Thesis Dissertation

AN EXPERIMENTAL STUDY OF TLS USAGE IN MAIL SYSTEMS

Georgia Christou

UNIVERSITY OF CYPRUS

COMPUTER SCIENCE DEPARTMENT

May 2021

UNIVERSITY OF CYPRUS

COMPUTER SCIENCE DEPARTMENT

An Experimental Study of TLS Usage in Mail Systems

Georgia Christou

Supervisor

Dr. Elias Athanasopoulos

Thesis submitted in partial fulfilment of the requirements for the award of degree of Bachelor in Computer Science at University of Cyprus

May 2021

Acknowledgements

As my thesis dissertation comes to an end, I would like to express my gratitude to the various people that made my road to this day feasible.

For starters, I would like to thank my supervisor, Dr. Elias Athanasopoulos, whose guidance, dedication and understanding were of absolute importance to the development of this project. I am grateful to be given the chance to work for this project, and I am certain it would hardly be the same experience if not for our excellent collaboration and communication.

Of course, I also have to acknowledge the importance of Dr. Georgios Portokalidis's contribution to this project, whose insight and ideas allowed for various optimizations to change the project for the better.

Finally, I owe a big thank you to my friends and family, having been on my side supporting me during the course of yet another important year, just like they did consistently and sincerely throughout my life.

Summary

To protect private information communicated over e-mail, it is common practice for e-mails to travel over the Internet using encryption. While an e-mail travels to reach its recipient's mailbox, encryption stands to hide the e-mail's contents by transforming them into an unreadable form so they become useless in the wrong hands. Unfortunately, though most e-mail providers today deploy encryption in their transits, there are cases that encryption is omitted, leaving an e-mail's sensitive information utterly unprotected in the wrong hands, which does not depend on the recipient but entirely on the delivery servers. This research aims to examine how frequently this happens, which websites expose potentially sensitive information, as well as the various factors that contribute to it.

The use of Transport Layer Security (TLS) in today's e-mail transits was examined by focusing on the TLS usage of multiple Gmail mailboxes. As a starting point, a Gmail-based TLS scanning tool was designed to be installed on Chrome browsers. The tool was then distributed to different Gmail users to scan the security of their mailbox and collect brief summaries of data to be used for evaluation purposes. As part of this thesis, a sample of Gmail mailboxes were examined, revealing a generally descent but not nearly ideal image of encrypted to unencrypted e-mails, as well as an unforeseen lack of consistency in the TLS behaviour of different e-mail providers.

Contents

List of Figures

List of Algorithms

List of Tables

Chapter 1

Introduction

Contents

1.1 Problem statement

Though e-mail security is a rather expected part of today's e-mail infrastructure, the original design of e-mail was not introduced with e-mail security into the picture. When e-mail was first established, over 50 years ago, e-mails were all transmitted in plain-text. Ultimately, this induced a number of security risks and concerns, which, in order to be tackled, demanded the introduction and development of various security tools and techniques that in turn formed the picture of e-mail infrastructure as we know it today.

Even when e-mail security came into the picture, nonetheless, it was never much of a burning issue or a concern for the general public, not until the most recent years.

In 2016, Google enhanced Gmail with a vital security update, introducing a red padlock [4] to any e-mail sent or delivered without encryption, an update that allowed users to ponder over the security of their mailboxes and consequently their privacy.

Gmail's introduction of the red padlock did not expose something concerning or afflicting Gmail users exclusively, but rather it revealed a worrying truth regarding the e-mail community as a whole. As e-mail users, unfortunately, we have no control over how our e-mails arrive to us or how they travel over the Internet before they reach our mailbox. What's more than that is that, in many cases, neither as the sender side of an e-mail exchange do we control or know how an e-mail leaves our mailbox. With that in mind, it is fairly rational that we do not only expect but trust on our e-mail providers to ensure our right to privacy. As Gmail's red padlock revealed, however, this is not something we should be taking for granted, as our privacy in many times is completely swept aside, with a number of e-mails even today still travelling over the Internet in plain-text.

1.2 Project objectives

As part of this project we present a sampling evaluation of today's use of Transport Layer Security (TLS) in e-mail transits, by collecting and processing header information from a number of e-mails delivered to various Gmail mailboxes.

Overall, our study reveals a generally descent yet far from perfect image of encrypted to unencrypted e-mails, with a significant number of e-mails found to have been delivered in plain-text and a noticeable number of e-mail providers found to be inconsistent in their TLS usage.

Our study's focus lies in examining both the profiles of individual e-mails as well as the behaviour of their e-mail providers.

We examine the TLS usage in individual e-mails by comparing and contrasting encrypted and unencrypted e-mails in regards to,

- i their year of delivery,
- ii their size,
- iii their MIME type and
- iv the DNS status of their e-mail provider.

Our analysis of approximately 100K e-mails reveals a declining trend in the volume of unencrypted e-mails delivered in the most recent years, a uniform distribution of encrypted and unencrypted e-mails between different sizes and DNS statuses of their providers, whereas it appears that some MIME types are more prone to come in plain-text than others.

A volume of around 1K e-mail providers are behind the delivery of our examined e-mails. As part of our evaluation, we target the TLS usage of these providers in respect to,

- i their portion of unencrypted deliveries,
- ii the consistency in their encrypted and/or unencrypted deliveries,
- iii their popularity among users and

iv their activity, as this arises from their total number of e-mail deliveries.

Our e-mail providers' examination reveals that providers with high popularity and activity are more likely to be responsible for none or very few unencrypted deliveries, while for the case of the most inconsistent providers, they are more likely to be of very low popularity and activity.

1.3 Project contributions

Though a number of previous studies focused on the analysis of e-mail patterns, their motivation did not lie in investigating how these pattern can potentially link to e-mail security. Similarly, though a number of studies in the most recent years examined the TLS usage in e-mail transits, their methodology did not rely on a number of volunteering participants but rather on servers scans, connections monitoring and/or security evaluation tests. Similarly to our study, the latter studies revealed an alarming volume of unencrypted deliveries in today's e-mail infrastructure, however they did not target how the profiles of such e-mails or their deliverers can potentially hint at this phenomenon.

Developed with Gmail's red padlock feature as a reference point, TLS SCANNER FOR GMAIL is a Google Chrome extension designed as part of this thesis in order to give us insight into the mailboxes of a number of Gmail users willing to participate in our study.

TLS SCANNER FOR GMAIL offers three major benefits. For starters, it gives Gmail users that harness it an overview of the security of their mailbox, providing them with links to any e-mails delivered to them in plain-text. Moreover, the tool stands as a research tool, offering its users the option to participate in our study by anonymously uploading brief summaries of their results to cloud so we can further analyse and explore them in regards to other Gmail mailboxes. The implementation of TLS SCANNER FOR GMAIL can moreover stand as a reference point to build similar tools and examine the TLS usage in other e-mail services, other than Gmail.

1.4 Thesis organization

This thesis starts off with an Ethics section meant to highlight the ethical details concerning the development and distribution of TLS SCANNER FOR GMAIL. It then proceeds with providing some Background knowledge necessary to understand the remaining of this project. The thesis continues with explaining the overall Architecture and Implementation details of the project, including the development of TLS SCANNER FOR GMAIL extension as well as the experimental setup and data collection procedure. The thesis concludes with presenting the Evaluation methodology and its emerging results. A section devoted to other Related Work is also included before the Conclusion section.

Chapter 2

Ethics

As part of this project we aim to inspect the TLS usage in today's e-mail infrastructure by preserving a cautious, privacy-focused approach towards our study's participants.

TLS SCANNER FOR GMAIL extension is developed to run as a local process, confining its execution to the user's machine. That way, any processing steps the extension performs and any data it goes over or collects do not forsake their confidentiality within the user's computer until the user consciously allows for it.

If a user of the extension decides to become a study participant by making their results public to be used as part of the study, then all they have to do is upload them to a shared Dropbox folder among the rest of the users' results. Dropbox offers multiple advantages, among which allowing for anonymous uploads, while at the same time ensuring that no one but the folder's creator can gain access into the folder's input, something that ensures that results uploaded by one participant cannot be seen or modified by any other participant.

Of course, the extension's local nature of execution and the advantages offered by the participants' Dropbox uploads are only as privacy-preserving as the extension's gathered data allow them to be. The most important factor to ensure a privacy-preserving policy in our data collection procedure is ensuring that any results uploaded to Dropbox do not contain information that could be used to identify the user or trace back to them in any way. To do that, the extension's insight into a user's data only goes as far as to fetch, process and record information found within e-mail headers. The results a user ultimately uploads to Dropbox merely consist of abstract descriptions of the e-mails found within the user's mailbox, including information such as e-mail sizes, dates and e-mail providers, among others, but no information on the e-mail contents or the identities of senders and/or recipients. TLS SCANNER FOR GMAIL, however, does need one piece of sensitive information from the users' mailboxes, a way to distinguish the users in case one user uploads multiple files of results. To do that while preserving user anonymity, Dropbox uploads come with a unique identifier for each distinct Gmail mailbox, a hashed value of the user’s Gmail account, which by nature ensures the uniqueness between distinct accounts and at the same time hides the user’s real identity behind a hash.

Chapter 3

Background

Contents

3.1 Transport Layer Security in e-mail communication

Transport Layer Security (TLS) [7] is an encryption protocol designed to provide security over communications over the Internet. When TLS is used for communication over e-mail, the protocol protects an e-mail's contents from being read by third parties while the e-mail is in transit. An e-mail, however, may pass through various servers before reaching the receiver's mailbox. At the very least, only two servers participate in the exchange, the sender's Mail Transfer Agent (MTA) and the receiver's MTA, both of which communicate with their respective Mail User Agent (MUA) in order to transfer the e-mail from or to the user's mailbox. In this simple scenario, the connection's security solemnly depends on the protocol used by the sender's MTA. When intermediate servers participate in the exchange, then the protocols used by each of them may vary. Each participating server receives the e-mail via an established connection, secure or unsecure, and passes the e-mail on to the next server by establishing its own connection, secure or unsecure. Eventually, the e-mail arrives at the receiver's mailbox via the connection established by the last participating server.

3.2 Routing information lying in e-mail headers

The complete list of servers participating in an e-mail's transfer can be found in the e-mail's headers, by unfolding the information contained in the headers' `Received` fields from bottom to top. `Received` fields are appended whenever one server receives the e-mail from another via a given protocol. The identities of the two servers as well as the given protocol are usually recorded in the respective `Received` field by the recipient MTA, however, it is not uncommon for a server to append the field with insufficient information on the transfer. Luckily, this is not the case for the latest appended `Received` fields when these are recorded by the recipient's MTA and MUA. In the case of Gmail, these fields bare a consistent and predictable structure that is not only reliable but presents a well-rounded understanding of how an e-mail ultimately arrives at the user's mailbox.

3.3 Google Chrome extensions

We can gain access into a Gmail mailbox via the use of a Google Chrome extension. Google Chrome extensions [1] are JavaScript based applications designed to be installed in Chrome browsers so as to alter or enhance the browsers' functionalities. Though most extensions offer a user interface, most often in the form of a mini popup appearing at the top-right corner of the screen with the use of a button, Chrome extensions are also capable to run as background processes, without requiring any user interaction. No matter the extension's design, nevertheless, in order for a Chrome extension to be functional, it requires a manifest.json file, a special file including defining information on the extension, such as a list of all the scripts it is allowed to run or any permissions it requires in order to access various Chrome APIs. When it comes to making a Chrome extension public for people to use, Chrome extensions are usually found and installed via the Chrome Web Store, but it is also possible for people to install an unpublished extension to their browser, considering the extension's source code is available for people to download on their computer.

Chapter 4

Architecture

Contents

4.1 The need for a tool to inspect Gmail mailboxes

Any Gmail user can manually inspect the headers of their e-mails [5] and determine the protocol with which each e-mail was delivered to Gmail's MTA. This is not a particularly difficult thing to do, however, it is surely a time consuming and automated procedure. With that in mind, a Chrome extension can perform this task on behalf of the user, gain access into their Gmail mailbox and scan the headers of all their e-mails, one after the other. Other than merely inspecting each e-mail's latest used transfer protocol, the extension can moreover collect further information lying in each e-mail's headers. TLS SCANNER FOR GMAIL is a Chrome extension designed as part of this research to do just that.

4.2 How a Chrome extension can gain insight into any Gmail mailbox

Figure 4.1 presents how developing and employing a Google Chrome extension can give us insight into anyone's Gmail mailbox.

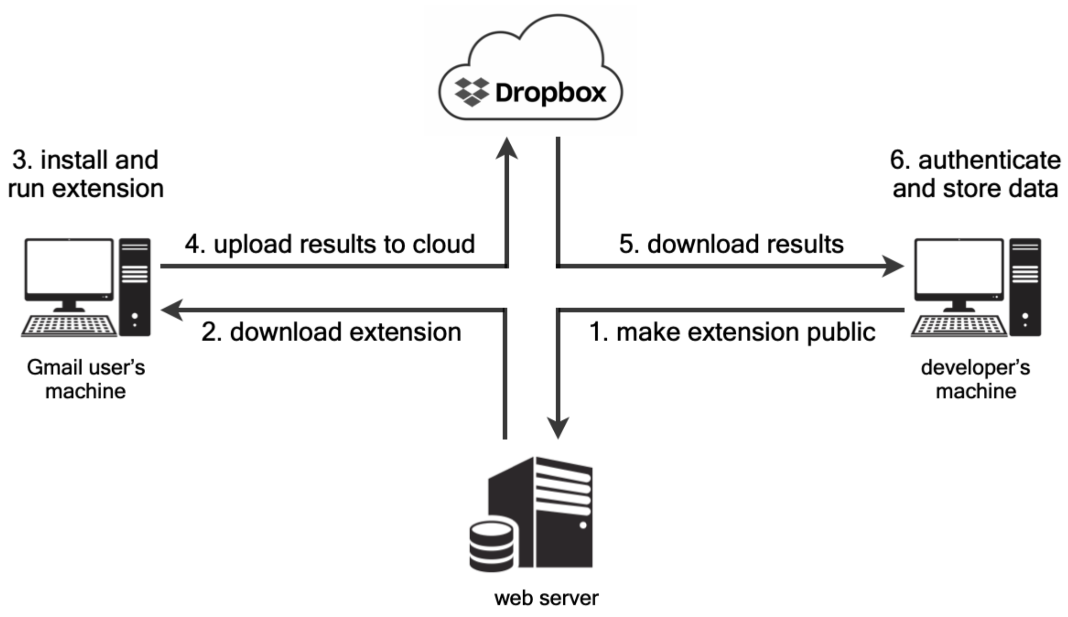


Figure 4.1: Overview of how a Chrome extension can be used as an experimental tool to gain insight into any Gmail mailbox

TLS SCANNER FOR GMAIL is an open source Chrome extension, available for download on a public website. Anyone with a local copy of the extension can install it on their Chrome browser and anyone with a Google account can then have it scan their Gmail mailbox.

