al. *PHP: Character classes - Manual*. URL: <https://secure.php.net/manual/en/regexp.reference.character-classes.php>.
- [9] Anna Squicciarini, Smitha Sundareswaran, and Dan Lin. “Preventing information leakage from indexing in the cloud”. In: *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*. IEEE. 2010, pp. 188–195.
- [10] Brad Templeton. *Reaction to the DEC Spam of 1978*. URL: <http://www.templetons.com/brad/spamreact.html>.
- [11] The Identity Theft Resource Center. May 2016. URL: <http://www.idtheftcenter.org/images/breach/ITRCBreachStatsReportSummary2016.pdf>.
- [12] Marwan Al-zarouni. *Tracing E-mail Headers*. 2004.

A. Code Listings

```

    useful, but WITHOUT ANY WARRANTY; without
    * even the implied warranty of MERCHANTABILITY or FITNESS
      FOR A PARTICULAR PURPOSE. See the GNU
    * Lesser Public License for more details.
    * <p>
    * You should have received a copy of the GNU Lesser Public
      License along with this program. If not,
    * see <http://www.gnu.org/licenses/>.
    * <p>
    * Please see the following page for the LGPL license: http
      ://www.gnu.org/licenses/lgpl.txt
    */
    public class SystemCommandExecutor {
    private List<String> commandInformation;
    private String adminPassword;
    private ThreadedStreamHandler inputStreamHandler;
    private ThreadedStreamHandler errorStreamHandler;

    /**
    * Pass in the system command you want to run as a List
    * of Strings, as shown here:
    * <p>
    * List<String> commands = new ArrayList<String>();
    *   commands.add("/sbin/ping");
    *   commands.add("-c"); commands.add("5"); commands.add("
    *   www.google.com"); SystemCommandExecutor
    *   commandExecutor = new SystemCommandExecutor(commands);
    *   commandExecutor.executeCommand();
    * <p>
    * Note: I've removed the other constructor that was here
    *   to support executing the sudo command.
    * I'll add that back in when I get the sudo command
    *   working to the point where it won't hang
    *   when
    * the given password is wrong.
    *
    package com.devdaily.system;

    import java.io.IOException;
    import java.io.InputStream;
    import java.io.OutputStream;
    import java.util.List;

    /**
    * This class can be used to execute a system command from
    *   a Java application. See the documentation
    * for the public methods of this class for more
    *   information.
    * <p>
    * Documentation for this class is available at this URL:
    * <p>
    * http://devdaily.com/java/java-processbuilder-process-
    *   system-exec
    * <p>
    * <p>
    * Copyright 2010 alvin j. alexander, devdaily.com.
    * <p>
    * This program is free software: you can redistribute it
    *   and/or modify it under the terms of the
    *   GNU Lesser Public License as published by the Free
    *   Software Foundation, either version 3 of the
    *   License, or (at your option) any later version.
    * <p>
    * This program is distributed in the hope that it will be

```

```

* @param commandInformation
*   The command you want to run.
*/
public SystemCommandExecutor (final List<String>
    commandInformation) {
    if (commandInformation == null) {
        throw new NullPointerException( "The command information
            is required.");
    }
    this.commandInformation = commandInformation;
    this.adminPassword = null;

    public int executeCommand () throws IOException,
        InterruptedException {
            int exitValue = -99;

            try {
                ProcessBuilder pb = new ProcessBuilder(
                    commandInformation);
                Process process = pb.start();

                // you need this if you're going to write something to
                // the command's input
                // stream
                // (such as when invoking the 'sudo' command, and it
                // prompts you for a
                // password).
                OutputStream stdout = process.getOutputStream();

                // i'm currently doing these on a separate line here
                // in case i need to set
                // them to null
                // to get the threads to stop.
                // see
                // http://java.sun.com/j2se/1.5.0/docs/guide/misc/

                threadPrimitiveDeprecation.html
                InputStream inputStream = process.getInputStream();
                InputStream errorStream = process.getErrorStream();

                // these need to run as java threads to get the
                // standard output and error
                // from the command.
                // the inputStream handler gets a reference to our
                // stdout in case we
                // need to write
                // something to it, such as with the sudo command
                inputStreamHandler = new ThreadedStreamHandler(
                    inputStream, stdout, adminPassword);
                errorStreamHandler = new ThreadedStreamHandler(
                    errorStream);

                // TODO the inputStreamHandler has a nasty side-effect
                // of hanging if the
                // given password is wrong; fix it
                inputStreamHandler.start();
                errorStreamHandler.start();

                // TODO a better way to do this?
                exitValue = process.waitFor();

                // TODO a better way to do this?
                inputStreamHandler.interrupt();
                errorStreamHandler.interrupt();
                inputStreamHandler.join();
                errorStreamHandler.join();
            } catch (IOException e) {
                // TODO deal with this here, or just throw it?
                throw e;
            } catch (InterruptedException e) {
                // generated by process.waitFor() call
                // TODO deal with this here, or just throw it?

```

```

        throw e;
    } finally {
        return exitValue;
    }
}

/**
 * Get the standard output (stdout) from the command you
 * just exec'd.
 */
public StringBuilder getStandardOutputFromCommand () {
    return inputStreamHandler.getOutputBuffer();
}

/**
 * Get the standard error (stderr) from the command you
 * just exec'd.
 */
public StringBuilder getStandardErrorFromCommand () {
    return errorStreamHandler.getOutputBuffer();
}

}

package com.devdaily.system;

import java.io.*;

/**
 * This class is intended to be used with the
 * SystemCommandExecutor class to let users execute
 * system commands from Java applications.
 *
 * <p>
 * This class is based on work that was shared in a
 * JavaWorld article named "When System.exec()
 * won't". That article is available at this url:

```

```

* <p>
* http://www.javaworld.com/javaworld/jw-12-2000/jw-1229-
  traps.html
* <p>
* Documentation for this class is available at this URL:
* <p>
* http://devdaily.com/java/java-processbuilder-process-
  system-exec
* <p>
* <p>
* Copyright 2010 alvin j. alexander, devdaily.com.
* <p>
* This program is free software: you can redistribute it
  and/or modify it under the terms of the
  GNU Lesser Public License as published by the Free
  Software Foundation, either version 3 of the
  License, or (at your option) any later version.
* <p>
* This program is distributed in the hope that it will be
  useful, but WITHOUT ANY WARRANTY; without
  even the implied warranty of MERCHANTABILITY or FITNESS
  FOR A PARTICULAR PURPOSE. See the GNU
  Lesser Public License for more details.
* <p>
* You should have received a copy of the GNU Lesser Public
  License along with this program. If not,
  see <http://www.gnu.org/licenses/>.
* <p>
* Please see the following page for the LGPL license: http
  ://www.gnu.org/licenses/lgpl.txt
*/
class ThreadedStreamHandler extends Thread {
    InputStream inputStream;
    String adminPassword;
    OutputStream outputStream;
    PrintWriter printWriter;