Just like a Gmail user would manually scan their mailbox locally, the extension imitates this behaviour and bounds its execution to the user's computer. Only after the extension's execution is over, TLS SCANNER offers the user the chance to make their results public. The way TLS SCANNER presents this choice is by offering the user a link to download a local copy of their results and another link to upload these results to cloud so we can ultimately gain access to them.

4.3 The practicality of Chrome extensions in developing Gmail-based applications

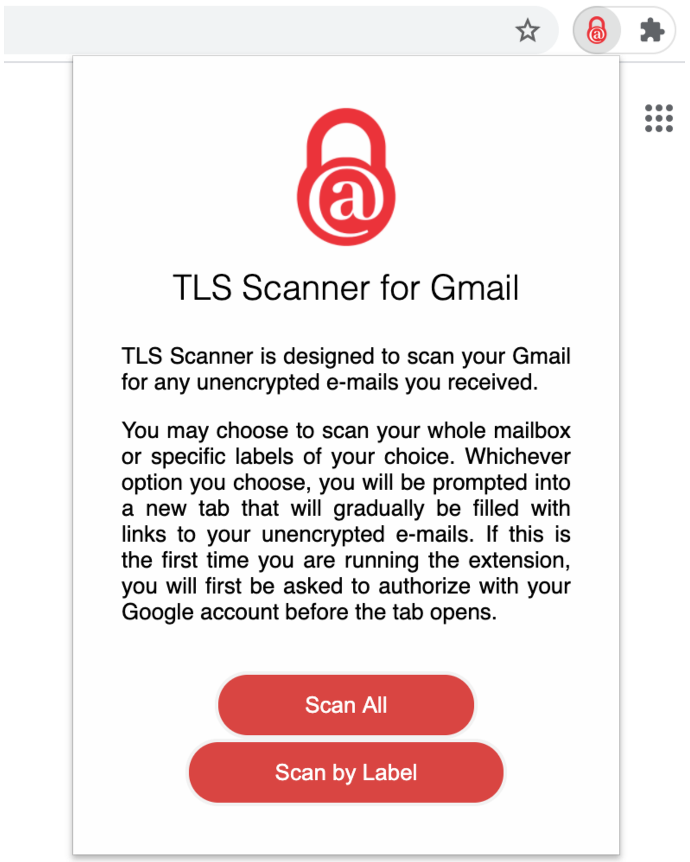
Considering both Gmail mail service and Chrome web browser are developed by Google, it is no surprise that their RESTful APIs offer functionalities that collaborate and interact in harmony with one another, making the development of Gmail Chrome extensions considerably handy. As a Gmail Chrome extension itself, TLS SCANNER makes the most out of Chrome API and Gmail API [3], while at the same time it harnesses its JavaScript's client side nature to confine its execution to the user's computer.

4.4 The extension's interface with the user

TLS SCANNER is free for anyone to access and download on their computer, whether they are interested to participate in the study or merely curious to scan the security of their mailbox. Following the instruction manuals found on the extension's website, anyone with a local copy of the extension can install it on their Chrome browser, regardless of whether they have a Gmail account or not. The extension appears among any other Chrome extensions at the top-right corner of the user's screen and presents an introductory popup whenever it is clicked, as this is illustrated in Figure 4.2. The popup appears even if the user doesn't have a Gmail account, but its functionality only goes this far. At the bottom of the popup, the extension asks the user to choose among two scanning options, to scan their whole mailbox or only specific labels of their choice, as these are determined dynamically via a custom modal window illustrated in Figure 4.3. To proceed with either of the two scanning options, the extension requires the user to own a Google account and consequently a Gmail account, necessary to authorize the extension with.

4.5 The extension's required permissions and data of interest

The extension takes a structured, iterative approach to scan a user's mailbox as soon as it's given permission to. The process requires read-only access to the user's e-mails, since there is no need to alter anything, merely to go through each e-mail's headers. While going through the user's mailbox, the extension keeps track of two distinct results statements, one for the user to see and another to be used as part of the experiment. The first results statement is updated whenever the extension classifies an e-mail as unsecure, while the second results statement is updated after any e-mail is processed. Specifically, whenever an e-mail is classified as unsecure, the user gets to see a brief summary of the given e-mail

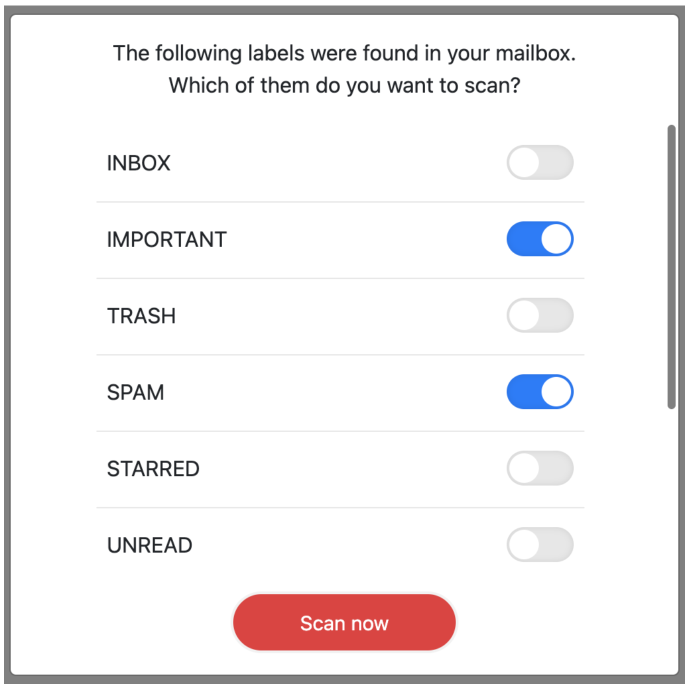


TLS Scanner for Gmail

TLS Scanner is designed to scan your Gmail for any unencrypted e-mails you received.

You may choose to scan your whole mailbox or specific labels of your choice. Whichever option you choose, you will be prompted into a new tab that will gradually be filled with links to your unencrypted e-mails. If this is the first time you are running the extension, you will first be asked to authorize with your Google account before the tab opens.

Figure 4.2: The extension’s popup screen



The following labels were found in your mailbox. Which of them do you want to scan?

INBOX	off
IMPORTANT	on
TRASH	off
SPAM	on
STARRED	off
UNREAD	off

Figure 4.3: The extension’s modal screen for labels customization

on their screen, including the e-mail's subject, sender and latest transfer protocol, as well as a link to open that e-mail in a new tab. At the same time, the extension appends its own results statement, which the user doesn't know of until the very end of execution. The extension's results statement consists of records of all the processed e-mails, with each record including,

i the e-mail service that initiated the e-mail's delivery,

ii information on the provider that ultimately delivered the e-mail to Gmail's MTA,

iii the transfer protocol the e-mail reached the user's mailbox with,

iv the date of the delivery,

v the e-mail's size estimate and

vi the type of data lying inside the e-mail, declared via the e-mail's media type (also known as MIME type).

4.6 How a user of the extension becomes a study participant

As soon as the extension's processing is over, it is now up to the user to complete their part in the experiment, by choosing to convey the extension's results statement to us. This is not performed automatically. The user downloads a copy of their results statement so they can examine it, if they want to, and make sure they comply with the data we ask from them. Considering the user decides to deliver their results statement, all they need to do is upload them to a shared cloud folder accessible via a Dropbox file request. Dropbox file requests have three major benefits. For starters, they allow for anyone with the link to the request to make an upload, regardless of whether they have or not a Dropbox account. Moreover, they allow for anonymous uploads. And finally, no matter how many people have access to a file request link, no one but the file request's creator has access to the files uploaded to that link, which protects individual user results from being read or modified by non-entitled third parties. The fact that a user's results cannot be modified after their upload does not make them, however, overall reliable or resistant to change. Not only can the file request link accidentally end up in the wrong hands, but even a user of the extension themselves can, for some reason, decide to modify their results statement before they upload it. For that reason, as illustrated in Figure 4.1, the complete circle of a user's data collection does not end with the user's upload.

Before anyone's data can be considered valid to include in the survey, the integrity and the origin of the data first need to be established. We manage to do that by authenticating each user's results statement via a unique digital signature. TLS SCANNER creates that signature at the end of its execution and includes it inside the results statement the user uploads to Dropbox. A digital signature is by nature unique for a specific document and if either the signature or the document is modified their unique pairing ceases to exist. That way, as soon as someone's results are on Dropbox so we can download them, we first aim to authenticate them, via the given digital signature, so we can either proceed to store them alongside the rest of valid results, or otherwise consider them unreliable and immediately discard them from the survey.

Chapter 5

Implementation

Contents

5.1 Communication between the extension's components

The development of a Chrome extension might require one or more files to function. TLS SCANNER FOR GMAIL is developed to work with a background script and two HTML files, each of which makes use of its own JavaScript code. Figure 5.1 illustrates how the different files communicate with one another as soon as someone installs the extension and clicks on its popup button.

By nature, an extension's background script runs from the moment the extension is installed on the browser, but, as Figure 5.1 illustrates, TLS SCANNER's background script does not serve any real purpose on its own. TLS SCANNER's background script means to serve as a bridge between the extension's HTML files, popup.html and scan.html, both of which stand to offer an interface for the user to interact with at different points of the extension's lifetime. The main difference between the two files is that, while popup.html does not need any privileged access to function, scan.html is only meant to come alive after the user allows for it. Specifically, the user triggers the need for scan.html to appear with a click to either of the scanning buttons of the popup screen of Figure 4.2. At the

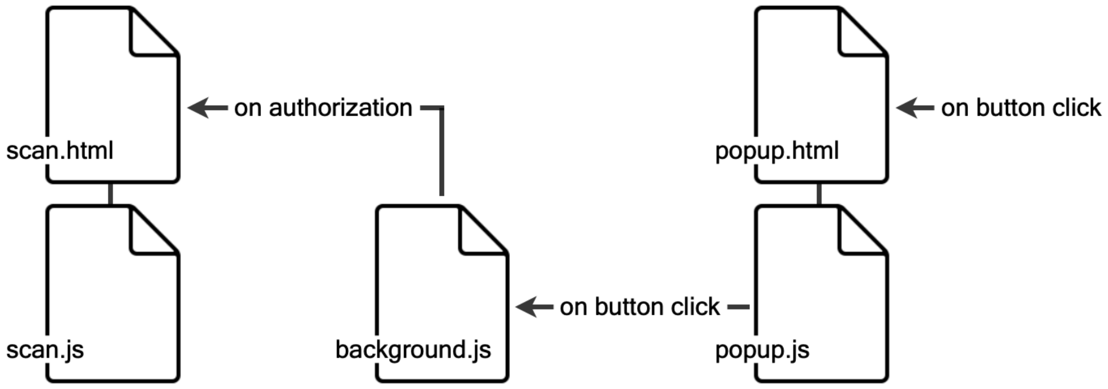


Figure 5.1: Communication between the extension's components

click of either of the two buttons, the background script takes over so as to gain the user's consent before the scanning process can begin.

Whenever the user clicks on the extension's icon, popup.html appears on the user's screen. The extension does this automatically because popup.html is declared inside its manifest file as the default `browser_action` at the icon's button click. When popup.html loads, so does its affiliate script, declared via the HTML `<script>` tag. As illustrated in Algorithm 5.1 of the popup's JavaScript pseudocode, the script follows some considerably simple steps. For so long as the user has the popup open without clicking on any of the scanning options, the script waits. It only awakes at the click of a button, something that is achieved using `addEventListener()`, a function that blocks the flow of execution until a specific event occurs, in this case a button click. Then, when the script unblocks via one of the two buttons, it makes use of `chrome.runtime.sendMessage()` to communicate with the background process the message of the user's scanning request.

From the moment the extension is installed, the background script waits. Just like popup script can get out of its blocking state at the trigger of an event, background script follows a similar approach. Using `chrome.runtime.onMessage.addListener()`, as Algorithm 5.2 illustrates, the background script can only awake at the event of a message, any message from any of the processes that belong in the scope of the extension. At some point, and considering the user decides to click on one of the scanning options, background receives a message coming from the popup script, given there are no other processes other than the two of them running. Background script proceeds to handle the popup's message by first taking on the authorization procedure and then loading scan.html window into a new tab. The scan script may as well handle the authorization procedure after it is loaded, however, background's initiative allows for any action on the user's mailbox to be possible right on the spot, as soon as scan.html opens.

Algorithm 5.1 popup.js

```
procedure SENDMSG(msg)
    chrome.runtime.sendMessage(
        request: msg
    )
end procedure

known var scanAllButton = $< buttonID >$
known var scanByLabelButton = $< buttonID >$

scanAllButton.addEventListener(
    "click", SENDMSG("scanAll")
)
scanByLabelButton.addEventListener(
    "click", SENDMSG("scanByLabel")
)
```

Algorithm 5.2 background.js

```
procedure OPENSCANINTERFACE(token,msg)
    if msg.request = "scanAll" then
        window.open("scan.html?all")
    else
        window.open("scan.html?custom")
end procedure

procedure HANDLEMSG(msg)
    chrome.identity.getAuthToken(
        "interactive": true,
        OPENSCANINTERFACE(token,msg)
    )
end procedure

chrome.runtime.onMessage.addListener(
    HANDLEMSG(msg)
)
```

5.2 User authorization

As line 9 of Algorithm 5.2 illustrates, the extension implements the authorization procedure with the use of `chrome.identity.getAuthToken()`, which uses the information written in `oauth2` section of the manifest file. This information consists of two parts, a `client_id`, unique for the extension and obtained via Google API Console [6], as well as a list of the extension's required `scopes` of access, in this case `www.googleapis.com/auth/gmail.readonly` scope. Given these two arguments, the function communicates with Google Authorization Server to obtain an access token to use in order to access the given scopes. The background script calls this function interactively, which means the user gets to sign into their Google account and manually approve the extension's requested scopes. When this is done, `getAuthToken()` of `chrome.identity` API returns an OAuth2 access token, with which the extension can access the user's mailbox via the Gmail API. The background script receives this token but it does not use it itself. Instead, the background script leaves the access token it acquires aside and proceeds to finally handle popup script's message. This message allows the background script to load scan.html in either of the two scanning modes, illustrated in lines 3 and 5 of Algorithm 5.2.

As soon as scan.html loads and before scanning can begin, it is first necessary for the extension to re-obtain user's authorization access token. This is achieved the same way background script achieves authorization, with the use of `getAuthToken()`, but with no need for user interaction a second time. It is worth noting, however, that an OAuth2 access token does not have an infinite lifetime, so when the extension exceeds approximately one hour of processing, it is essential to re-invoke `getAuthToken()` and obtain a new token to continue with processing if the first token expires. An expired token is not hard to recognise. For each request the extension makes to the Gmail API, it immediately afterwards checks whether the request was or not successful. HTTP `401 Unauthorized` error indicates that the reason of a failed request is lack of authorization, something that can only mean an expired access token and can only be resolved by obtaining a new token.

5.3 How the extension's execution differs for scanning different scopes of a mailbox

The flow of execution of the scan script varies depending on scan.html's page URL, though ultimately the scanning process itself winds up the same. As Algorithm 5.3 illustrates, the extension differentiates the two scanning cases before it proceeds to scanning. At the same time and via a simple switch case, the extension cuts off the flow of execution if the page's URL does not match neither of the two valid URLs that the background script is programmed to open the page with.

To begin with processing, the extension needs first to know what it is that needs processing. The answer to this is obvious and immediate when scan.html is open in `all` mode. In such case, the extension needs to process the user's whole mailbox, the messages of which fall under the `All Mail` label. Unlike the rest of the user's labels, `All Mail` label is not technically a label and hence cannot be identified via a unique ID. It can be identified and accessed, however, via its lack of ID. Given that, as line 26 of Algorithm 5.3 shows, the extension simply needs to push an object with name `All Mail` and an empty ID into its list of pending scan labels before it proceeds to scanning.

When in `custom` mode of execution, it is up to the user to decide which labels to scan before the extension pushes the according list of labels into its pending list. The extension's scan script follows a number of steps to implement that, starting from line 14 of Algorithm 5.3. The script makes its first use of the Gmail API, by invoking the `users.labels.list` method, which returns a list of all the labels in the given user's mailbox. After obtaining this list, the script proceeds to customize the modal screen of Figure 4.3 and present it to the user, before proceeding to block itself via the `addEventListener()` function. The script can only unblock at the user's button click, a trigger that at the same time closes the modal screen and indicates the reach of a decision. When unblocked, the extension merely needs to examine each of the modal's toggle buttons and only push the user's desired labels and their unique IDs into the pending labels list, as illustrated in line 11 of Algorithm 5.3.

5.4 Steps and hurdles of fetching e-mail IDs

As line 32 of Algorithm 5.4 shows, the extension handles each of the labels pushed in the pending list individually and only finishes when it examines them all. The extension follows two steps to process the e-mails under each label. First, it fetches a list of all the e-mail IDs that fall under the given label, and afterwards it performs individual requests for each of the given e-mails. As lines 4 and 23 of Algorithm 5.4 show, both steps use the `users.messages.get` method to acquire their information, only with different parameters. These parameters, nonetheless, distinguish the two calls by their response size.

Fetching a single e-mail's information has a rather predictable and small response size, determined by the fact that we merely care to process specific parts of the metadata of the e-mail, noted in line 5 of Algorithm 5.4. For that reason, a single invocation of `users.messages.get` method for a specific message ID is enough to get this information on the spot. When using the `users.messages.get` method to acquire not one e-mail's information but a label's information, however, the response size is not nearly predictable.

Algorithm 5.3 scan.js initial steps

```
known var token = $< oauth2Token >$
known var modalToggles = $< togglesID >$
known var modalButton = $< buttonID >$

var scanLabels = [ ]
var pageURL = window.location.href

**procedure** PushPendingLabels(labels)
    for every label in labels
        if modalToggles[id].checked then
            scanLabels.push({label[name],label[id]})
**end procedure**

**procedure** FetchScanLabels()
    var labels = LIST gmail.users.labels{
        access_token=token
    }
    DisplayCustomLabelsModal(labels)
    modalButton.addEventListener(
        "click", PushPendingLabels(labels)
    )
**end procedure**

switch (pageURL)
    "$< extensionID >$/scan.html?all":
        scanLabels.push({name:"All Mail",id:$null$})
        BeginProcessing()
        break
    "$< extensionID >$/scan.html?custom":
        FetchScanLabels()
        BeginProcessing()
        break
    default:
        DisplayErrorMsg()
```

Algorithm 5.4 scan.js processing steps

```
var resultsLog = [ ]

procedure SCANMESSAGEBYID(msgID)
    var msg = GET gmail.users.messages[msgID]{access_token=token,
        format=metadata, fields=sizeEstimate, payload(mimeType, headers)}
    var sender, date, secondReceived, hops = 0
    for every field in msg["payload"]["headers"]
        if field["name"] = "From" then sender = field["value"].split("@")[1]
        if field["name"] = "Date" then date = field["value"]
        if field["name"] = "Received" and hops < 2 then
            hops+=1; secondReceived = field["value"]
    var protocol = /by mx.google.com with < protocol > /.exec(secondReceived)
    if !protocol.endsWith("S") then APPENDUNSECUREMAILTOSCREEN(msgID)
    var deliverer = /from < sender > by mx.google.com /.exec(secondReceived)
    var query = fetch dns.google.resolve(deliverer)
    resultsLog.push{sender, date, protocol, query[status], query[domain],
        msg["sizeEstimate"], msg["payload"]["mimeType"]}
end procedure

procedure PROCESSBYLABEL(label)
    var pendingMails = true, nextPage = null
    while pendingMails do
        var msgs = GET gmail.users.messages{access_token=token,
            labelIds=label, pageToken=nextPage}
        if !msgs.hasField("messages") then break
        if msgs.hasField("nextPageToken") then nextPage = msgs["nextPageToken"]
        else pendingMails = false
        for each msg in msgs SCANMESSAGEBYID(msgs["messages"][msg]["id"])
end procedure

procedure BEGINPROCESSING()
    while scanLabels.length > 0 do PROCESSBYLABEL(scanLabels.pop()[id])
    var profile = GET gmail.users.getProfile{access_token=token}
    resultsLog.append{HASH(profile["emailAddress"]), SIGN(resultsLog)}
    DISPLAYRESUTSLINKANDDROPBOXLINK()
end procedure
```

The label may have numerous e-mails under it, and oftentimes this number is dramatically large to fetch in one go. Gmail API only responds with a maximum of 100 e-mails at one invocation of the method. To acquire the full list of a label's e-mails, the extension usually needs to perform multiple fetch requests, each time by filling in the method's `pageToken` parameter with the value of the previous response's `nextPageToken` field. Obviously, when invoking the method for the first time, the `pageToken` parameter is unknown and unnecessary, so it's left empty. By default, omitting the `pageToken` parameter asks the method to fetch the first 100 or less messages, as well as the `nextPageToken` field if necessary. When the `nextPageToken` field ceases to appear in the response body, then the extension can tell that the processing of all the labels' messages of all the page tokens has completed. The `nextPageToken` field is not the only field to indicate that however. The absence of the `messages` field after the first request indicates an empty list of messages that correspond to this label, something that also points out to the extension to stop further processing this label and continue with the next, if there is one.

5.5 The handling of e-mail metadata

Given a list of message IDs, the fetching and processing of individual messages' metadata becomes trivial. Given an e-mail's headers, as shown in line 7 of Algorithm 5.4, the extension examines one header field after the other while recording its fields of interest. The e-mail's size estimate and MIME type come as individual fields of the message's metadata response, so the only information of interest lying in the headers include the sender's e-mail provider, extracted from the `From` field, the `Date` of the e-mail, as well as the second-from-top `Received` field, most likely the one appended by Gmail's MTA. Given the second-from-top `Received` field, the extension attempts first to verify that it is indeed appended by Gmail's MTA and then proceeds to extract additional information from it.

Gmail's MTA appended `Received` field starts by declaring the delivery server the e-mail came `from`, followed by the provider the e-mail was received `by`, which in the case of Gmail's MTA this is `mx.google.com`, and continues with noting the protocol the e-mail was delivered `with`. Given this predictable structure, the extension manages to extract both the delivery server and the transfer protocol from any `Received` field that is indeed received by `mx.google.com`. In the rare case that it is not Gmail's MTA that appended this field, the extension merely chooses to skip over to the next e-mail.

Before updating its results statement with a new summary of an e-mail, the extension uses the name of the delivery server to find the domain it belongs to. To do that, as line 15 of Algorithm 5.4 illustrates, the extension performs a DNS query to Google Public DNS, which responds with the name of the domain and a DNS response status, among others.

Alongside all the previously noted data, the extension notes only the domain name and the DNS status from the query response so as to ultimately create the e-mail's summary, append it to the user's results and immediately proceed with the processing of the next e-mail.

5.6 Preparation and validation of user results

Whether the pending list of labels consists of multiple custom labels or only the `All Mail` label, when the pending list eventually empties, the extension only needs to follow a couple more steps before finishing its execution.

As line 33 of Algorithm 5.4 illustrates, the extension makes use of Gmail API's `users.getProfile` method to acquire a user's profile information, which, among others, includes the user's unique e-mail address. The extension digests the `emailAddress` field using the SHA-256 hash algorithm, which produces a unique identifier of the given user, which, at the same time, does not reveal the user's identity. This is helpful to both ensure user's anonymity and also distinguish results statements that come from the same user so as not to store or process identical e-mail summaries more than once. The extension appends the user's hash to the user's results and goes on to produce a unique HMAC signature of the whole document. This is implemented given a shared secret key, in this case found inside the manifest file, and with the use of `window.crypto.subtle.sign()` function. The extension also appends the signature to the results statement before presenting the user with a link to download the file and another link to upload it to Dropbox, using the `chrome.downloads` API and HTML's `href` attribute respectively. Given the uploaded results statement and the knowledge of the extension's key, we can authenticate any results statement with the use of `window.crypto.subtle.sign()`'s mirror function, `window.crypto.subtle.verify()`, which merely checks if the given results statement matches or fails to match its given HMAC signature, which consequently determines data authenticity, or lack thereof.

Chapter 6

Evaluation

Contents

6.1 General observations on the gathered data

As Table 6.1 illustrates, TLS SCANNER FOR GMAIL processed a total of 16 Gmail mailboxes as part of this thesis, giving us insight into a total of 105,955 e-mails delivered to Gmail's MTA from 1084 distinct e-mail providers. The study's participants are of multiple ages and backgrounds, however, the majority of them are pregraduate and postgraduate Computer Science students. The evaluation of the users' data consists of two main pillars, examining the TLS usage of individual e-mails as well as the TLS usage of their respective e-mail providers.

Though only a handful of mailboxes were examined, as the user histogram of Figure 6.1 shows, there is an apparent deviation in the encrypted to unencrypted e-mail deliveries for each mailbox, with one mailbox's unencrypted deliveries being as low as 40%,

	e-mails	mailboxes	e-mail providers
total	105,955	16	1084
encrypted deliveries	92.71%	85.93%$_{\text{avg}}$	82.20%
unencrypted deliveries	7.29%	14.07%$_{\text{avg}}$	29.61%
overlap	none	none	11.81%

Table 6.1: Encrypted and unencrypted e-mail distributions for, (i) the total of inspected e-mails, (ii) an average user mailbox and (iii) between the different e-mail providers

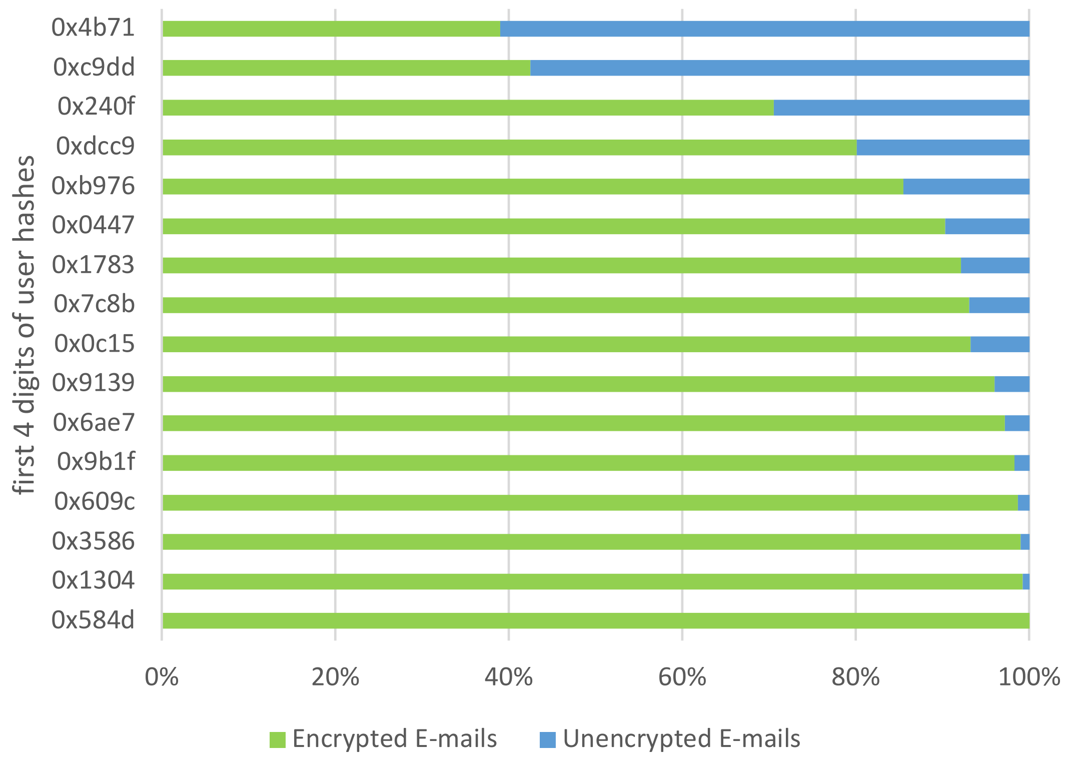


Figure 6.1: Ratio of encrypted to unencrypted e-mails per user mailbox

though with the significant majority of mailboxes indicating an encrypted delivery portion above 90%.

As Table 6.1 shows, the overall portion of unencrypted e-mails corresponds to a bit more than 7% of the total processed e-mails, whereas this number doubles for an average user mailbox. Looking back at Figure 6.1, the latter number does not represent the generally descent portions of encrypted deliveries in most mailboxes, but only arises as a result of a few mailboxes with deviating high rates of unencrypted deliveries.

The portion of unencrypted e-mail deliveries can moreover differentiate the behaviour of e-mail providers, splitting them into those that deliver encrypted and those that deliver unencrypted e-mails, corresponding to 82.20% and 29.61% respectively. Unlike individual e-mail deliveries per se, which we can classify as either encrypted or unencrypted but not both, the sum of the e-mail providers' deliveries indicates an overlap of 11.81%, which in turn indicates that 11.81% of the e-mail providers deliver both encrypted and unencrypted e-mails. This allows us to classify e-mail providers into not two but three categories, those consistent in their TLS usage, those consistent in their lack of TLS usage, and those inconsistent in their TLS usage altogether, either through time or between different mailboxes. Given that, subtracting the overlapping 11.81% portion of TLS inconsistent providers from Table 6.1's providers' deliveries reveals that, out of the 82.20% of providers that send encrypted e-mails, only 70.39% of them do this consistently, whereas out of the total of 29.61% providers guilty for unencrypted e-mails, it is in fact 17.80% out of the total providers that constantly follow this pattern.

6.2 TLS usage of individual e-mails

Regardless of their e-mail providers behaviour, different e-mails may share some common patterns with one another, which may potentially explain or hint at their TLS usage. This section compares the encrypted and unencrypted e-mails based on four factors extracted from their headers,

i their delivery year,

ii their size,

iii their MIME type and

iv their e-mail provider's DNS status.

6.2.1 E-mail deliveries comparisons per year

We begin by examining the different e-mails in regards to their year of delivery.