```

```

StringBuffer outputBuffer = new StringBuffer();
private boolean sudoIsRequested = false;

/**
 * A simple constructor for when the sudo command is not
 * necessary. This constructor will just
 * run
 * the command you provide, without running sudo before
 * the command, and without expecting a
 * password.
 *
 * @param inputStream
 * @param streamType
 */
ThreadedStreamHandler (InputStream inputStream) {
    this.inputStream = inputStream;
}

/**
 * Use this constructor when you want to invoke the 'sudo'
 * command. The outputStream must not be
 * null. If it is, you'll regret it. :)
 *
 * <p>
 * TODO this currently hangs if the admin password given
 * for the sudo command is wrong.
 *
 * @param inputStream
 * @param streamType
 * @param outputStream
 * @param adminPassword
 */
ThreadedStreamHandler (InputStream inputStream,
    OutputStream outputStream, String adminPassword) {
    this.inputStream = inputStream;
    this.outputStream = outputStream;
    this.printWriter = new PrintWriter(outputStream);
    this.adminPassword = adminPassword;
    this.sudoIsRequested = true;
}

public void run () {
    // on mac os x 10.5.x, when i run a 'sudo' command, i
    // need to write
    // the admin password out immediately; that's why this
    // code is
    // here.
    if (sudoIsRequested) {
        // doSleep(500);
        printWriter.println(adminPassword);
        printWriter.flush();
    }

    BufferedReader bufferedReader = null;
    try {
        bufferedReader = new BufferedReader(new
            InputStreamReader(inputStream));
        String line = null;
        while ((line = bufferedReader.readLine()) != null) {
            outputBuffer.append(line + "\n");
        }
    } catch (IOException ioe) {
        // TODO handle this better
        ioe.printStackTrace();
    } catch (Throwable t) {
        // TODO handle this better
        t.printStackTrace();
    } finally {
        try {
            bufferedReader.close();
        } catch (IOException e) {
            // ignore this one
        }
    }
}

```

```

    }
}

private void doSleep (long millis) {
    try {
        Thread.sleep(millis);
    } catch (InterruptedException e) {
        // ignore
    }
}

public StringBuilder getOutputBuffer () {
    return outputBuffer;
}

}

package emailheaderinformation.analysers;

import emailheaderinformation.MainWindow;
import emailheaderinformation.model.Header;

import java.util.Collections;
import java.util.HashSet;
import java.util.stream.Collectors;

/**
 * Uses results found in http://isyou.info/jisis/vol5/vol5/no1/jisis-2015-vol5-no1-04.pdf
 */
public class ClientInferer extends HeaderAnalyser {
    private final Header mHeader;
    private final MainWindow mMainWindow;

    private final HashSet<String> outlookAppleKeywords;

    public ClientInferer (Header header, MainWindow
        mainWindow) {
        super(header, mainWindow);
        mHeader = header;
        mMainWindow = mainWindow;

        String[] kws = { "Accept-Language", "Content-Language",
            "Threat-Index", "Threat-Topic" };
        outlookAppleKeywords = new HashSet<>();

        Collections.addAll(outlookAppleKeywords, kws);
    }

    @Override public Object call () throws Exception {
        run();
        return null;
    }

    private void run () {
        String product = null;
        String hint = null;
        String lookup = null;
        String messageId = mHeader.getFields().get("Message-ID")
            ;
        if (messageId != null && !messageId.isEmpty() &&
            messageId.contains("email. android.com")) {
            product = "Android_E-Mail";
            hint = "Message-ID";
            lookup = "android";
        } else if (mHeader.getFields().containsKey("X-Mailer"))
            {
                String field = mHeader.getFields().get("X-Mailer");
                hint = "X-Mailer";
                if (field.contains("iPhone")) {
                    product = "Apple_iOS_Mail";
                    lookup = "apple:iphone";
                }
            }
        }
    }
}

```

```

    } else if (field.contains("Outlook_Express")){
        product = "Microsoft_Outlook_Express";
        lookup = "microsoft:outlook_express";
    } else if (field.toLowerCase().contains("php")) {
        product = "PHPMailer";
        lookup = "phpmailer";
    } else if (field.contains("Foamail")) {
        product = "Foamail";
        lookup = "foamail";
    }
} else if (mHeader.getFields().containsKey("User-Agent"))
    ) {
    // Presence of User-Agent string implies Thunderbird
    product = "Thunderbird";
    hint = "User-Agent";
    lookup = "thunderbird";
} else if (!mHeader.getFields()
    .keySet()
    .stream()
    .filter(outlookAppleKeywords::contains)
    .collect(Collectors.toList())
    .isEmpty()) {
    // Presence of keys from outlookAppleKeywords implies
    // Apple Mail
    // or Outlook
    if (mHeader.getFields().containsKey("X-Mailer")) {
        // If X-Mailer is seen, then more likely to have
        // been sent by
        // Apple Mail
        product = "AppleMail";
        hint = "X-Mailer";
        lookup = "apple:mail";
    } else {
        // Otherwise, more likely to be Outlook
        product = "Microsoft_Outlook";
        hint = "Accept-Language";
    }
}

        lookup = "outlook";
    }
}

    if (product != null) {
        Object[] arr = { "Application", product, "Inferred",
            hint };
        mMainWindow.addToTable(arr);
        mMainWindow.getVfm().lookupVulnerabilityForKeyword(
            lookup);
        mMainWindow.getFoundInformation().setSoftware(product)
            ;
    }
}

package emailheaderinformation.analysers;

import emailheaderinformation.MainWindow;
import emailheaderinformation.model.Device;
import emailheaderinformation.model.Header;

import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.concurrent.ExecutionException;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import static emailheaderinformation.analysers.
    GeoIPAnalyser.NO_RESULT_FOUND;
import static java.lang.Float.parseFloat;

public class DeviceAnalyser extends HeaderAnalyser {

    private Pattern pattern;
    private Pattern hostname;

```



```

        e.printStackTrace();
    }

    return null;
}

private void extractInformation (String devName, Device
device)
    throws InterruptedException, java.util.concurrent.
        ExecutionException, UnknownHostException {
    // Regex to match IP addresses, with some bounds
    // checking
    String ip = "";
    float lat = 0;
    float lon = 0;
    String owner = "";
    Matcher matcher = pattern.matcher(devName);
    while (matcher.find() && lat == 0 && lon == 0) {
        String tempIp = matcher.group();
        GeoIPAnalyser geoIPAnalyser = new GeoIPAnalyser(
            mHeader, mMainWindow, tempIp);
        String result = (String) mMainWindow.
            submitToExecutorService(geoIPAnalyser).get();
        // Allow for local IP addresses being found
        if (!INO_RESULT_FOUND.equals(result)) {
            ip = matcher.group();
            lat = parseFloat(result.split(",")[0]);
            device.setLatitude(lat);
            lon = parseFloat(result.split(",")[1]);
            device.setLongitude(lon);
        }
    }

    // Coincidentally, just above the middle of the Atlantic
    // . No servers here
    try {

```

```

        public DeviceAnalyser (Header header, MainWindow
        mainWindow) {
            super(header, mainWindow);
        }

        @Override public Device call () throws ExecutionException
        {
            Device device = mHeader.getStartDevice();

            pattern = Pattern.compile("
                ([0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3})"
            );
            String validHostnameRegex = "([a-zA-Z0-9]|[a-zA-Z0-9][a-
                zA-Z0-9\\-]*[a-zA-Z0-9])\\.?" +
                "[A-Za-z0-9][A-Za-z0-9\\-]*[A-
                Za-z0-9]";
            hostname = Pattern.compile(validHostnameRegex);

            while (device.getNext() != null) {
                try {
                    String devName = device.getName();
                    extractInformation(devName, device);
                } catch (InterruptedException | ExecutionException |
                    UnknownHostException e) {
                    e.printStackTrace();
                }

                device = device.getNext();
            }

            // Special case for the last device
            try {
                extractInformation(device.getOrigin(), device);
            } catch (InterruptedException | UnknownHostException e)
            {