Figure 6.2(a) displays how encrypted and unencrypted e-mails are distributed throughout the years. The figure shows that more than 25% of all unencrypted e-mail deliveries correspond to the year 2015, while the majority of encrypted e-mails are more recent, corresponding to 2020. The worryingly large volume of unencrypted e-mails in 2015, especially in regards to the previous years, might potentially hint at the need for Gmail's red padlock update in 2016, a feature that not only would make users sceptical on their mailboxes' security, but at the same time it would most likely obligate many e-mail providers to reassess their use of e-mail encryption. This could in turn explain why the volume of encrypted e-mails starts to raise after 2015, while unencrypted e-mail numbers raise up to 2015 before they begin to drop.

Figure 6.2(b) reinforces this observation, presenting the encrypted to unencrypted ratio for each year. Though up until 2014, the portion of unencrypted e-mails compared to encrypted ones is more than 50%, the volume of unencrypted e-mails starts to fade in the next years, and after 2016 it even settles down to less than 5%, a still alarming number yet drastically better compared to the magnitude of unencrypted e-mails before 2016.

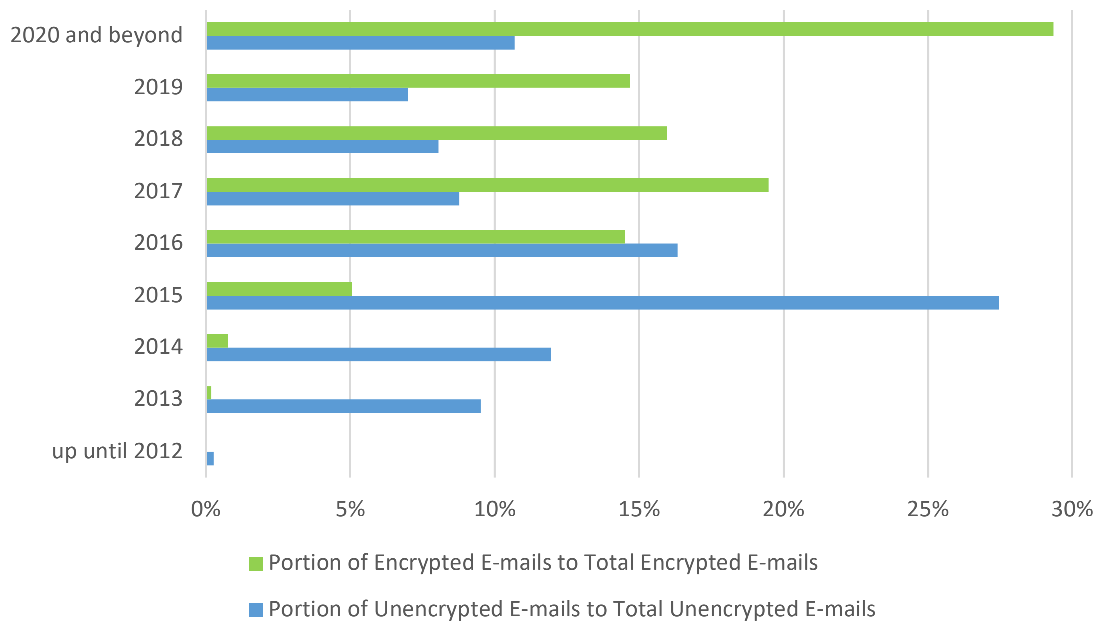
6.2.2 E-mail deliveries comparisons per e-mail size

We can moreover classify and examine e-mails based on their size, searching for a possible correlation between an e-mail's size and its TLS usage.

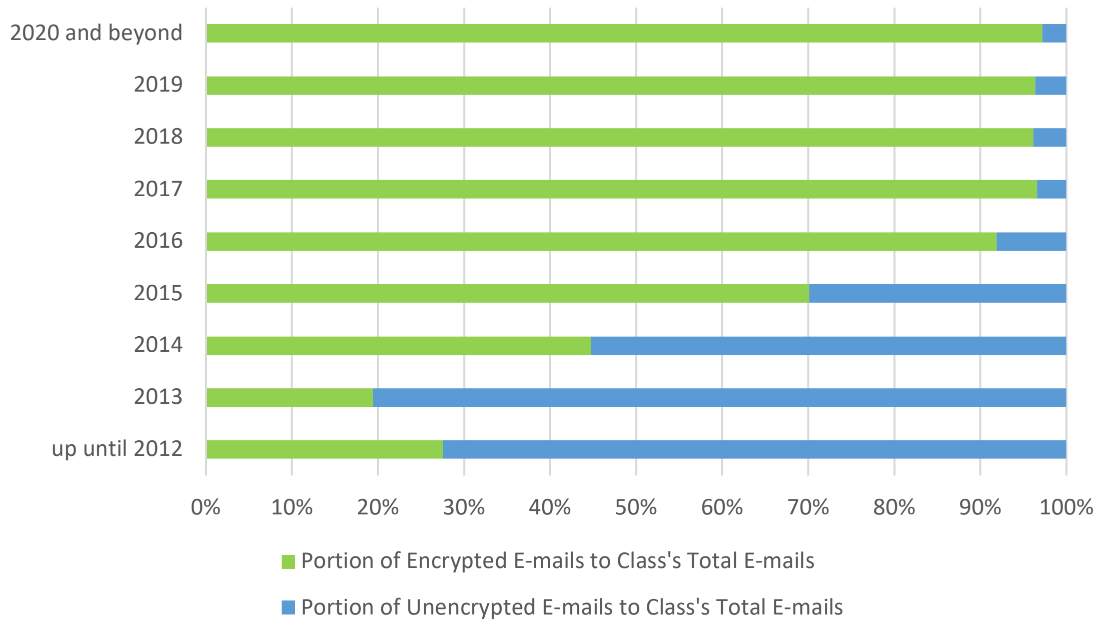
Figure 6.3(a) shows a uniform distribution of the encrypted e-mails in regards to the unencrypted ones. The majority of encrypted e-mails, corresponding to around 45% of their total, belong in the size spectrum [20,40)KB, while the majority of unencrypted e-mails, corresponding to around 37% of their total, belong in the same size spectrum. The smallest size spectrum, with e-mails of size less than 20KB, represents the second most popular spectrum for both encrypted and unencrypted e-mails, with around 15% of all encrypted e-mails and around 26% of all unencrypted e-mails. The distributions of e-mails among the rest of the size classes are very similar, with a deviation that never exceeds 4%.

The overall distribution of encrypted and unencrypted e-mails, illustrated in Figure 6.3(a), does not seem to hint at the e-mails' TLS usage, as both encrypted and unencrypted e-mails have their own range of e-mail sizes, quite similar to each other. Their similarity also appears in Figure 6.3(b) of the ratios of encrypted to unencrypted e-mails for each size. The slight deviations for each size displayed in Figure 6.3(a) re-appear in Figure 6.3(b), displaying a generally low rate of unencrypted e-mails, below 8% in most cases, with a few exceptions of unencrypted e-mails fluctuating between 12% and 28%. These exceptions all correspond to the most apparent deviations of Figure 6.3(a), where unencrypted e-mails surpass the encrypted ones, therefore justifying their slightly bigger portion in their ratio compared to other sizes.

It seems that neither Figure 6.3(a) nor Figure 6.3(b) display any sort of correlation between e-mails' sizes and their use of TLS, therefore rejecting the idea that an e-mail's security could potentially depend on its size. The most worrying ratio of unencrypted e-mails appears for the largest e-mail sizes, but the overall volume of e-mails corresponding to this size is not big enough to support that the largest e-mails are more likely than the smallest ones to be delivered without encryption, though it could be a possibility. What the two figures do support is rather the absence of any correlation between e-mail sizes and their security, which makes the smallest e-mails just as prone to lacking encryption as the largest ones.

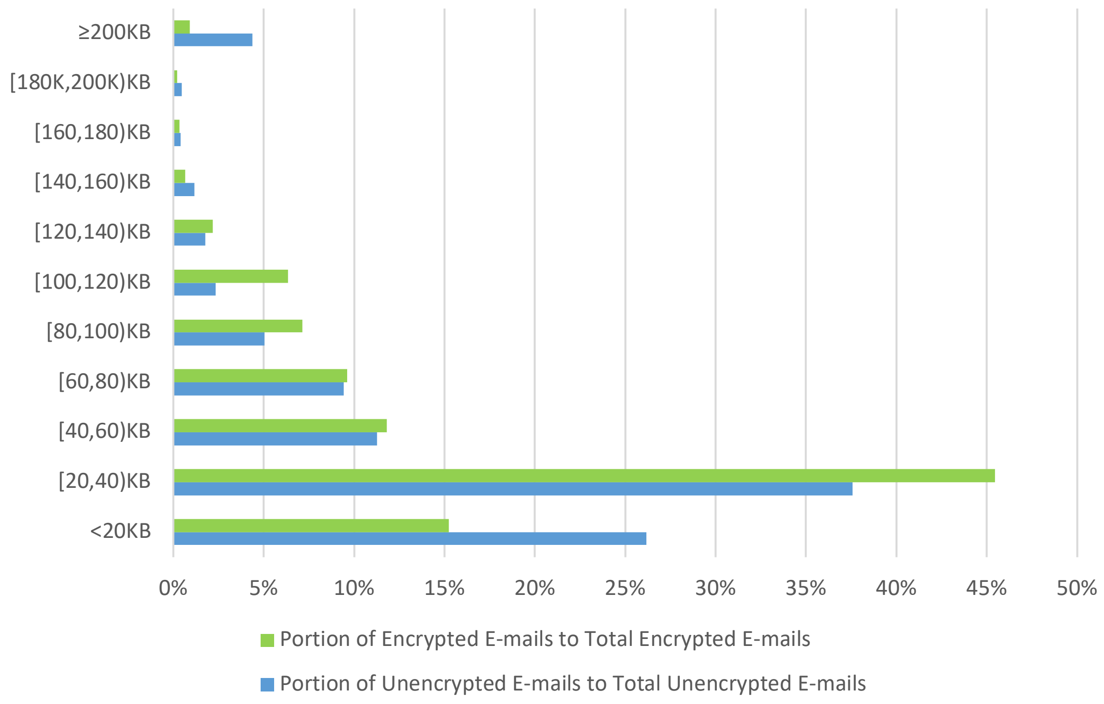


(a) Distribution of encrypted and unencrypted e-mails among years

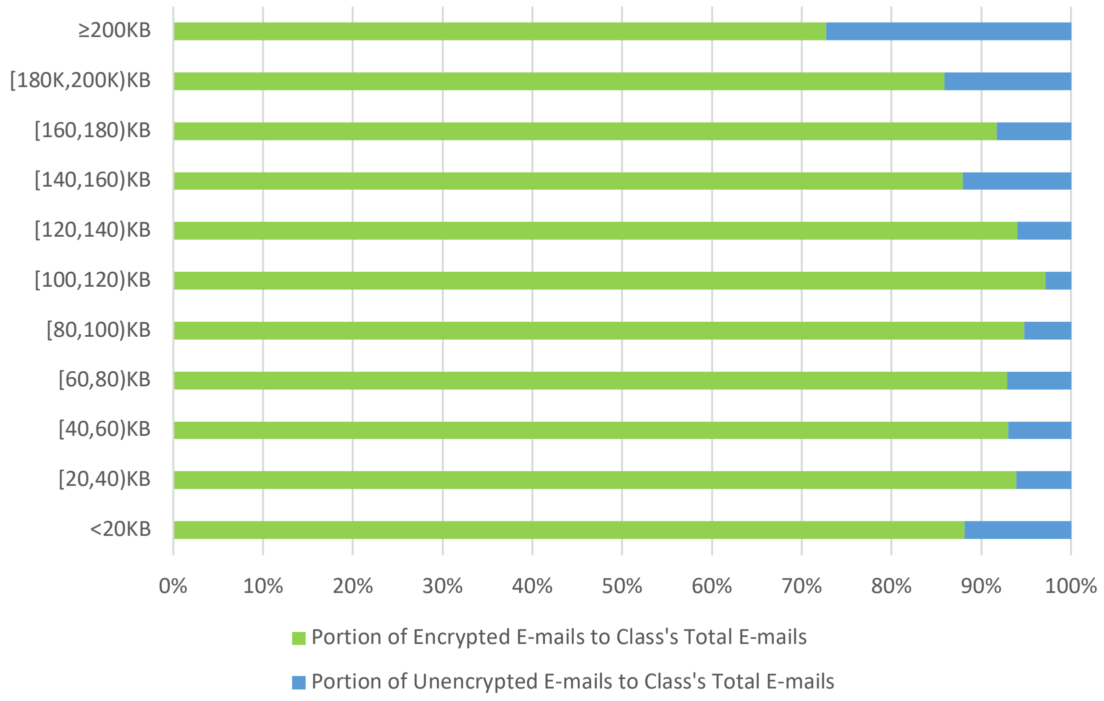


(b) Ratio of encrypted to unencrypted e-mails per year

Figure 6.2: E-mail observations per year



(a) Distribution of encrypted and unencrypted e-mails among e-mail sizes



(b) Ratio of encrypted to unencrypted e-mails per e-mail size

Figure 6.3: E-mail observations per e-mail size

6.2.3 E-mail deliveries comparisons per MIME type

Another parameter worth examining when comparing the profiles of encrypted and unencrypted e-mails is the type of data lying within the e-mails, in other words their data format, a piece of information presented in an e-mail's headers via its MIME type.

Figure 6.4(a) shows that more than half of both encrypted and unencrypted e-mails are of MIME type `multipart/alternative`, corresponding to almost 85% for encrypted e-mails and a bit less than 65% for unencrypted ones. Though a number of different MIME types appear in Figure 6.4(a), up to 95% of the total of encrypted e-mails are made up of only `multipart/alternative` and `text/html` MIME types, with the remaining 5% appearing very sparingly. For the case of unencrypted e-mails, however, the specific two MIME types only make up for 80% of the total of unencrypted e-mails, which leaves a remaining 20% of e-mails distributed among the remaining MIME types.

As illustrated in Figure 6.4(b), the two most prevailing MIME types for encrypted deliveries, that is `multipart/alternative` and `text/html`, correspond to the two best ratios of encrypted to unencrypted e-mails, with less than 10% of their total e-mails being unencrypted. This is not the case for the ratios of the rest of the MIME types, with their unencrypted portions of e-mails fluctuating from 15% and up to 100%.

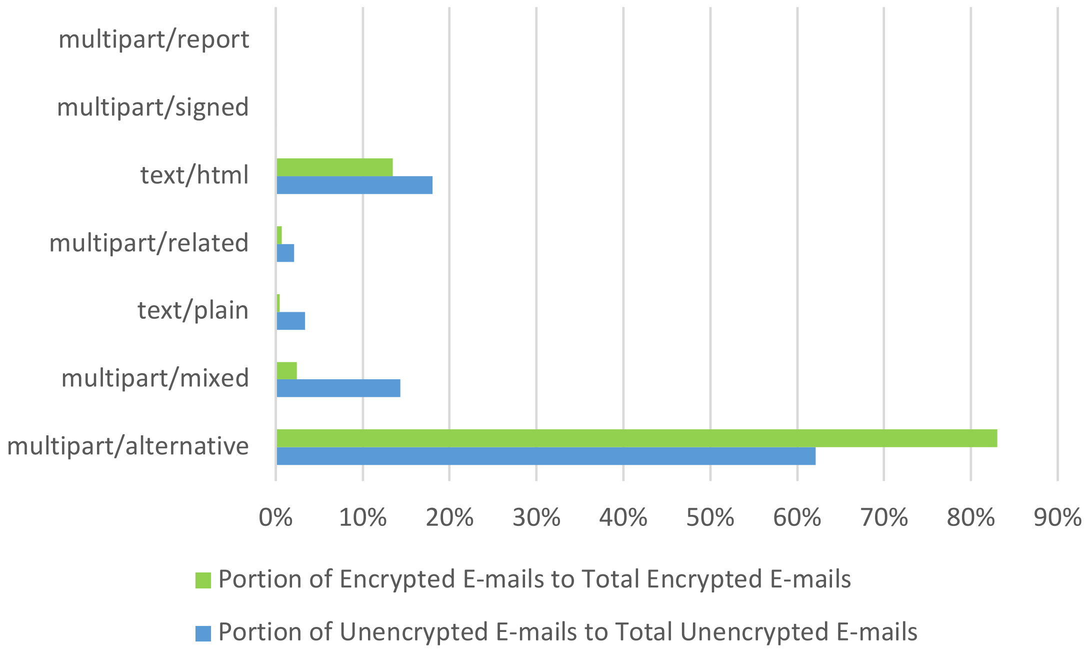
Though Figure 6.4(a) presents a vast majority of unencrypted e-mails with MIME types `multipart/alternative` and `text/html`, Figure 6.4(b) shows that the ratios of encrypted to unencrypted e-mails for the specific MIME types are not quite alarming. The most alarming ratios are more likely to appear for MIME types other than these two. Though `multipart/alternative` and `text/html` MIME types still have their share of unencrypted e-mails, it seems that their popularity among encrypted e-mails makes their unencrypted instances pretty rare, something that in turn makes the e-mails of the rest of the MIME types, other than these two, more likely to come without encryption.

6.2.4 E-mail deliveries comparisons per providers' DNS status

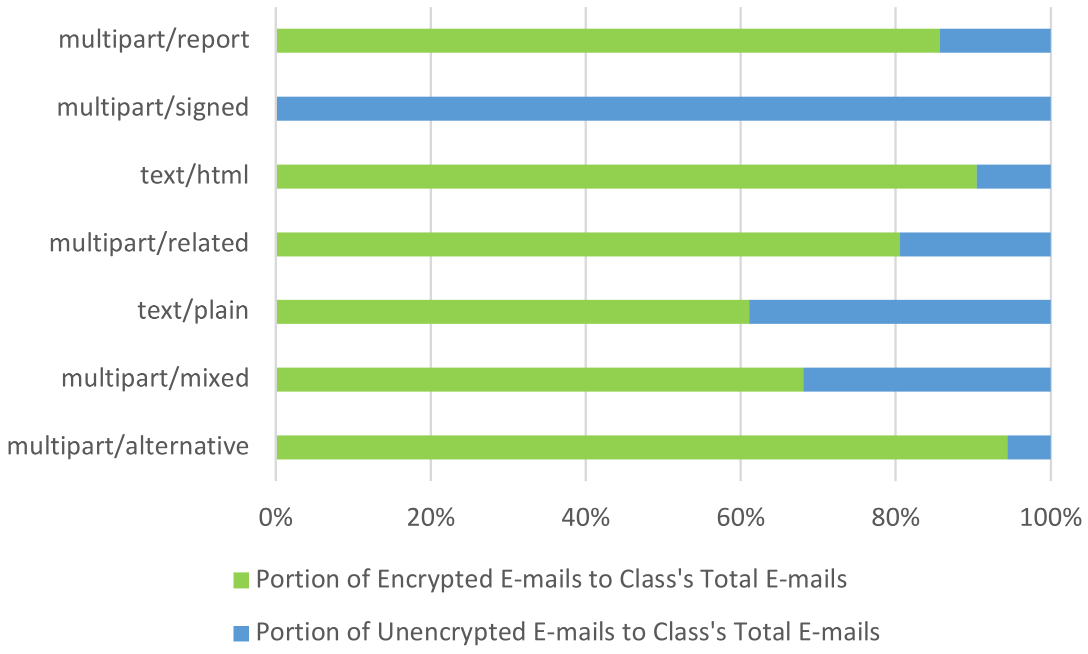
A final parameter to consider when looking at e-mail profiles is the DNS status of their e-mail providers.

The response of a DNS query can indicate a number of things, including format or communication errors, among others. For this study, only `DNS Status 0` and `DNS Status 3` were examined, the former indicating a living instance of the specified server within its domain, and the latter indicating that the specified server is no longer functional within its domain.

The motivation for examining the DNS status of a delivery server is to check whether the TLS usage in an e-mail and hence the TLS usage applied or omitted by the e-mail's provider has anything to do with the server's current state. Perhaps, servers found to have

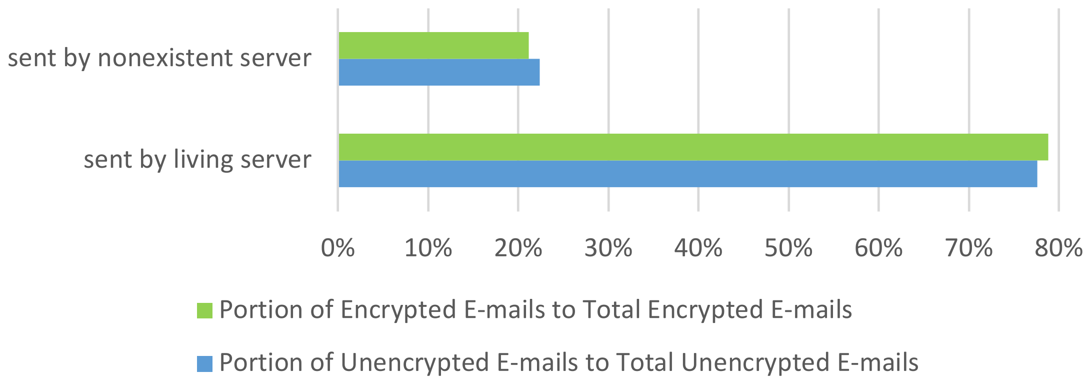


(a) Distribution of encrypted and unencrypted e-mails among MIME types

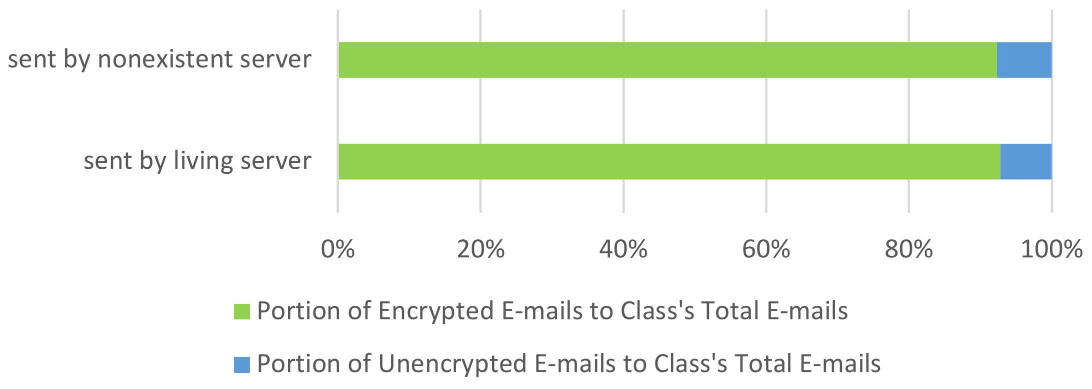


(b) Ratio of encrypted to unencrypted e-mails per MIME type

Figure 6.4: E-mail observations per MIME type



(a) Distribution of encrypted and unencrypted e-mails among providers' DNS statuses



(b) Ratio of encrypted to unencrypted e-mails per providers' DNS status

Figure 6.5: E-mail observations per providers' DNS status

delivered unencrypted e-mails are now out of use, while servers found to have delivered encrypted e-mails are more likely to still be functioning.

Figures 6.5(a) and 6.5(b) present a similar image of the distributions and ratios of e-mails sent by nonexistent and living servers. The volume of encrypted e-mails sent by nonexistent servers is not very different than the unencrypted volume of e-mails sent by nonexistent servers, which in turn makes the encrypted and unencrypted portions of e-mails sent by living servers also very similar. The ratios of unencrypted to encrypted e-mails is not very different between the two cases either, corresponding to around 8% of unencrypted e-mails for both cases.

The state of an e-mail's delivery server does not seem to play any part in the e-mail's TLS usage, as both Figure 6.5(a) and Figure 6.5(b) support. A server currently out of use is just as likely to have delivered an encrypted e-mail as it is to have delivered an unencrypted one, with the e-mail's use of TLS having nothing to do with its provider's current state.

6.3 TLS usage of the various e-mail providers

Though the DNS status and hence the state of e-mail providers does not seem to relate in any way to the provider's use of encryption, this does not mean that there are not still other factors to consider when looking over the behaviour of e-mail providers. Rather than looking at individual e-mails' correlations, we may as well examine the profiles of the different e-mail providers and compare them with each other.

6.3.1 A definition of TLS inconsistency

As already explained, we can classify e-mail providers given their TLS usage and their consistency, or lack thereof. We measure the TLS inconsistency of a provider by computing the frequency of how many times the provider's behaviour swaps from delivering encrypted e-mails to delivering unencrypted e-mails or vice versa, whether these swaps occur between different users or even within the same mailbox. The higher a provider's protocol swap frequency, the more inconsistent a provider is considered, whereas for the case of providers with protocol swap frequency equal to zero, their overall unencrypted rate of e-mails can reveal whether their consistency is that of always using or always omitting TLS in their transits.

6.3.2 Correlations between providers of unencrypted deliveries

This section aims to examine the possibility of various correlations between the profiles and behaviours of the providers found to have delivered at least one unencrypted e-mail. Specifically, we examine the different providers in regards to,

i their portion of unencrypted deliveries,

ii their consistency, as this arises from their protocol swap frequency,

iii their popularity among users and

iv their activity, as this arises from their total number of e-mail deliveries.

We begin by examining the whole range of providers that send unencrypted e-mails, regardless of whether their behaviour is consistent or inconsistent.

Table 6.2(a) presents one medium and many weak correlations between the providers guilty of unencrypted deliveries. With a medium, negative correlation of -0.529 between providers' unencrypted deliveries and their popularity among users, Table 6.2(a) indicates that the more popular a provider is the less likely it is to be delivering high portions of unencrypted e-mails, which is surely a comforting observation, as the most popular of

	> 0	
	unencrypted e-mail ratio	protocol swap frequency
protocol swap frequency	-0.374	-
user popularity	-0.529	-0.027
delivered e-mails	-0.294	-0.057

(a) Correlations between e-mail providers with positive unencrypted e-mail ratio

		> 0
	unencrypted e-mail ratio	protocol swap frequency
protocol swap frequency	0.152	-
user popularity	-0.386	-0.346
delivered e-mails	-0.237	-0.196

(b) Correlations between e-mail providers with positive protocol swap frequency

Table 6.2: Correlations between providers of unencrypted deliveries

providers tend to be less guilty of frequently omitting TLS in their transits. The remaining correlations are not strong enough to present any relationship between two factors that could justify a provider's levels of unencrypted deliveries or its levels of inconsistency, and unfortunately this does not change for Table 6.2(b)'s correlations either.

Table 6.2(b) displays how Table 6.2(a)'s correlations change after confining the whole list of providers guilty of unencrypted deliveries to only those that are TLS inconsistent. Table 6.2(b) reveals some even weaker correlations than before, with the correlations with the providers' unencrypted e-mail ratios dropping even more and the correlations with the providers' protocol swap frequencies raising noticeably but not enough, making the case of TLS inconsistent providers all the more unjustifiable and unsettling.

6.3.3 An overview of some of the most prevailing e-mail providers

This section aims to present a more targeted overview of e-mail providers by discussing the top 10 of,

i the most popular providers,

ii the most active providers,

iii the most unsecure providers, in terms of their unencrypted e-mail ratio, and

iv the most inconsistent providers, in terms of their protocol swap frequency.

The goal of this section is double. For one, going over specific providers' profiles can perhaps tell us more on the behaviour of specific groups of e-mail providers than the so far calculated correlations revealed. Moreover, this section stands as a reference point for participants of this study or otherwise to gain an idea of what the profiles of many known e-mail providers look like.

	User Popularity	Delivered E-mails	Unencrypted E-mail Ratio	Protocol Swap Frequency
google.com.	100.00%	30151	0.00%	0.00%
protection.outlook.com.	100.00%	532	0.00%	0.00%
smtp-out.amazonses.com.	93.75%	3579	1.23%	0.08%
smtp-out.us-west-2.amazonses.com.	87.50%	1547	0.00%	0.00%
outbound-mail.sendgrid.net.	81.25%	450	0.22%	0.22%
sparkpostmail.com.	81.25%	1190	0.08%	0.17%
smtp-out.eu-west-1.amazonses.com.	81.25%	2628	0.30%	0.11%
mcdlv.net.	81.25%	1067	17.62%	0.09%
facebook.com.	81.25%	7799	1.31%	0.09%
mailgun.net.	81.25%	689	0.00%	0.00%

Table 6.3: Top 10 most popular e-mail providers

	User Popularity	Delivered E-mails	Unencrypted E-mail Ratio	Protocol Swap Frequency
google.com.	100.00%	30151	0.00%	0.00%
twitter.com.	50.00%	10983	0.95%	0.41%
facebook.com.	81.25%	7799	1.31%	0.09%
smtp-out.amazonses.com.	93.75%	3579	1.23%	0.08%
smtp-out.eu-west-1.amazonses.com.	81.25%	2628	0.30%	0.11%
quora.com.	56.25%	2220	0.00%	0.00%
emsmtp.us.	43.75%	2172	0.00%	0.00%
linkedin.com.	56.25%	1935	0.00%	0.00%
smtp-out.us-west-2.amazonses.com.	87.50%	1547	0.00%	0.00%
mailjet.com.	62.50%	1337	0.00%	0.00%

Table 6.4: Top 10 most active e-mail providers

Tables 6.3 and 6.4 present the most popular and the most active providers among the study's participants respectively, highlighting with red colour any providers that appear between both the most popular and the most active ones.

By looking at Tables 6.3 and 6.4, with a single exception of a provider with 17.62% ratio of unencrypted deliveries, it appears that neither the most popular nor the most active providers are guilty of unencrypted e-mail ratios that surpass 1.5%, nor are they significantly inconsistent in their TLS usage.

As Table 6.5 illustrates, the providers that are the most guilty for their lack of TLS usage are actually providers with medium to low popularity and activity. Table 6.5 reveals a list of providers with ranging popularities and activities, from some very unpopular and inactive providers to some alarmingly more popular and active ones.