```

```

Matcher hostMatcher = hostname.matcher(devName);
if (hostMatcher.find()) {
    String host = hostMatcher.group(0);
    if (lat == 0 && lon == 0) {
        InetAddress inetAddress = InetAddress.getByName(
            host);
        GeoIPAnalyser geoIPAnalyser = new GeoIPAnalyser(
            mHeader,
            mMainWind, emailheaderinformation.MainWindow;
            mMainWind, emailheaderinformation.model.Header;
            inetAddress, java.util.Set;
            .
            .
            getHost, public class ExchangeHeaderAnalyser extends HeaderAnalyser
                {
                ;
                String result = (String) mMainWindow.
                    submitToExecutorService(geoIPAnalyser).get();
                // Allow for local IP addresses being found
                if (!NO_RESULT_FOUND.equals(result)) {
                    ip = inetAddress.getHostAddress();
                    lat = parseFloat(result.split(",")[0]);
                    device.setLatitude(lat);
                    lon = parseFloat(result.split(",")[1]);
                    device.setLongitude(lon);
                }
            }
            WhoIsAnalyser wia = new WhoIsAnalyser(mHeader,
                mMainWindow, host);
            owner = (String) mMainWindow.submitToExecutorService
                (wia).get();
        } catch (UnknownHostException ignored) {
        }
    }
    mMainWindow.getVfm().lookupVulnerabilityForKeyword(
        device.getSoftware());
    package emailheaderinformation analysers;
    import com.maxmind.geoip2.DatabaseReader;
    mMainWindow.getFoundInformation().addDevice(ip, lat, lon
        , device.getSoftware(), owner);
    }
}
package emailheaderinformation analysers;
import emailheaderinformation.MainWindow;
import emailheaderinformation.model.Header;
mMainWind, emailheaderinformation.model.Header;
inetAddress, java.util.Set;
.
.
getHost, public class ExchangeHeaderAnalyser extends HeaderAnalyser
    {
    ;
    public ExchangeHeaderAnalyser (Header header, MainWindow
        mainWindow) {
        super(header, mainWindow);
    }
    @Override public Object call () throws Exception {
        Set<String> keySet = mHeader.getFields().keySet();
        for (String key : keySet) {
            if (key.startsWith("X-MS-Exchange")) {
                Object[] arr = { "Exchange", key, "Yes", mHeader.
                    getFields().get(key) };
                mMainWindow.addToTable(arr);
            }
        }
        return null;
    }
}
package emailheaderinformation analysers;
import com.maxmind.geoip2.DatabaseReader;

```

```

import com.maxmind.geoip2.DatabaseReader.Builder;
import com.maxmind.geoip2.exception.
    AddressNotFoundException;
import com.maxmind.geoip2.exception.GeoIp2Exception;
import com.maxmind.geoip2.model.CityResponse;
import emailheaderinformation.MainWindow;
import emailheaderinformation.model.Header;

import java.io.File;
import java.io.IOException;
import java.net.InetAddress;

class GeoIPAnalyser extends HeaderAnalyser<String> {

    static final String NO_RESULT_FOUND = "NO_RESULT_FOUND";
    private final String mAddress;

    GeoIPAnalyser (Header header, MainWindow mainWindow,
        String ipAddress) {
        super(header, mainWindow);
        mAddress = ipAddress;
    }

    @Override public String call () {
        Builder builder = new Builder(new File(GeoIPAnalyser.
            class.getResource("/GeoLite2-City.mmdb")
                .getFile()));
        try (DatabaseReader dbr = builder.build()) {
            CityResponse city = dbr.city(InetAddress.getByName(
                mAddress));
            return String.format("%s,%s",
                city.getLocation().getLatitude(),
                city.getLocation().getLongitude());
        } catch (AddressNotFoundException anfe) {
            return NO_RESULT_FOUND;
        } catch (IOException | GeoIp2Exception e) {
            ;
        }
    }
}

import com.maxmind.geoip2.DatabaseReader.Builder;
import com.maxmind.geoip2.exception.
    AddressNotFoundException;
import com.maxmind.geoip2.exception.GeoIp2Exception;
import com.maxmind.geoip2.model.CityResponse;
import emailheaderinformation.MainWindow;
import emailheaderinformation.model.Header;

import java.io.File;
import java.io.IOException;
import java.net.InetAddress;

package emailheaderinformation.analysers;

import emailheaderinformation.MainWindow;
import emailheaderinformation.model.Header;

import java.util.concurrent.Callable;

/**
 * @author joshua
 */
abstract public class HeaderAnalyser<V> implements Callable
    <V> {
    final Header mHeader;
    final MainWindow mMainWindow;

    public HeaderAnalyser (Header header, MainWindow
        mainWindow) {
        mHeader = header;
        mMainWindow = mainWindow;
    }

    package emailheaderinformation.analysers;

    import emailheaderinformation.MainWindow;
    import emailheaderinformation.model.VulnerabilityDisclosure
        ;
}

```

```

import java.util.*;

import javax.mail.internet.*;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import java.util.concurrent.ExecutionException;

public class SenderInformationExtractor extends
    HeaderAnalyser {

    public SenderInformationExtractor (Header header,
        MainWindow mainWindow) {
        super(header, mainWindow);
    }

    @Override public Object call () {
        String fromField = mHeader.getFields().get("From");
        int emailStart = fromField.indexOf("<");
        int emailEnd = fromField.indexOf(">");
        mMainWindow.getFoundInformation().setName(fromField.
            substring(0, emailStart - 1));
        Object[] arr = {
            "Personal", "Sender_Information", "Yes", fromField.
                substring(0, emailStart - 1) };
        mMainWindow.addToTable(arr);
        String email = fromField.substring(emailStart+1,
            emailEnd);
        System.out.println(email);
        if (isValidEmailAddress(email)) {
            String domain = email.split("@")[1];
            WhoIsAnalyser wia = new WhoIsAnalyser(mHeader,
                mMainWindow, domain);
            String result = "";
            try {
                System.out.println("WIA_Lookup_started");
                result = (String) mMainWindow.
                    submitToExecutorService(wia).get();
            }
        }
    }
}

package emailheaderinformation analysers;
import emailheaderinformation.MainWindow;

import java.util.*;

public class NoopVulnerabilityFinderManager implements
    VulnerabilityFinderManager {
    public NoopVulnerabilityFinderManager (MainWindow mainWindow)
    {
    }

    @Override public Set<String> getKeywords () {
        return Collections.emptySet();
    }

    @Override public Collection<VulnerabilityDisclosure>
        getVulnerabilities () {
        return Collections.emptySet();
    }

    @Override public Map<String, List<VulnerabilityDisclosure>
        >> getMostRelevantVulnerabilities () {
        return Collections.emptyMap();
    }

    @Override public boolean noVulnerabilitiesFound () {
        return false;
    }

    @Override public void lookupVulnerabilityForKeyword (
        String keyword) {
        // NO-OP
    }
}

package emailheaderinformation analysers;
import emailheaderinformation.MainWindow;

```

```

        System.out.println("WIA_Lookup_ended");
    } catch (InterruptedException | ExecutionException e)
    {
        e.printStackTrace();
    }
    if (result.isEmpty()) {
        mMainWindow.getFoundInformation().setOrganisation("
        Not_found");
    } else {
        mMainWindow.getFoundInformation().setOrganisation(
            result);
    }
}

return null;
}

private static boolean isValidEmailAddress (String email)
{
    boolean result = true;
    try {
        InetAddress emailAddr = new InetAddress(email)
        ;
        emailAddr.validate();
    } catch (AddressException ex) {
        result = false;
    }
    return result;
}

package emailheaderinformation analysers;

import com.google.gson.Gson;
import com.google.gson.JsonObject;
import emailheaderinformation.MainWindow;
import emailheaderinformation.database.DbManager;
import emailheaderinformation.model.*;

import java.text.DateFormat;
import java.text.ParseException;