Though the most unsecure providers' behaviour as a whole is upsetting, as every e-mail of every context delivered by them always lacked encryption, at the very least their behaviour is predictable. Table 6.6 presents the list of the most inconsistent and hence

	User Popularity	Delivered E-mails	Unencrypted E-mail Ratio	Protocol Swap Frequency
neoquin.com.	6.25%	688	100.00%	0.00%
cs.ucy.ac.cy.	50.00%	637	100.00%	0.00%
bigfishgames.com.	6.25%	472	100.00%	0.00%
play.com.	6.25%	216	100.00%	0.00%
csrv.ucy.ac.cy.	56.25%	104	100.00%	0.00%
newsgroup.kathimerini.com.cy.	6.25%	56	100.00%	0.00%
authoritynutrition.com.	6.25%	43	100.00%	0.00%
delivery.net.	12.50%	38	100.00%	0.00%
vsmail-cluster.visualsoft.co.uk.	6.25%	37	100.00%	0.00%
em.ea.com.	25.00%	36	100.00%	0.00%

Table 6.5: Top 10 most unsecure e-mail providers

	User Popularity	Delivered E-mails	Unencrypted E-mail Ratio	Protocol Swap Frequency
olympiagroup.gr.	12.50%	6	50.00%	83.33%
pbs.org.	6.25%	9	33.33%	55.56%
sendlabs.com.	12.50%	2	50.00%	50.00%
netsuite.com.	12.50%	2	50.00%	50.00%
graphisoft.hu.	6.25%	2	50.00%	50.00%
woozworld.com.	6.25%	2	50.00%	50.00%
acemserv.com.	6.25%	2	50.00%	50.00%
jawbone.com.	12.50%	4	25.00%	50.00%
chelseafc.com.	6.25%	91	54.95%	42.86%
srv2.de.	18.75%	43	51.16%	41.86%

Table 6.6: Top 10 most inconsistent e-mail providers

unpredictable of providers, a list of providers with considerably low popularities and some definitely low levels of activity. Just by looking at those providers' overall activities, the apparent lack of information gathered for most of them can possibly explain their high levels of inconsistency, as these may arise not from the providers' unpredictability as a whole but as a mere result of not enough information gathered on them. It is therefore highly possible that collecting some more information on these providers would not only raise their overall levels of activity but could also change their TLS inconsistency levels altogether.

Chapter 7

Related Work

Contents

7.1 Studies that analyse e-mail patterns

Back in 2007, in their paper, *A study of e-mail patterns* [11], S. Shah and B. D. Noble discuss how they used their department's servers to observe millions of e-mails over the course of months, capturing and analysing several e-mail parameters, among which lying some common to our study's parameters of interest, such as message sizes and attachment content types.

Contrary to our study's focus on e-mail security, S. Shah and B. D. Noble's goal did not target the TLS usage in e-mail transits. Instead, their motivation lay in collecting enough data to improve the modelling of e-mail workloads used for benchmarking and systems engineering.

In 2009, in their paper, *Behavioral Profiles for Advanced Email Features* [10], T. Karagiannis and M. Vojnovic present their own study of analysing e-mail patterns, aiming to explore the design possibilities of advanced e-mail features by investigating the presence and significance of any factors that impact the probability and/or time of e-mail replies.

The work of T. Karagiannis and M. Vojnovic is different than ours not only in regards to their study's motive but in regards to their data of interest as well. Our work takes a strictly anonymous approach that does not investigate or even address the study's participants, but the case is different for T. Karagiannis and M. Vojnovic's study, whose target on e-mail replies relies not only in inspecting e-mail properties but also the behavioral profiles that describe their study's users, as these emerge from the users' behaviour in processing and handling their e-mails.

7.2 Studies that target TLS in e-mail communication

In 2016, in their paper, *TLS in the Wild: An Internet-wide Analysis of TLS-based Protocols for Electronic Communication* [8], R. Holz et al. present an experimental investigation of the TLS usage in communication over e-mail and chat.

Though the study's motive is similar to our project's, the means via which R. Holz et al. targeted the security of the Internet messaging infrastructure were different than our own. Rather than scanning volunteering mailboxes of e-mails like TLS SCANNER FOR GMAIL extension was designed to do, R. Holz et al. used active servers scans and passive client connections monitoring to collect their parameters of interest and uncover the degree of secure communications over unsecure ones. Similarly to our study's results, R. Holz et al. also revealed an alarming volume of unencrypted e-mails, however they did not examine how the profiles of these e-mails or their deliverers can potentially hint at this phenomenon.

In the recent 2020, in their paper, *What Email Servers Can Tell to Johnny: An Empirical Study of Provider-to-Provider Email Security* [9], G. Kambourakis et al. assess the security between MTA-to-MTA communications, as this was measured with the use of a custom designed tool composed of a set of security evaluation tests.

Our work also targets the security between e-mail providers, however, our study does not target the behaviour of e-mail providers as a whole. Our focus instead, lies in how individual e-mail providers communicate with Gmail's MTA specifically.

Finally, via a Google Transparency Report [2], Google sheds light on its inbound and outbound encrypted e-mail traffic originating back from 1998 and up to today.

Similarly to our study's results, emerging only from a sample of processed e-mails, Google's Transparency Report also presents a clear milestone in the rising volume of unencrypted e-mail traffic in 2015, while it moreover presents the volume of encrypted e-mail traffic corresponding to the most active domains by region, with some of the most prevailing domains recorded by Google appearing within our sampling as well. Google's Transparency Report, nonetheless, does not record or present patterns on the characteristics of individual e-mails or how these patterns link to their security.

Chapter 8

Conclusion

To measure the TLS usage in today's e-mail transits does not entail a one and only solution. As part of this project, we designed TLS SCANNER FOR GMAIL, a Google Chrome extension meant to scan the e-mail headers within any Gmail mailbox, collecting brief summaries of information to describe the profile of individual e-mail headers, while at the same time classifying the e-mails as either delivered with or without encryption, by going over the protocols via which different e-mails arrive to Gmail's MTA with.

The information gathered via distributing TLS SCANNER FOR GMAIL revealed a generally descent but not nearly flawless image of the distribution of encrypted and unencrypted deliveries, while at the same time it disclosed a worrying number of providers inconsistent in their use of TLS.

Our analysis on individual e-mails' profiles rejected any correlation between e-mails' TLS usage with both their size and their providers' DNS status, but it did reveal a declining trend in the volumes of unencrypted e-mails delivered throughout the years, while it also hinted that some e-mail MIME types are more likely to come without encryption than others. Our focus on the profiles of the different e-mail providers moreover disclosed how providers' popularity and activity relate to their volumes of unencrypted deliveries and their levels of inconsistency, indicating that the most popular and active providers are more likely to be responsible for none or very few unencrypted deliveries, while the most inconsistent providers are more likely to be of very low popularity and activity, which in turn makes their worrying levels of inconsistency less likely to come as result of unpredictability but more likely as a result of insufficient information gathered on them.

TLS SCANNER FOR GMAIL can be distributed to more users and gain a higher and more defining volume of information as a result, considering the extension is adjusted to comply with all the necessary guidelines of the General Data Protection Regulation (GDPR). Possible further future work can include enhancing the extension so that it provides a better interface to its users. Specifically, the extension can be updated to offer its users the choice to run it only for a limited time and/or a specific size of e-mails rather than having to wait indefinitely. In addition to that, the extension can offer its users the choice to be completely automated, capable to handle the complete circle of scanning and data delivery at the users' consent.

Bibliography

[1] Documentation for Chrome extensions developers. `https://developer.chrome.com/docs/extensions/`.

[2] Email encryption in transit - Google Transparency Report. `https://transparencyreport.google.com/safer-email/overview`.

[3] Gmail API. `https://developers.google.com/gmail/api`.

[4] Gmail Help: Check the security of your emails. Answer in `https://support.google.com/mail/`.

[5] Gmail Help: Trace an email with its full headers. Answer in `https://support.google.com/mail/`.

[6] Using OAuth 2.0 to Access Google APIs. `https://developers.google.com/identity/protocols/oauth2`.

[7] What is TLS & How Does it Work? | ISOC Internet Society. `https://www.internetsociety.org/deploy360/tls/basics/`.

[8] R. Holz et al. TLS in the Wild: An Internet-wide Analysis of TLS-based Protocols for Electronic Communication. In *NDSS Symposium 2016*. Copyright 2016 Internet Society.

[9] G. Kambourakis et al. What Email Servers Can Tell to Johnny: An Empirical Study of Provider-to-Provider Email Security. IEEE Access, 2020.

[10] T. Karagiannis and M. Vojnovic. Behavioral Profiles for Advanced Email Features. Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009.

[11] S. Shah and B. D. Noble. A study of e-mail patterns. In *Wiley InterScience (www.interscience.wiley.com)*, pages 1515–1538. Copyright 2007 John Wiley & Sons, Ltd.

Appendix A

The extension’s manifest.json is presented in this appendix.

```
{
  "manifest_version": 2,
  "name": "TLS␣Scanner␣for␣Gmail",
  "description": "Scan␣your␣Gmail␣for␣any␣unencrypted␣emails",
  "version": "0.1",
  "icons": {
    "16": "icon16.png",
    "32": "icon32.png",
    "48": "icon48.png",
    "128": "icon128.png"
  },
  "browser_action": {
    "default_icon": "icon128.png",
    "default_popup": "popup.html"
  },
  "permissions": [
      "identity",
    "downloads",
      "*://*.google.com/*",
    "tabs",
    "https://apis.google.com/*"
  ],
  "content_scripts": [
    {
      "matches": ["<all_urls>"],
      "js": ["popup.js"]
    }
  ],
  "background": {
    "scripts": ["background.js"]
  },
  "oauth2": {
      "client_id": "420203891442-2nlco587r0qkpa3jbk6rf14hhe0oh21d.apps.
          googleusercontent.com",
      "scopes": [
        "https://www.googleapis.com/auth/gmail.readonly"
      ]
  },
  "content_security_policy": "script-src␣'self'␣https://apis.google.com␣https://maxcdn.
      bootstrapcdn.com␣https://ajax.googleapis.com␣https://cdnjs.cloudflare.com;␣object
      -src␣'self'",
  "key": "MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAzwmrtHKgVgZB3Uj5462TH7yM/4
      xDZpAC6TZT6wDvBCZ4j+CW+5HTsB5WS1sl6yJUQrf+
      nq1E0rsJyt0oGSv3h7N73PVABVeFsTKI4gQSSvxkC5VZwP+
        LWAweEflpqGylTkPzfp25VAtep6TvqJ4TsVNsOfGNTtSEgMbw7DNnfR5bVIawgrN5E/
        ZHwydWqsNMDHWjfdLpn3d9TaNMjfihSXdAoBs2p1ZKRQEe0XnRNUz5/WV7aORpbTwJ/
        TCajNbJVW4OQFTI6uBRTD90ak5Xj95txrFEDI6p+tE/eZjkZ2cenV31TytqR2/
        tndSAky1DRZdktggJfPyTnKSbgcq7swIDAQAB"
}
```

Appendix B

The extension's background.js is presented in this appendix.

```
//wait for popup's request to authenticate and open scan.html
chrome.runtime.onMessage.addListener(
  function(message, sender, sendResponse) {
    if (message.request === "authenticate_All") {
      chrome.identity.getAuthToken({'interactive': true},
      async function (token) {
        window.open('scan.html?all');
      });
    }
    else if (message.request === "authenticate_ByLabel") {
      chrome.identity.getAuthToken({'interactive': true},
      async function (token) {
        window.open('scan.html?custom');
      });
    }
  }
);
```

Appendix C

The extension's popup.html is presented in this appendix.

```
<!DOCTYPE html>
<html>
<style>

:root {
  --white: #FEFEFE;
  --whitegray: #E7E7E7;
  --bluegray: #f2f2f7;
  --red: #eb333a;
  --lightred: #fc4f55;
}

body {
  width: 280px;
  background: var(--white);
  font-family: "Helvetica";
  padding-left: 10px;
  padding-right: 10px;
  padding-top: 10px;
  padding-bottom: 20px;
}

.title {
  font-weight: 100;
  font-size: 20px;
  text-align: center;
  margin-top: -10px;
}

.logo {
  margin-bottom: 0px;
  text-align: center;
}

.introduction {
  text-align: justify;
  font-size: 13px;
  margin-left: 10px;
  margin-right: 10px;
}

.scan {
  text-align: center;
}
```

```

.scan button {
  color: white;
  text-align: center;
  font-size: 13px;
  padding: 10px 50px;
  position: center;
  background-color: #eb333a;
  border-style: solid;
  border-radius: 23px;
  border-color: #f2f2f2;
  cursor: pointer;
}

.scan button:focus {
  outline: 0;
}

.scan button:hover {
  background-color: var(--lightred);
}

</style>
<body>

  <p class="logo"><img src="logo.png" style="width:80px;height:80px;"></p>

  <div class="title">
    <p>TLS Scanner for Gmail</p>
  </div>

  <div class="introduction">
    <p>TLS Scanner is designed to scan your Gmail for any unencrypted e-mails you
        received.<p>
    <p>You may choose to scan your whole mailbox or specific labels of your choice.
        Whichever option you choose, you will be prompted into a new tab that will
        gradually be filled with links to your unencrypted e-mails. If this is the
        first time you are running the extension, you will first be asked to authorize
        with your Google account before the tab opens.</p>
  </div>

  <div class="scan">
    <br>
    <button id="scanAll_Button">Scan All</button>
    <br>
    <button id="scanByLabel_Button">Scan by Label</button>
    <script type="text/javascript" src="popup.js"></script>
  </div>

</body>
</html>
```

Appendix D

The extension's popup.js is presented in this appendix.

```
var scanAll_Button = document.getElementById("scanAll_Button");
if (scanAll_Button) {
  scanAll_Button.addEventListener("click", messageBackground_All);
}
var scanByLabel_Button = document.getElementById("scanByLabel_Button");
if (scanByLabel_Button) {
  scanByLabel_Button.addEventListener("click", messageBackground_ByLabel);
}

function messageBackground_All() {
  chrome.runtime.sendMessage({request: "authenticate_All"});
  return;
}
function messageBackground_ByLabel() {
  chrome.runtime.sendMessage({request: "authenticate_ByLabel"});
  return;
}
```