```

```

public class UsernameHeaderAnalyser extends HeaderAnalyser
{
    public UsernameHeaderAnalyser (Header header, MainWindow
    mainWindow) {
        super(header, mainWindow);
    }

    @Override public Object call () {
        String[] kws = {"X-Oxford-Username", "X-Username", "X-
        Authenticated-User"};
        for (String kw : kws) {
            if (mHeader.getFields().containsKey(kw)) {
                Object[] arr = {
                    "Username", "Username", "Yes", mHeader.getFields
                    ().get(kw) };
                mMainWindow.addToTable(arr);
                mMainWindow.getFoundInformation().addUsername("
                Oxford", mHeader.getFields().get(kw));
            }
            return null;
        }
    }

    package emailheaderinformation analysers;

    import com.google.gson.Gson;
    import com.google.gson.JsonObject;
    import emailheaderinformation.MainWindow;
    import emailheaderinformation.database.DbManager;
    import emailheaderinformation.model.*;

    import java.text.DateFormat;
    import java.text.ParseException;

```



```

        .setIntegrity(Value.valueOf(
            impactJson.get("integrity").
                getAsString()))
            .createImpact();
    }

    DateFormat df2 = new SimpleDateFormat("yyyy-MM-dd'T'HH
        :mm:ss.SSSXX");
    Date date = null;
    try {
        date = df2.parse(jsonObject.get("Published").
            getAsString());
    } catch (ParseException e) {
        e.printStackTrace();
    } finally {
        date = date == null ? new Date() : date;
    }

    VulnerabilityDisclosureBuilder vdb = new
        VulnerabilityDisclosureBuilder();
    VulnerabilityDisclosure vd = vdb.setAccess(access)
        .setImpact(impact)
        .setProduct(mKeyword)
        .setDate(date)
        .setCveId(jsonObject.get
            ("id").getAsString()
        )
        .setScore(jsonObject.get
            ("cvss").getAsFloat
            ())
        .setSummary(jsonObject.
            get("summary").
            getAsString())
        .
        createVulnerabilityDisc
            ();

    mVulnerabilityDisclosureSet.put(date, vd);
    });
    return null;
}
}

package emailheaderinformation analysers;

import emailheaderinformation.model.VulnerabilityDisclosure
    ;

import java.util.Collection;
import java.util.List;
import java.util.Map;
import java.util.Set;

public interface VulnerabilityFinderManager {
    Set<String> getKeywords ();

    Collection<VulnerabilityDisclosure> getVulnerabilities ();

    Map<String, List<VulnerabilityDisclosure>>
        getMostRelevantVulnerabilities ();

    boolean noVulnerabilitiesFound ();

    void lookupVulnerabilityForKeyword (String keyword);
}

package emailheaderinformation analysers;

import emailheaderinformation.MainWindow;
import emailheaderinformation.model.VulnerabilityDisclosure
    ;

```

```

import java.util.*;
import java.util.concurrent.ConcurrentHashMap;
import java.util.concurrent.ConcurrentSkipListSet;

public class VulnerabilityFinderManagerImpl implements
    VulnerabilityFinderManager {
    private Set<String> mKeywordSet;
    private MainWindow mMainWindow;
    private Map<Date, VulnerabilityDisclosure> disclosureSet;

    public VulnerabilityFinderManagerImpl (MainWindow
        mainWindow) {
        mKeywordSet = new ConcurrentSkipListSet<>();
        mMainWindow = mainWindow;
        disclosureSet = new ConcurrentHashMap<>();
    }

    @Override public Set<String> getKeywords () {
        return mKeywordSet;
    }

    @Override public Collection<VulnerabilityDisclosure>
        getVulnerabilities () {
        return disclosureSet.values();
    }

    @Override
    public Map<String, List<VulnerabilityDisclosure>>
        getMostRelevantVulnerabilities() {
        Map<String, List<VulnerabilityDisclosure>> map = new
            HashMap<>();
        for (String keyword : mKeywordSet) {
            ArrayList<VulnerabilityDisclosure>
                vulnerabilityDisclosures = new ArrayList<>();
            disclosureSet.forEach((date, disclosure) -> {
                if (disclosure.getProduct().equals(keyword)) {
                    if (vulnerabilityDisclosures.size() < 3) {
                        vulnerabilityDisclosures.add(disclosure);
                    } else {
                        VulnerabilityDisclosure oldest = null;
                        for (VulnerabilityDisclosure disclosure1 :
                            vulnerabilityDisclosures) {
                            if (oldest == null ||
                                (oldest.getDate().before(disclosure1.
                                    getDate()) && disclosure1.getDate().
                                    before(
                                        disclosure.getDate())) {
                                oldest = disclosure1;
                            }
                        }
                        if (oldest != null) {
                            vulnerabilityDisclosures.remove(oldest);
                            vulnerabilityDisclosures.add(disclosure);
                        }
                    }
                }
            });
            map.put(keyword, vulnerabilityDisclosures);
        }
        return map;
    }

    @Override public boolean noVulnerabilitiesFound () {
        return disclosureSet.isEmpty();
    }

    @Override public void lookupVulnerabilityForKeyword (
        String keyword) {
        StringBuilder sb = new StringBuilder();
        String trimmedKeyword = keyword.trim();

```



```

char[] chars = new char[trimmedKeyword.length()];
trimmedKeyword.getChars(0, trimmedKeyword.length(),
    chars, 0);
for (int i = 0, length = chars.length; i < length; i++)
{
    char c = chars[i];
    if (Character.isSpaceChar(c)) {
        sb.append(' ');
    } else if (c == '(' || c == '[' || c == '{') {
        if (Character.isSpaceChar(chars[i - 1])) {
            sb.deleteCharAt(sb.length() - 1);
        }
        break;
    } else {
        sb.append(c);
    }
}

String normalisedProduct = sb.toString().toLowerCase();

// don't repopulate
if (!mKeywordSet.contains(normalisedProduct)) {
    // prevent other processes adding the same results
    mKeywordSet.add(normalisedProduct);
    VulnerabilityAnalyser va = new VulnerabilityAnalyser(
        null,
        mMainWindow,
        'normalise',
        'disclosure');

    mMainWindow.submitToExecutorService(va);
}
}
}

package emailheaderinformation.analysers;

import emailheaderinformation.MainWindow;
import emailheaderinformation.model.Header;
import uk.bl.wa.whois.JRubyWhois;
import uk.bl.wa.whois.record.WhoisResult;

public class WhoIsAnalyser extends HeaderAnalyser {
    private final String lookup;

    public WhoIsAnalyser (Header header, MainWindow main,
        String lookup) {
        super(header, main);
        this.lookup = lookup;
    }

    @Override public String call () {
        JRubyWhois jRubyWhoIs = new JRubyWhois();
        String search = lookup;
        while (search.contains(" ")) {
            System.out.printf("WhoIs: %s%n", search);
            WhoisResult wir = jRubyWhoIs.lookup(lookup);
            String[] lines = wir.getParts().get(0).getBody().split
                ("\\n");
            for (String line : lines) {
                int index = line.indexOf("-name");
                // if found, and we've not just found the domain
                // name, which is a bit pointless
                if (index > 0 && !line.startsWith("Domain")) {
                    System.out.printf("Org: %s%n", line.substring(
                        index + 5).trim());
                    return line.substring(index + 5).trim();
                }
            }
            int nextDot = search.indexOf(".");

```

```

        search = search.substring(nextDot+1);
    }
    return "NOT_FOUND";
}

package emailheaderinformation.database;

import com.mongodb.Block;
import com.mongodb.MongoClient;
import com.mongodb.client.FindIterable;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.model.Projections;
import org.bson.Document;

import java.util.ArrayList;
import java.util.List;
import java.util.stream.Stream;

import static com.mongodb.client.model.Filters.*;

public class DbManager {
    private MongoCollection<Document> coll;

    public DbManager () {
        init();
    }

    private void init () {
        MongoClient mc = new MongoClient();
        MongoDatabase db = mc.getDatabase("cvedb");
        coll = db.getCollection("cves");
    }

    public Stream<String> findVulnerabilitiesForProduct (
        String product) {
        List<String> vulns = new ArrayList<>();
        findVulnerabilities(product).forEach((Block<? super
            Document>) document -> {
            vulns.add(document.toJson());
        });
        return vulns.parallelStream();
    }

    /**
     * Finds all the available vulnerabilities for a given
     * product that do not require local access in
     * order to execute
     *
     * @param product
     *      the name of the product to lookup
     *
     * @return an iterable collection of vulnerabilities
     */
    private FindIterable<Document> findVulnerabilities (
        String product) {
        return coll.find(and(ne("access.network", "LOCAL"),
            regex("vulnerable_configuration", product)))
            .projection(Projections.excludeId());
    }
}

package emailheaderinformation;

import com.google.gson.Gson;
import emailheaderinformation.analysers.*;
import emailheaderinformation.model.FoundInformation;
import emailheaderinformation.model.Header;
import emailheaderinformation.parser.EmailParser;

import javax.swing.*;

```

```

import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.io.*;
import java.net.URISyntaxException;
import java.nio.charset.Charset;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;
import java.util.concurrent.Callable;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Future;

import static java.util.concurrent.Executors.
    newFixedThreadPool;

public class MainWindow {

    private VulnerabilityFinderManager vfm;
    private JFrame mFrame;
    private JButton mStart;
    private String minputEmail;
    private MainWindow mSelf = this;
    private JTable mFoundInformationTable;
    private FoundInformation foundInformation;
    private ExecutorService mExecutorService;
    private WebServer mWebServer;

    /**
     * Create the application.
     */
    public MainWindow () {
        initialize();
        vfm = new VulnerabilityFinderManagerImpl(this);

```

```

        mExecutorService = newFixedThreadPool(8);
        foundInformation = new FoundInformation();
        minputEmail = "";
    }

    /**
     * Initialize the contents of the mFrame.
     */
    private void initialize () {
        mFrame = new JFrame();
        mFrame.setBounds(100, 100, 450, 300);
        mFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JPanel blo = new JPanel(new BorderLayout());

        JPanel inputFrame = new JPanel();
        JButton mOpen = new JButton();
        final JFileChooser fc = new JFileChooser();
        mOpen.setText("Open_Email_file");
        mOpen.setHorizontalAlignment(SwingConstants.LEFT);
        mOpen.addActionListener(e -> {
            if (fc.showOpenDialog(mFrame) == JFileChooser.
                APPROVE_OPTION) {
                File file = fc.getSelectedFile();
                if (file.exists() && file.isFile() && file.canRead()
                    ) {
                    Path path = Paths.get(file.toURI());
                    try {
                        List<String> stream = Files.readAllLines(path,
                            Charset.defaultCharset());
                        StringBuilder sb = new StringBuilder();
                        for (String s : stream) {
                            sb.append(s);
                            sb.append('\n');
                        }
                        minputEmail = sb.toString();
                        mStart.setEnabled(true);

```

```

    } catch (IOException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }
} else {
    JOptionPane.showMessageDialog(mFrame, "Please select a valid file");
}
});
mStart = new JButton();
mStart.setText("Parse");
mStart.setHorizontalAlignment(SwingConstants.CENTER);
mStart.setEnabled(false);
mStart.addActionListener(e -> {
    EmailParser emailParser = new EmailParser();
    Header header = emailParser.parse(mInputEmail);
    HeaderAnalyser oha = new UsernameHeaderAnalyser(header
        , mSelf);
    HeaderAnalyser eha = new ExchangeHeaderAnalyser(header
        , mSelf);
    HeaderAnalyser ci = new ClientInferer(header, mSelf);
    HeaderAnalyser si = new SenderInformationExtractor(
        header, mSelf);
    HeaderAnalyser da = new DeviceAnalyser(header, mSelf);
    Callable[] headerAnalysers = new Callable[] { oha, eha
        , ci, si, da };
    Collection<Callable<Object>> has = new ArrayList<>();
    for (Callable callable : headerAnalysers) {
        has.add(callable);
    }
    try {
        mExecutorService.invokeAll(has);
    } catch (InterruptedException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }
}

JButton mShowVulnerabilities = new JButton();
mShowVulnerabilities.setText("Show Found CVEs");
mShowVulnerabilities.setHorizontalAlignment(
    SwingConstants.RIGHT);
mShowVulnerabilities.setEnabled(true);
mShowVulnerabilities.addActionListener(actionEvent -> {
    if (vfm.noVulnerabilitiesFound()) {
        JOptionPane.showMessageDialog(mFrame, "No
            vulnerabilities found yet!");
    } else {
        try {
            File file = new File(this.getClass().getResource("
                /result-template.html").toURI());
            try (InputStream in = new FileInputStream(file);
                InputStreamReader isr = new InputStreamReader(
                    in);
                BufferedReader br = new BufferedReader(isr)) {
                StringBuilder webpage = new StringBuilder();
                br.lines().forEach((String s) -> {
                    Gson gson = new Gson();
                    if (s.contains("${name}") {
                        webpage.append(s.replace("${name}",
                            foundInformation.getName()));
                    } else if (s.contains("var cveentries=[,]"))
                        {
                            StringBuilder sb = new StringBuilder();
                            sb.append('[');
                            vfm.getVulnerabilities().forEach(vd -> {
                                sb.append(gson.toJson(vd));
                                sb.append(',');
                                sb.append(System.lineSeparator());
                            });
                        }
                    }
                });
            }
        } catch (IOException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
    }
});

```

```

sb.append('');
webPage.append(s.replace("[ ]", sb.toString()));
};
} else if (s.contains("van_factentries[ ]")) {
{
StringBuilder sb = new StringBuilder();
sb.append(' ');
foundInformation.getFacts().forEach(f -> {
sb.append(gson.toJson(f));
sb.append(', ');
sb.append(System.lineSeparator());
});
sb.append(' ');
webPage.append(s.replace("[ ]", sb.toString()));
};
} else if (s.contains("van_usernameEntries[ ]")) {
[ ];
StringBuilder sb = new StringBuilder();
sb.append(' ');
foundInformation.getUsernameList().forEach(u
-> {
sb.append(gson.toJson(u));
sb.append(', ');
sb.append(System.lineSeparator());
});
sb.append(' ');
webPage.append(s.replace("[ ]", sb.toString()));
};
} else if (s.contains("${software}")) {
webPage.append(s.replace("${software}",
foundInformation.getSoftware()));
} else if (s.contains("${org}")) {
webPage.append(s.replace("${org}",
foundInformation.getOrganisation()));
} else if (s.contains("van_products[ ]")) {
webPage.append(s.replace("[ ]", gson.toJson(
vfm.getKeywords())));
} else if (s.contains("van_servers[ ]")) {
webPage.append(s.replace("[ ]", gson.toJson(
foundInformation.getDevices())));
} else {
webPage.append(s);
}
webPage.append(System.lineSeparator());
});
Path out = Paths.get("/tmp/webpage.html");
Files.write(out, webPage.toString().getBytes());
if (mWebServer != null) {
mWebServer.stop();
}
mWebServer = new WebServer(mFrame, webPage.
toString(), 3142);
} catch (IOException e) {
e.printStackTrace();
}
} catch (URISyntaxException e) {
e.printStackTrace();
}
}
});
inputFrame.add(mOpen);
inputFrame.add(mStart);
inputFrame.add(mShowVulnerabilities);
String[] columnNames = { "Class", "Type", "Present", "
Details" };
mFoundInformationTable = new JTable(new
DefaultTableModel(columnNames, 0));
JScrollPane scrollPane = new JScrollPane(
mFoundInformationTable);