Appendix E

The extension's scan.html is presented in this appendix.

```
<!DOCTYPE html>
<html>
<style>

:root {
  --white: #FEFEFE;
  --whitegray: #E7E7E7;
  --bluegray: #f2f2f7;
  --red: #eb333a;
  --darkred: #ffcccc;
  --blue: #2153F6;
  --lightpink: #fcefef;
  --lightred: #fc4f55;
}

body {
  font-family: "Helvetica";
  text-align: justify;
  font-size: 15px;
  padding-left:150px;
  padding-right:150px;
  padding-top:50px;
  padding-bottom:50px;
}
.title {
  font-weight: 200;
  font-size: 40px;
  text-align: center;
  margin-top: 5px;
}

.logo {
  margin-bottom: 0px;
  text-align: center;
}

.modal-content {
  margin-bottom: 28%;
  margin-top: 28%;
}

* {
    padding: 0;
    box-sizing: border-box;
```

```
}

a {
    color: inherit;
    text-decoration: none;
}

ol {
  list-style: none;
}

label {
    cursor: pointer;
    margin:0px;
}

[type="checkbox"] {
    position: absolute;
  left: -9999px;
}

.switches {
    width: 80%;
  margin: 0px auto 0;
  margin-top: -15px;
  margin-bottom: -15px;
    border-radius: 0px;
    color: var(--black);
    background: var(--white);
}

.switches li {
    position: relative;
    counter-increment: switchCounter;
}

.switches li:not(:last-child) {
  margin-top: 6px;
  border-bottom: 1px solid var(--whitegray);
  margin-bottom: 6px;
    border-bottom: 1px solid var(--whitegray);
}

.switches li::before {
    content: counter(switchCounter);
    position: absolute;
    top: 50%;
    left: -30px;
    transform: translateY(-50%);
    font-size: 2rem;
    font-weight: bold;
    visibility: hidden;
}

.switches label {
    display: flex;
    align-items: center;
    justify-content: space-between;
    padding: 8px;
}

.switches span:last-child {
    position: relative;
    width: 50px;
    height: 26px;
    border-radius: 15px;
    box-shadow: inset 0 0 5px rgba(0, 0, 0, 0.1);
    background: var(--whitegray);
    transition: all 0.3s;
}

.switches span:last-child::before,
.switches span:last-child::after {
    content: "";
    position: absolute;
}

.switches span:last-child::before {
    left: 2px;
    top: 2px;
    width: 22px;
    height: 22px;
    background: var(--white);
    border-radius: 50%;
    z-index: 1;
    transition: transform 0.3s;
}

.switches span:last-child::after {
  top: 50%;
    right: 8px;
    width: 12px;
    height: 12px;
    transform: translateY(-50%);
    background-size: 12px 12px;
}

.switches [type="checkbox"]:checked + label span:last-child {
  background: var(--blue);
}

.switches [type="checkbox"]:checked + label span:last-child::before {
  transform: translateX(24px);
}

.switches [type="checkbox"]:checked + label span:last-child::after {
  width: 14px;
  height: 14px;
  left: 8px;
  background-size: 14px 14px;
}

.modal-header {
  margin: 0 auto;
```

```css
}

.modal-title {
  text-align: center;
}

.modal-footer {
  margin: 0 auto;
}

.modal-button {
  text-align: center;
}

.mail-buttons {
  text-align: right;
  margin-bottom: 3px;
  margin-top: 3px;
}

.modal-button button {
  color: white;
  text-align: center;
  font-size: 15px;
  padding: 10px 50px;
  position: center;
  background-color: #eb333a;
  border-style: solid;
  border-radius: 23px;
  border-color: #f2f2f2;
  cursor: pointer;
}

.modal-button button:focus {
  outline: 0;
}

.mail-buttons button {
  color: white;
  text-align: right;
  font-size: 12px;
  padding: 5px 25px;
  position: right;
  background-color: #eb333a;
  border-style: solid;
  border-radius: 23px;
  border-color: #f2f2f2;
  cursor: pointer;
}

.mail-buttons button:focus {
  outline: 0;
}

.modal-button button:hover {
  background-color: var(--lightred);
}
```

```

.mail-buttons button:hover {
  background-color: var(--lightred);
}

.modal-body {
  background-color: var(--white);
  margin: 2px;
}

.modal-body::-webkit-scrollbar {
  -webkit-appearance: none;
  width: 6px;
}

.modal-body::-webkit-scrollbar-thumb {
  border-radius: 5px;
  background-color: rgba(0,0,0,.5);
  -webkit-box-shadow: 0 0 1px rgba(255,255,255,.5);
}

.processBox {
  position: relative;
  display: block;
}
.percentageBox {
  position: absolute;
  height: 100%;
  text-align: center;
  width: 100%;
}
.percentageBox:before {
  content: '';
  display: inline-block;
  height: 100%;
  vertical-align: middle;
}
.percentage {
  display: inline-block;
  font-family: "Helvetica";
  font-size: 12px;
}

.loaderBox {
  text-align: center;
  font: ;
}

.step1 {
    display: flex;
}

.unselectableArea {
  -webkit-touch-callout: none;
  -webkit-user-select: none;
  -khtml-user-select: none;
  -moz-user-select: none;
  -ms-user-select: none;
  font-size: 15px;
  text-align: justify;
}

</style>

<body>

  <p class="logo"><img src="logo.png" style="width:100px;height:100px;"></p>
  <div class="title">
    <p>TLS Scanner for Gmail</p>
  </div>

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/
        bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></
        script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min
        .js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"
        ></script>

  <div id="modalBox" class="modal␣fade">
    <div class="modal-dialog␣modal-dialog-scrollable" role="document">
      <div class="modal-content">
        <div class="modal-header" style="border:none;">
          <div class="modal-title">The following labels were found in your mailbox.<br>
              Which of them do you want to scan?</div>
        </div>
        <div id="modal-body" class="modal-body">
          <div toggles>
          <ol class="switches" id="switches">
            <!-- labels will be displayed here -->
          </ol>
          </div>
        </div>
        <div class="modal-footer" style="border:none;">
          <div class="modal-button">
            <button id="modalButton" type="button" data-dismiss="modal">Scan now</
                button>
          </div>
        </div>
      </div>
    </div>
  </div>

  <div id="submitBox" class="modal␣fade">
    <div class="modal-dialog␣modal-dialog-scrollable" role="document">
      <div class="modal-content" style="overflow:␣auto;">

        <div class="unselectableArea">
          <div id="modal-body" class="modal-body">

            <div>Thank you for running TLS Scanner for Gmail!</div>

            <hr id="alertLine">

            <div>Any results displayed on your screen is only for you to see. The
                extension has kept track of its own results logfile which does not
                include any of your private information. In order for these results to
                reach us, follow the steps below.</div>
            <p style="font-size:3px;">

            <div class="step1" id="step1" style="right-padding:3px;">Step 1: Download
                your results logfile<div style="color:white;">.</div><a id="
                downloadLink" class="downloadLink" href="#">results.log</a></div>

            <div class="step1note" id="step1note" style="font-size:10px;">Note: results
                .log only includes information on your e-mail headers, not your e-mail
                contents.</div>
            <p style="font-size:3px;">

            <div class="step2" id="step2" style="display:none;">Step 2: Upload your
                results to <a class="dropboxLink" id="dropboxLink" href="https://www.
                dropbox.com/request/CuRM83pXIdoxAmGlVHN6" target="_blank">TLS Scanner
                Dropbox</a></div>
            <div class="step2-invisible" id="step2-invisible" style="color:␣var(--
                whitegray);">Step 2: Upload your results to TLS Scanner Dropbox</div>

            <div class="step2note" id="step2note" style="font-size:10px;display:none;">
                Note: If you have a Dropbox account and want to hide your identity,
                then sign out before you upload your results. Dropbox will ask for your
                 name and e-mail to complete the upload if you are not signed in. Your
                e-mail will be used by Dropbox to verify your upload and won't␣be␣
                visible␣to␣us.␣Your␣name,␣however,␣will␣appear␣next␣to␣your␣filename,␣
                so␣write␣something␣invalid␣in␣the␣name␣field,␣like␣a␣dot␣(.)␣or␣dash␣
                (-),␣to␣ensure␣your␣anonymity.</div>
␣␣␣␣␣␣␣␣␣␣␣␣<div␣class="step2note-invisible"␣id="step2note-invisible"␣style="font-size
    :10px;color:␣var(--whitegray);">Note:␣If␣you␣have␣a␣Dropbox␣account␣and␣want␣to␣
    hide␣your␣identity,␣then␣sign␣out␣before␣you␣upload␣your␣results.␣Dropbox␣will␣ask␣
    for␣your␣name␣and␣e-mail␣to␣complete␣the␣upload␣if␣you␣are␣not␣signed␣in.␣Your␣e-
    mail␣will␣be␣used␣by␣Dropbox␣to␣verify␣your␣upload␣and␣won't be visible to us. Your
     name, however, will appear next to your filename, so write something invalid in
    the name field, like a dot (.) or dash (-), to ensure your anonymity.</div>

            <p style="font-size:3px;">

            <div class="step3" id="step3" style="display:none;">Step 3: Click Finish</
                div>
            <div class="step3-invisible" id="step3-invisible" style="color:␣var(--
                whitegray);">Step 3: Click Finish</div>

            <hr id="alertLine">

            <div class="mail-buttons">
              <button id="exitButton" type="button" data-dismiss="modal" style="display
                  :none;">Finish</button>
              <button id="waitButton" type="button" style="background-color:var(--
                  lightred);" disabled>Finish</button>
            </div>

          </div>

```

```
            </div>

          </div>
        </div>
      </div>

    <script type="text/javascript" src="https://apis.google.com/js/client.js?onload=
        callbackFunction"></script>
    <script type="text/javascript" src="scan.js"></script>

</body>
</html>
```