```

```

        blo.add(inputFrame, BorderLayout.PAGE_START);
        blo.add(scrollPane, BorderLayout.CENTER);
        mFrame.add(blo);
    }

    /**
     * Launch the application.
     */
    public static void main (String[] args) {
        EventQueue.invokeLater(() -> {
            try {
                MainWindow window = new MainWindow();
                window.mFrame.setVisible(true);
            } catch (Exception e) {
                e.printStackTrace();
            }
        });
    }

    public Future submitToExecutorService (Callable task) {
        return mExecutorService.submit(task);
    }

    /**
     * Add new information to the table
     */
    public synchronized void addToTable (Object[] arr) {
        foundInformation.addFact((String) arr[0], (String) arr
            [1], (String) arr[3]);
        DefaultTableModel model = (DefaultTableModel)
            mFoundInformationTable.getModel();
        model.addRow(arr);
    }

    public VulnerabilityFinderManager getVfm () {
        return vfm;
    }

    public FoundInformation getFoundInformation () {
        return foundInformation;
    }
}

package emailheaderinformation.model;

public class Access {
    private Vector vector;
    private Complexity complexity;
    private Authentication authentication;

    Access (Vector vector, Complexity complexity,
        Authentication authentication) {
        this.vector = vector;
        this.complexity = complexity;
        this.authentication = authentication;
    }

    public Vector getVector () {
        return vector;
    }

    public Complexity getComplexity () {
        return complexity;
    }

    public Authentication getAuthentication () {
        return authentication;
    }

    public enum Vector {LOCAL, ADJACENT_NETWORK, NETWORK;}

    public enum Complexity {HIGH, MEDIUM, LOW;}
}

```

```

    public enum Authentication {MULTIPLE, SINGLE_INSTANCE,
        SINGLE, NONE;}
}

package emailheaderinformation.model;

import java.util.Date;

public class Device {
    private final String name;
    private final String software;
    private final Date receivedTime;
    private final String origin;
    private float latitude;
    private float longitude;
    private Device next;
    private boolean setNext = false;
    private String owningOrganisation;

    public Device (Device next, String name, String software,
        Date receivedTime, String origin) {
        this.name = name;
        this.software = software;
        this.receivedTime = receivedTime;
        this.origin = origin;
    }

    public float getLongitude () {
        return longitude;
    }

    public void setLongitude (float longitude) {
        this.longitude = longitude;
    }

    public float getLatitude () {
        return latitude;
    }
}

public enum Authentication {MULTIPLE, SINGLE_INSTANCE,
    SINGLE, NONE;}
}

package emailheaderinformation.model;

import emailheaderinformation.model.Access.Authentication;
import emailheaderinformation.model.Access.Complexity;
import emailheaderinformation.model.Access.Vector;

public class AccessBuilder {
    private Authentication authentication;
    private Complexity complexity;
    private Vector vector;

    public AccessBuilder setVector (Vector vector) {
        this.vector = vector;
        return this;
    }

    public AccessBuilder setComplexity (Complexity complexity)
    {
        this.complexity = complexity;
        return this;
    }

    public AccessBuilder setAuthentication (Authentication
        authentication) {
        this.authentication = authentication;
        return this;
    }

    public Access createAccess () {
        return new Access(vector, complexity, authentication);
    }
}

```

```

public void setLatitude (float latitude) {
    this.latitude = latitude;
}

public Device getNext () {
    return next;
}

public void setNext (Device next) {
    if (!setNext) {
        this.next = next;
        setNext = true;
    }
}

public String getName () {
    return name;
}

public String getSoftware () {
    return software;
}

public Date getReceivedTime () {
    return receivedTime;
}

public String getOrigin () {
    return origin;
}

public String getOwningOrganisation () {
    return owningOrganisation;
}

public void setOwningOrganisation (String
    owningOrganisation) {
    this.owningOrganisation = owningOrganisation;
}
}

package emailheaderinformation.model;

import java.util.Date;

public class DeviceBuilder {
    private Device next = null;
    private String name;
    private String software;
    private Date receivedTime;
    private String origin;

    public DeviceBuilder setNext (Device next) {
        this.next = next;
        return this;
    }

    public DeviceBuilder setName (String name) {
        this.name = name;
        return this;
    }

    public DeviceBuilder setSoftware (String software) {
        this.software = software;
        return this;
    }

    public DeviceBuilder setReceivedTime (Date receivedTime) {
        this.receivedTime = receivedTime;
        return this;
    }
}

```



```

    }

    public Device createDevice () {
        return new Device(next, name, software, receivedTime,
            origin);
    }

    public DeviceBuilder setOrigin (String origin) {
        this.origin = origin;
        return this;
    }
}

package emailheaderinformation.model;

public class DeviceInformation {
    private int id;
    private String ipAddress;
    private float lat;
    private float lon;
    private final String software;
    private final String owner;

    public DeviceInformation (int id, String ip, float lat,
        float lon, String software, String owner) {
        this.id = id;
        this.ipAddress = ip;
        this.lat = lat;
        this.lon = lon;
        this.software = software;
        this.owner = owner;
    }
}

package emailheaderinformation.model;

public Device createDevice () {
    return new Device(next, name, software, receivedTime,
        origin);
}

public DeviceBuilder setOrigin (String origin) {
    this.origin = origin;
    return this;
}
}

package emailheaderinformation.model;

public class DeviceInformation {
    private int id;
    private String ipAddress;
    private float lat;
    private float lon;
    private final String software;
    private final String owner;

    public DeviceInformation (int id, String ip, float lat,
        float lon, String software, String owner) {
        this.id = id;
        this.ipAddress = ip;
        this.lat = lat;
        this.lon = lon;
        this.software = software;
        this.owner = owner;
    }
}

package emailheaderinformation.model;

import java.util.ArrayList;
import java.util.List;

public class FoundInformation {
    private String name;
    private String organisation;
    private String software;
    private List<Fact> facts = new ArrayList<>();
    private List<Username> usernameList = new ArrayList<

```

```

        Username>());

    public List<DeviceInformation> getDevices () {
        return devices;
    }

    private List<DeviceInformation> devices = new ArrayList
    <>();

    private int deviceCount = 0;

    public List<Username> getUsernameList () {
        return usernameList;
    }

    public void addUsername (String context, String username)
    {
        usernameList.add(new Username(context, username));
    }

    public String getName () {
        return name == null ? "Not_found" : name;
    }

    public void setName (String name) {
        this.name = name;
    }

    public void addFact (String factClass, String factType,
        String factDetails) {
        facts.add(new Fact(factClass, factType, factDetails));
    }

    public List<Fact> getFacts () {
        return facts;
    }

    public String getSoftware () {
        return software == null ? "Not_found,or_using_web_u
        client" : software;
    }

    public void setSoftware (String software) {
        this.software = software;
    }

    public void addDevice (String ip, float lat, float lon,
        String software, String owner) {
        DeviceInformation di = new DeviceInformation(deviceCount
        ++, ip, lat, lon, software, owner);
        devices.add(di);
    }

    public String getOrganisation () {
        return organisation;
    }

    public void setOrganisation (String organisation) {
        this.organisation = organisation;
    }

    package emailheaderinformation.model;

    import java.util.HashMap;
    import java.util.Map;

    public class Header {
        private Device startDevice;
        private Map<String, String> fields = new HashMap<String,
        String>();

        public Header () {

```

```

        super();
    }

    public Device getStartDevice () {
        return startDevice;
    }

    public void setStartDevice (Device startDevice) {
        this.startDevice = startDevice;
    }

    public Map<String, String> getFields () {
        return fields;
    }

    public void setFields (Map<String, String> fields) {
        this.fields = fields;
    }
}

package emailheaderinformation.model;

public class Impact {
    private Value availability;
    private Value confidentiality;
    private Value integrity;

    Impact (Value availability, Value confidentiality, Value
        integrity) {
        this.availability = availability;
        this.confidentiality = confidentiality;
        this.integrity = integrity;
    }

    @Override public String toString () {
        return String.format("%s,%s,%s",
            availability.toString(),
            confidentiality.toString(),
            integrity.toString());
    }

    public Value getConfidentiality () {
        return confidentiality;
    }

    public Value getIntegrity () {
        return integrity;
    }

    public Value getAvailability () {
        return availability;
    }

    public enum Value {NONE, PARTIAL, COMPLETE;}
}

package emailheaderinformation.model;

import static emailheaderinformation.model.Impact.Value;

public class ImpactBuilder {
    private Value availability;
    private Value confidentiality;
    private Value integrity;

    public ImpactBuilder setAvailability (Value availability)
    {
        this.availability = availability;
        return this;
    }
}

```

```

public ImpactBuilder setConfidentiality (Value
    confidentiality) {
    this.confidentiality = confidentiality;
    return this;
}

public ImpactBuilder setIntegrity (Value integrity) {
    this.integrity = integrity;
    return this;
}

public Impact createImpact () {
    return new Impact(availability, confidentiality,
        integrity);
}

package emailheaderinformation.model;

/**
 * Created by jaclark on 18/04/16.
 */
public class VulnerabilityDisclosure implements Comparable
{
    private final String cveId;
    private final String summary;
    private final float score;
    private final String product;
    private final Impact impact;
    private final Access access;
    private final Date date;

    VulnerabilityDisclosure (String cveId, String summary,
        float score, String product, Impact impact,
            Access access, Date date) {
        this.cveId = cveId;
        this.summary = summary;
        this.score = score;
        this.product = product;
        this.impact = impact;
        this.access = access;
        this.date = date;
    }

    @Override public int compareTo (Object o) {
        if (o instanceof VulnerabilityDisclosure) {
            return this.cveId.compareTo(((VulnerabilityDisclosure)

```

```

        o).cveId);
    } else {
        return -1;
    }
}

public String getProduct () {
    return product;
}

public Date getDate () {
    return date;
}
}

package emailheaderinformation.model;

import org.apache.commons.lang3.SystemUtils;

import java.util.Date;

public class VulnerabilityDisclosureBuilder {
    private String cveId;
    private String summary;
    private float score;
    private String product;
    private Impact impact;
    private Access access;
    private Date date;

    public VulnerabilityDisclosureBuilder setDate (Date date)
    {
        System.out.println(date.toString());
        this.date = date;
        return this;
    }
}

    public VulnerabilityDisclosureBuilder setCveId (String
        cveId) {
        this.cveId = cveId;
        return this;
    }

    public VulnerabilityDisclosureBuilder setSummary (String
        summary) {
        this.summary = summary;
        return this;
    }

    public VulnerabilityDisclosureBuilder setScore (float
        score) {
        this.score = score;
        return this;
    }

    public VulnerabilityDisclosureBuilder setImpact (Impact
        impact) {
        this.impact = impact;
        return this;
    }

    public VulnerabilityDisclosureBuilder setAccess (Access
        access) {
        this.access = access;
        return this;
    }

    public VulnerabilityDisclosureBuilder setProduct (String
        product) {
        this.product = product;
        return this;
    }
}

```

```

public VulnerabilityDisclosure
    createVulnerabilityDisclosure () {
        return new VulnerabilityDisclosure(cveId, summary, score
            , product, impact, access, date);
    }
}

package emailheaderinformation.parser;

import java.text.SimpleDateFormat;
import java.util.Date;

/**
 * Created by joshua on 14/11/15.
 */
public class DateUtil {

    // non-standard date formats to take into account spacing
    // introduced in parser
    // module
    private static final SimpleDateFormat rfc822DateFormats[]
        = new SimpleDateFormat[] {
            new SimpleDateFormat( "EEE,ddMMM_yyy_HH:mm:ss_z"),
            new SimpleDateFormat( "EEE,ddMMM_yyy_HH:mm_z"),
            new SimpleDateFormat( "EEE,ddMMM_yyy_HH:mm:ss_z"),
            new SimpleDateFormat( "EEE,ddMMM_yyy_HH:mm:ss_z"),
            new SimpleDateFormat( "ddMMM_yyy_HH:mm_z"),
            new SimpleDateFormat( "ddMMM_yyy_HH:mm:ss_z"),
            new SimpleDateFormat( "ddMMM_yyy_HH:mm_z"),
            new SimpleDateFormat( "ddMMM_yyy_HH:mm:ss_z");
        };

    /**
     * Parse an RFC 822 date string.
     *
     * @param dateString
     */
    public static Set<String> keywords = new HashSet<String>
        >();
}

```

```

* The date string to parse
*
* @return The date, or null if it could not be parsed.
*/
public static Date parseRfc822DateString (String
    dateString) {
    Date date = null;
    for (SimpleDateFormat sdf : rfc822DateFormats) {
        try {
            date = sdf.parse(dateString);
        } catch (java.text.ParseException e) {
            // Don't care, we'll just run through all
        }
        if (date != null) {
            return date;
        }
    }
    return null;
}

package emailheaderinformation.parser;

import com.devdaily.system.CommandExecutor;
import emailheaderinformation.model.Device;
import emailheaderinformation.model.DeviceBuilder;
import emailheaderinformation.model.Header;

import java.io.IOException;
import java.util.*;

public class EmailParser {
    private static Set<String> keywords = new HashSet<String>
        >();
}

```

```

public EmailParser () {
    setupParser();
}

private static void setupParser () {
    String[] kws = {
        "Received", "from", "by", "via", "with", "id", "for"
        , ";" };
    Collections.addAll(keywords, kws);
}

public Header parse (String inputEmail) {
    try {
        ArrayList<String> command = new ArrayList<>();
        command.add("python2");
        command.add("HeaderParserTrace.py");
        command.add(inputEmail);
        SystemCommandExecutor commandExecutor = new
            SystemCommandExecutor(command);
        int result = commandExecutor.executeCommand();
        if (result == 0) {
            StringBuilder output = commandExecutor.
                getStandardOutputFromCommand();
            Header header = new Header();
            header = createDeviceChain(header, output.toString()
                );
            command.set(1, "HeaderParserFields.py");
            SystemCommandExecutor commandExecutor1 = new
                SystemCommandExecutor(command);
            if (commandExecutor1.executeCommand() == 0) {
                StringBuilder fields = commandExecutor1.
                    getStandardOutputFromCommand();
                String[] keyval = fields.toString().split("\n");
                for (int i = 0; i < keyval.length - 1; i += 2) {
                    if (header != null) {
                        header.getFields().put(keyval[i], keyval[i +
1]]);
                    }
                }
                return header;
            } else {
                System.err.println(commandExecutor.
                    getStandardErrorFromCommand().toString());
                return null;
            }
        } else {
            System.err.println(commandExecutor.
                getStandardErrorFromCommand().toString());
            return null;
        }
    } catch (IOException | InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return null;
}

/**
 * @param header
 * - the header model object to populate
 * @param output
 * - the header entry string
 *
 * @return
 */
private static Header createDeviceChain (Header header,
    String output) {
    try {
        Device lastDevice = null;
        Device firstDevice = null;
        for (String line : output.split("\n")) {
            Device device = constructDevice(new StringChunker(

```

```

        line));
        if (firstDevice == null) {
            firstDevice = device;
        }
        if (lastDevice != null) {
            lastDevice.setNext(device);
        }
        lastDevice = device;
    }
    header.setStartDevice(firstDevice);
    return header;
} catch (NullPointerException e) {
    // we've reached the end of the text to parse
    e.printStackTrace();
    return header;
}
}

/**
 * RFC 822 defines the structure of the Received section
 * as follows:
 *
 * <p>
 * Example Received field:
 *
 * <p>
 * Received: from relay12.mail.ox.ac.uk (129.67.1.163) by
 *          HUB05.ad.oak.ox.ac.uk (163.1.154.231)
 * with Microsoft SMTP Server id 14.3.169.1; Sat, 14 Nov
 * 2015 10:55:35 +0000
 *
 * <p>
 * received = "Received" ":" ; one per relay ["from"
 *          domain] ; sending host ["by" domain] ;
 * receiving host ["via" atom] ; physical path *("with"
 *          atom) ; link/mail protocol ["id" msg-id] ;
 * receiver msg id ["for" addr-spec] ; initial form ":" ;
 * date-time ; time received
 *
 * <p>

```

```

* addr-spec = local-part "@" domain ; global address
* <p>
* local-part = word *("." word) ; uninterpreted ; case-
*          preserved
* <p>
* domain = sub-domain *("." sub-domain) sub-domain =
*          domain-ref / domain-literal domain-ref =
*          atom domain-literal = "[" *(dtext / quoted-pair) "]"
*          dtext = <any CHAR excluding "[", ; => may
*          be folded ">," ">" & CR, & including linear-white-
*          space>
* <p>
* msg-id = "<" addr-spec ">" ; Unique message id
* <p>
* date-time = [ day "," ] date time ; dd mm yy hh:mm:ss
*          zzz
* <p>
* day = "Mon" / "Tue" / "Wed" / "Thu" / "Fri" / "Sat" /
*          "Sun"
* <p>
* date = 1*2DIGIT month 2DIGIT ; day month year ; e.g.
*          20 Jun 82
* <p>
* month = "Jan" / "Feb" / "Mar" / "Apr" / "May" / "Jun" /
*          "Jul" / "Aug" / "Sep" / "Oct" / "Nov" /
*          "Dec"
* <p>
* time = hour zone ; ANSI and Military
* <p>
* hour = 2DIGIT ":" 2DIGIT [":" 2DIGIT] ; 00:00:00 -
*          23:59:59
* <p>
* zone = "UT" / "GMT" ; Universal Time ; North American :
*          UT / "EST" / "EDT" ; Eastern: - 5/ - 4
*          / "CST" / "CDT" ; Central: - 6/ - 5 / "MST" / "MDT" ;
*          Mountain: - 7/ - 6 / "PST" / "PDT" ;

```



```

* Pacific: - 8/ - 7 / 1ALPHA ; Military: Z = UT; ; A:-1;
  (J not used) ; M:-12; N:+1; Y:+12 / (
* ("+" / "-") 4DIGIT ) ; Local differential ; hours+min.
  (HHMM)
* <p>
* Atoms may not contain SPACE
*
* @param headerTokenizer
*   - the existing object to separate each field in the
  header
*
* @return a @Device object containing all the necessary
  information
*/
private static Device constructDevice (StringChunker
headerTokenizer) {
  DeviceBuilder builder = new DeviceBuilder();

  // Received:
  String peeked = headerTokenizer.next();
  boolean finished = false;
  while (!finished) {
    switch (peeked) {
      case "from":
        headerTokenizer.next();
        String[] origin = extractUntilKeyword(
          headerTokenizer);
        builder.setOrigin(origin[0]);
        peeked = origin[1];
        break;
      case "by":
        headerTokenizer.next();
        String[] name = extractUntilKeyword(
          headerTokenizer);
        builder.setName(name[0]);
        peeked = name[1];
        break;
      case "id":
      case "for":
      case "via":
        headerTokenizer.next();
        peeked = extractUntilKeyword(headerTokenizer)[1];
        break;
      case "with":
        headerTokenizer.next();
        String[] software = extractUntilKeyword(
          headerTokenizer);
        builder.setSoftware(software[0]);
        peeked = software[1];
        break;
      default:
        if (peeked.endsWith(",") {
          finished = true;
          headerTokenizer.next();
        } else {
          peeked = headerTokenizer.peek();
        }
        break;
    }
  }

  Date date = null;
  StringBuilder dateStrB = new StringBuilder("");
  while (date == null && headerTokenizer.hasNext()) {
    String token = headerTokenizer.next();
    dateStrB.append(token);
    if (dateStrB.charAt(0) == ',') {
      dateStrB.deleteCharAt(0);
      if (!dateStrB.toString().isEmpty()) {
        while (Character.isWhitespace(dateStrB.charAt(0)))
          {
            dateStrB.deleteCharAt(0);
          }
      }
    }
  }
}

```

```

    }
    }
    dateStrB.append("\u");
    date = DateUtil.parseRfc822DateString(dateStrB.
    toString());
}

builder.setReceivedTime(date);

return builder.createDevice();
}

private static String[] extractUntilKeyword (
    StringChunker headerTokenizer) {
    String token = "";
    String lastSeen;
    lastSeen = headerTokenizer.peek();
    while (headerTokenizer.hasNext() && !(keywords.contains(
        lastSeen) || lastSeen.endsWith(";"))) {
        token += "\u" + headerTokenizer.next();
        if (headerTokenizer.hasNext()) {
            lastSeen = headerTokenizer.peek();
        }
    }
    return new String[] { token, lastSeen };
}

package emailheaderinformation.parser;

import java.util.Arrays;

class StringChunker {
    private String[] words;
    private int position = 0;

StringChunker(String s) {
    words = s.replace(";", "\u");
}

boolean hasNext() {
    return words.length > position;
}

String peek() {
    return words[position];
}

String next() {
    return words[position++];
}

package emailheaderinformation;
import fi.iki.elonen.NanoHTTPD;

import javax.swing.*;
import java.awt.*;
import java.io.*;
import java.net.URISyntaxException;

class WebServer extends NanoHTTPD {
    private String mPage;

/**
 * Constructs an HTTP server on given port.
 *
 * @param port
 */
WebServer (Component component, String page, int port)

```

```

throws IOException {
    super(port);
    start(NanoHTTPD.SOCKET_READ_TIMEOUT, false);
    JOptionPane.showMessageDialog(component,
        String.format("Running! Point
            ↳your_browsers_to" +
                "http://
                    localhost:%d
                        /", port));

    mPage = page;
}

/**
 * Override this to customize the server.
 * <p/>
 * <p/>
 * (By default, this returns a 404 "Not Found" plain text
  error response.)
 *
 * @param session
 *     The HTTP session
 *
 * @return HTTP response, see class Response for details
 */
@Override public Response serve (IHTTPSession session) {
    switch (session.getUri()) {

```

```

        case "/world-50m.json":
            try {
                File file = new File(this.getClass().getResource(
                    "/world-50m.json").toURI());
                try (InputStream in = new FileInputStream(file)) {
                    try (InputStreamReader isr = new
                        InputStreamReader(in)) {
                        try (BufferedReader br = new BufferedReader(isr
                            )) {
                            StringBuilder sb = new StringBuilder();
                            br.lines().forEach(sb::append);
                            return newFixedLengthResponse(sb.toString());
                        }
                    }
                } catch (IOException | URISyntaxException e) {
                    e.printStackTrace();
                }
                break;
            }
            default:
                return newFixedLengthResponse(mPage);
        }
        return null;
    }
}

```