Appendix F

The extension's scan.js is presented in this appendix.

```
function APPEND_ERROR_PARAGRAPH(par, theEnd) {
  HIDE_LOADER();
  var paragraph = document.createElement('p');
  paragraph.style["text-align"] = "center";
  paragraph.style["color"] = "var(--red)";
  paragraph.style["border-style"] = "solid";
  paragraph.style["border-color"] = "var(--darkred)";
  paragraph.style["border-width"] = "0.4px";
  paragraph.style["background"] = "var(--lightpink)";
  paragraph.style["padding"] = "10px";
  var text = document.createTextNode(par);
  paragraph.appendChild(text);
  document.body.appendChild(paragraph);
  if (!theEnd) {
    APPEND_LOADER();
  }
  return;
}

function APPEND_LEFT_PARAGRAPH(par) {
  var paragraph = document.createElement('p');
  var text = document.createTextNode(par);
  paragraph.style["text-align"] = "left";
  paragraph.appendChild(text);
  document.body.appendChild(paragraph);
  return;
}

function APPEND_CENTER_PARAGRAPH(par) {
  var paragraph = document.createElement('p');
  paragraph.style["text-align"] = "center";
  var text = document.createTextNode(par);
  paragraph.appendChild(text);
  document.body.appendChild(paragraph);
  return;
}

function APPEND_RIGHT_PARAGRAPH(par) {
  var paragraph = document.createElement('p');
  paragraph.style["text-align"] = "right";
  paragraph.style["font-size"] = "10px";
  var text = document.createTextNode(par);
  paragraph.appendChild(text);
  document.body.appendChild(paragraph);
```

```
    return;
}

function APPEND_LINK(label, from, subject, date, protocol, link, provider) {

    APPEND_LINE();

    // from
    APPEND_LEFT_PARAGRAPH('FROM: ' + from);

    // icon, subject, link
    var p, a, textNode;
      p = document.createElement('p');
      textNode = document.createTextNode('SUBJECT: ');
      p.appendChild(textNode);
    a = document.createElement('a');
    textNode = document.createTextNode(subject);
    a.appendChild(textNode);
    a.href = link;
    a.target = "_blank";
    p.appendChild(a);
    document.body.appendChild(p);

    // date
    APPEND_LEFT_PARAGRAPH('DATE: ' + date);

    // protocol, label, provider
    APPEND_RIGHT_PARAGRAPH('LABEL: ' + label);
    APPEND_RIGHT_PARAGRAPH('PROTOCOL: ' + protocol);
    APPEND_RIGHT_PARAGRAPH('PROVIDER: ' + provider);

    return;
}

function APPEND_LINE() {
    var hr = document.createElement('hr');
    hr.align = "right";
    hr.style["height"] = "1px";
    hr.style["width"] = "80%";
    hr.style["border"] = "0";
    hr.style["background-image"] = "linear-gradient(200deg, gray, transparent)";
    document.body.appendChild(hr);
}

async function APPEND_STATISTICS(processingStatus) {
    if (processingStatus !== "success") {
      APPEND_ERROR_PARAGRAPH('Processing stopped unexpectedly. ' + processingStatus);
      HIDE_LOADER();
      APPEND_CENTER_PARAGRAPH(totalUnencrypted + " out of " + totalReceived + " received 
          emails were found unencrypted");
    }
    else {
      HIDE_LOADER();
      APPEND_CENTER_PARAGRAPH(totalUnencrypted + " out of " + totalReceived + " received 
          emails were found unencrypted");

      if (messagesTotal == 0) return;

      // 1. prepare logfile body
      var logBody = 'Errors␣count:␣' + errorsCount + '\nClient␣hash:␣' + clientHash + '\n
          \n';
      let i;
      var totalDomains = resultsAsObjects.length;
      for (i=0; i<totalDomains; i++) {
        let obj = resultsAsObjects[i];
        logBody += '(\"' + obj.lastDomain + '\",\"' + obj.dnsStatus + '\",\"' + obj.date
            + '\",\"' + obj.security + '\",\"' + obj.mimeType + '\",\"' + obj.
            sizeEstimate + '\",\"' + obj.senderDomain + '\")\n';
      }

      // 2. create hmac symmetric key and hmac of logfile body
      var hmacSignature;

      function str2ab(str) {
        const buf = new ArrayBuffer(str.length);
        const bufView = new Uint8Array(buf);
        for (let i=0, strLen=str.length; i<strLen; i++) {
          bufView[i] = str.charCodeAt(i);
        }
        return buf;
      }

      const manifestKey = "
          MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAzwmrtHKgVgZB3Uj5462TH7yM/4
          xDZpAC6TZT6wDvBCZ4j+CW+5HTsB5WS1sl6yJUQrf+
          nq1E0rsJyt0oGSv3h7N73PVABVeFsTKI4gQSSvxkC5VZwP+
          LWAweEflpqGylTkPzfp25VAtep6TvqJ4TsVNsOfGNTtSEgMbw7DNnfR5bVIawgrN5E/
          ZHwydWqsNMDHWjfdLpn3d9TaNMjfihSXdAoBs2p1ZKRQEe0XnRNUz5/WV7aORpbTwJ/
          TCajNbJVW4OQFTI6uBRTD90ak5Xj95txrFEDI6p+tE/eZjkZ2cenV31TytqR2/
          tndSAky1DRZdktggJfPyTnKSbgcq7swIDAQAB";
      const binaryDerString = window.atob(manifestKey);
      const binaryDer = str2ab(binaryDerString);

      var symmetricKey;

      let cryptoKeyPromise = await window.crypto.subtle.importKey(
          "jwk", //can be "jwk" or "raw"
          {   //this is an example jwk key, "raw" would be an ArrayBuffer
              kty: "oct",
              k: buf2hex(binaryDer),
              alg: "HS256",
              ext: true,
          },
          {   //algorithm options
              name: "HMAC",
              hash: {name: "SHA-256"},
              //length: 256, //optional, if you want your key length to differ from the
                  hash function's block length
          },
          false, //key not extractable
          ["sign", "verify"]
      )
      .then(function(key){
          symmetricKey = key;
      });

      let encodedData = new TextEncoder().encode(logBody);

      let signPromise = await window.crypto.subtle.sign(
        {
          name: "HMAC",
        },
        symmetricKey,
        encodedData
      ).then(function(signature){
        hmacSignature = buf2hex(new Uint8Array(signature));
      });

      // 3. prepare logfile content
      var logResults;
      var timeNow = new Date();
      logResults = 'tls_scanner_results.log␣with␣hmac␣signature␣' + hmacSignature + '\ncreated␣at␣+␣' + timeNow.toUTCString() + '\n';
      logResults += '\n----RESULTS␣BODY␣BELOW.␣BEWARE␣THAT␣ANY␣ALTERATIONS␣OF␣BELOW␣SECTION␣CAN␣BE␣DETECTED␣WITH␣HMAC␣SIGNATURE␣AND␣WILL␣IMMEDIATELY␣DISCLUDE␣YOUR␣RESULTS␣FROM␣THE␣SURVEY----\n\n';
      logResults += logBody;

      $('#submitBox').modal({
        backdrop: 'static'
      });

      var downloadLink = document.getElementById("downloadLink");
      if (downloadLink) {
        downloadLink.addEventListener("click", function() {

          var blob = new Blob([logResults], {type: "binary/octet-stream"});
          var fileURL = URL.createObjectURL(blob);
          chrome.downloads.download({
            url: fileURL,
            filename: 'tls_scanner_results.log',
            saveAs: true
          });

          var step2_inv = document.getElementById("step2-invisible");
          step2_inv.style.display = "none";
          var step2note_inv = document.getElementById("step2note-invisible");
          step2note_inv.style.display = "none";

          var step2 = document.getElementById("step2");
          step2.style.display = "";
          var step2note = document.getElementById("step2note");
          step2note.style.display = "";

          var dropboxLink = document.getElementById("dropboxLink");
          if (dropboxLink) {
            dropboxLink.addEventListener("click", function() {

              var step3_inv = document.getElementById("step3-invisible");
              step3_inv.style.display = "none";

              var step3 = document.getElementById("step3");
              step3.style.display = "";

              var waitButton = document.getElementById("waitButton");
              waitButton.style.display = "none";

              var exitButton = document.getElementById("exitButton");
              exitButton.style.display = "";

            });
          }
        });
      }
    }
    return;
  }

  function APPEND_MODAL(labels, total) {
    //customize modal
    for (var i=0; i<total; i++) {
      let thisLabel = labels.shift();
      let label_name = thisLabel["name"];
      let label_id = "toggleSwitch_" + thisLabel["id"];
      let label_type = thisLabel["type"];

      if (label_type === "system") {
        if (label_name!=="INBOX" && label_name!=="SPAM" && label_name!=="TRASH" && label_name!=="UNREAD" && label_name!=="STARRED" && label_name!=="IMPORTANT") {
          continue; //ignore all other system labels
        }
      }
      var element = $('#switches');
      var toggle_button = "<li>\
  ␣␣␣␣␣␣␣␣␣␣␣␣<input␣type='checkbox'␣id='" + label_id + "'>\
  ␣␣␣␣␣␣␣␣␣␣␣␣␣␣<label␣for='" + label_id + "'>\
  ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<span>" + label_name + "</span>\
  ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<span></span>\
  ␣␣␣␣␣␣␣␣␣␣␣␣␣␣</label>\
  ␣␣␣␣␣␣␣␣␣␣␣␣</li>";
      element.append(toggle_button);
    }

    //display modal
    $('#modalBox').modal({
      backdrop: 'static'
    });
    return;
  }

  function APPEND_LOADER() {
    HIDE_LOADER();
    processBox = document.createElement('div');
    processBox.className = "processBox";
    percentageBox = document.createElement('div');
    percentageBox.className = "percentageBox";
    percentage = document.createElement('div');
    percentage.className = "percentage";
    if (messagesTotal == 0) {
      percentage.innerHTML = "0%";
    }
    else {
      percentage.innerHTML = Math.round(processedTotal/messagesTotal) + '%';
    }
    percentageBox.appendChild(percentage);
    processBox.appendChild(percentageBox);
    loaderBox = document.createElement('div');
    loaderBox.className = "loaderBox";
    loadingImage = document.createElement('img');
    loadingImage.src = "loader.gif";
    loadingImage.style = "height:65px;width:65px;";
    loaderBox.appendChild(loadingImage);
    processBox.appendChild(loaderBox);
    document.body.append(processBox);
  }

  function HIDE_LOADER() {
    if (processBox) {
      processBox.style.display = "none";
    }
    return;
  }

  //START OF EXECUTION

  var totalReceived = 0;
  var totalUnencrypted = 0;

  var scanLabels = [];
  var scanLabelsCopy = [];
  var scanLabel = '';
  var scanLabelId = '';

  var emailIds = [];
  var unsecureDomains = [];

  var pendingAppends = [];

  var pendingEmails = true;
  var pendingPageToken = '';

  var authToken = '';

  var processBox, percentageBox, percentage, loaderBox, loadingImage;
  var processedTotal = 0;
  var messagesTotal = 0;
  var messagesTotalFetched = false;

  var resultsAsObjects = [];
  var clientHash = '';

  var errorsCount = 0;

  var thisUrl = window.location.href;
```

```

switch (thisUrl) {
  case "chrome-extension://eedgfjpgakgkecokjonmcbhchfneimhk/scan.html?all":
    scanLabel = 'All␣Mail';
    beginProcessing();
    break;
  case "chrome-extension://eedgfjpgakgkecokjonmcbhchfneimhk/scan.html?custom":
    fetchLabelsAndDisplayModal();
    break;
  default:
    APPEND_ERROR_PARAGRAPH("\"" + thisUrl + "\"␣request␣is␣not␣valid.␣TLS␣Scanner␣
        terminated.", true);
}

function buf2hex(buffer) {
  return Array.prototype.map.call(new Uint8Array(buffer), x => ('00' + x.toString(16)).
      slice(-2)).join('');
}

async function digestMessage(message) {
  var msgUint8 = new TextEncoder().encode(message); //encode as utf-8
  var hashBuffer = await crypto.subtle.digest('SHA-256', msgUint8); //hash the message
  var hashArray = Array.from(new Uint8Array(hashBuffer)); //convert buffer to byte
      array
  var hashHex = buf2hex(hashArray); //convert bytes to hex string
  return hashHex;
}

function hex2buf(hex) {
    var byteArray = [];
    for (var i=0; i<hex.length; i+=2) {
      byteArray.push(parseInt(hex.substr(i, 2), 16));
    }
  var uint8Array = new Uint8Array(byteArray.length);
  for(var i=0; i<uint8Array.length; i++) {
    uint8Array[i] = byteArray[i];
  }
    return uint8Array;
}

function fetchLabelsAndDisplayModal() {
  chrome.identity.getAuthToken({'interactive': true},
  async function (token) {
    authToken = token;

    // fetch labels
    var url = "https://www.googleapis.com/gmail/v1/users/me/labels?access_token=" +
        authToken;

    let response = await fetch(url).catch((error) => {
      APPEND_ERROR_PARAGRAPH('Some␣error␣occured␣trying␣to␣fetch␣your␣labels:␣' + error
          , true);
      return;
    });

    if (!response.ok) {
      let responseStatus = response.status;
      var error = 'HTTP␣' + responseStatus + '␣error␣trying␣to␣fetch␣your␣labels';
      APPEND_ERROR_PARAGRAPH(error, true);
      return;
    }

    var labels = await response.json().catch((error) => {
      APPEND_ERROR_PARAGRAPH('Some␣error␣occured␣trying␣to␣fetch␣your␣labels:␣' + error
          , true);
      return;
    });;

    var labelsWithIds = []; var labelsWithIds_copy = [];
    var total = labels["labels"].length;
    var i;
    for (i=0; i<total; i++) {
      labelsWithIds.push({name: labels["labels"][i]["name"], id: labels["labels"][i]["
          id"], type: labels["labels"][i]["type"]});
      labelsWithIds_copy.push({name: labels["labels"][i]["name"], id: labels["labels"][
          i]["id"], type: labels["labels"][i]["type"]});

    }

    APPEND_MODAL(labelsWithIds_copy, total);

    var modalButton = document.getElementById("modalButton");
    if (modalButton) {
      modalButton.addEventListener("click", function() {

        for (i=0; i<total; i++) {
          let thisLabel = labelsWithIds.shift();
          let label_name = thisLabel["name"];
          let label_id = thisLabel["id"];

          let element = document.getElementById("toggleSwitch_" + label_id);
          if (!element) { continue; } //element does not exist
          if (!element.checked) { continue; }
          scanLabels.push({name: label_name, id: label_id});
        }

        scanLabelsCopy = [...scanLabels]; //needed to fetch message totals
        prepareNextLabel();
        beginProcessing();
      });
    }
    return;

  });
}

function prepareNextLabel() {
  if (scanLabels.length>0) {
    let thisLabel = scanLabels.shift();
    scanLabel = thisLabel["name"];
    scanLabelId = '&labelIds=' + thisLabel["id"];
    pendingEmails = true;
    pendingPageToken = '';
  }
  else {
    pendingEmails = false;
  }
}

function inboxProcessingWithNewAuthToken() {

  chrome.identity.getAuthToken({'interactive': true},
  async function (token) {
    authToken = token;

    if (!messagesTotalFetched) {
      let totalLabelsToBeFetched = 1;
      let url = '';
      if (thisUrl === "chrome-extension://eedgfjpgakgkecokjonmcbhchfneimhk/scan.html?all") {
        url = 'https://www.googleapis.com/gmail/v1/users/me/profile?access_token=' + authToken;
      }
      else {
        totalLabelsToBeFetched = scanLabelsCopy.length;
        if (scanLabelsCopy.length>0) {
          //prepare first url
          let thisLabel = scanLabelsCopy.shift();
          url = 'https://www.googleapis.com/gmail/v1/users/me/labels/' + thisLabel["id"] + '?access_token=' + authToken;
        }
      }

      let i;
      for (i=0; i<totalLabelsToBeFetched; i++) {
        if (i>0) {
          //prepare next url
          let thisLabel = scanLabelsCopy.shift();
          url = 'https://www.googleapis.com/gmail/v1/users/me/labels/' + thisLabel["id"] + '?access_token=' + authToken;
        }
        //fetch url
        let response = await fetch(url).catch((error) => {
          APPEND_STATISTICS('Some␣error␣occured␣trying␣to␣fetch␣messages␣total:␣' + error);
          return;
        });

        if (!response.ok) {
          let responseStatus = response.status;
          APPEND_STATISTICS('HTTP␣' + responseStatus + '␣error␣trying␣to␣fetch␣messages␣total');
          return;
        }

        var info = await response.json().catch((error) => {
          APPEND_STATISTICS('Some␣error␣occured␣trying␣to␣fetch␣messages␣total:␣' + error);
          return;
        });;

        if (!info.hasOwnProperty("messagesTotal")) {
          APPEND_STATISTICS('Some␣error␣occured␣trying␣to␣fetch␣messages␣total:␣
              Important␣json␣property␣missing');
          return;
        }
        messagesTotal += info["messagesTotal"];
      }
      messagesTotalFetched = true;
    }

    async function inboxProcessing() {
      if (!pendingEmails) {
        prepareNextLabel();
        if (!pendingEmails) { //pendingEmails is false still
          APPEND_STATISTICS("success");
          return;
        }
      }

      APPEND_LOADER();

      var url;
      if (pendingPageToken === '') { //first request
        url = "https://www.googleapis.com/gmail/v1/users/me/messages?access_token=" +
            authToken + scanLabelId;
      }
      else {
        url = "https://www.googleapis.com/gmail/v1/users/me/messages?access_token=" +
            authToken + scanLabelId + "&pageToken=" + pendingPageToken;
      }

      let response = await fetch(url).catch((error) => {
        APPEND_STATISTICS('Some␣error␣occured␣trying␣to␣fetch␣your␣messages:␣' + error)
            ;
        return;
      });

      if (!response.ok) {
        HIDE_LOADER();
        let responseStatus = response.status;
        if (responseStatus == 401) {
          inboxProcessingWithNewAuthToken();
          return;
        }
        APPEND_STATISTICS('HTTP␣' + responseStatus + '␣error␣trying␣to␣fetch␣your␣
            messages');
        return;
      }

      var msgs = await response.json().catch((error) => {
        APPEND_STATISTICS('Some␣error␣occured␣trying␣to␣fetch␣your␣messages:␣' + error)
            ;
        return;
      });;

      if (!msgs.hasOwnProperty("messages")) {
        HIDE_LOADER();
        pendingEmails = false;
        prepareNextLabel();
        inboxProcessing();
        return;
      }

      var i;
      var headers = [];
      var mimeTypes = [];
      var sizeEstimates = [];
      var total = msgs["messages"].length;

      for (i=0; i<total; i++) { //fetch all messages of this token

        var msgUrl = "https://www.googleapis.com/gmail/v1/users/me/messages/" + msgs["
            messages"][i]["id"] + "?access_token=" + authToken + "&format=metadata&
            fields=sizeEstimate,payload(mimeType,headers)";

        let response = await fetch(msgUrl).catch((error) => {
          APPEND_STATISTICS('Some␣error␣occured␣trying␣to␣fetch␣an␣e-mail␣header:␣' +
              error);
          return;
        });

        if (!response.ok) {
          HIDE_LOADER();
          let responseStatus = response.status;
          if (responseStatus == 401) {
            inboxProcessingWithNewAuthToken();
            return;
          }
          APPEND_STATISTICS('HTTP␣' + responseStatus + '␣error␣trying␣to␣fetch␣an␣e-
              mail␣header');
          return;
        }

        var msg = await response.json().catch((error) => {
          APPEND_STATISTICS('Some␣error␣occured␣trying␣to␣fetch␣an␣e-mail␣header:␣' +
              error);
          return;
        });;

        emailIds.push(msgs["messages"][i]["id"]);
        headers.push(msg["payload"]["headers"]);
        mimeTypes.push(msg["payload"]["mimeType"]);
        sizeEstimates.push(msg["sizeEstimate"]);

      }

      //a maximum of 100 mails were fetched successfully

      if (msgs.hasOwnProperty("nextPageToken")) {
        pendingPageToken = msgs["nextPageToken"];
      }
      else {
        pendingEmails = false;
      }
```

```
      for (i=0; i<total; i++) { //process this token's e-mail headers

        var thisMimeType = mimeTypes.pop();
        var thisSizeEstimate = sizeEstimates.pop();

        var emailId = emailIds.pop();
        var header = headers.pop();
        var fields = header.length;

        var from = '';
        var fromSet = false;

        var subject = '';
        var subjectSet = false;

        var date = '';
        var dateSet = false;

        var lastReceived = '?';
        var hops = 0;

        for (var j=0; j<fields; j++) {

          if (!fromSet && header[j]["name"] === "From") {
            from = header[j]["value"];
            fromSet = true;
          }

          if (!subjectSet && header[j]["name"] === "Subject") {
            subject = header[j]["value"];
            subjectSet = true;
          }

          if (!dateSet && header[j]["name"] === "Date") {
            date = header[j]["value"];
            dateSet = true;
          }

          if (header[j]["name"] === "Received") {
            hops++;
            if (hops == 2) {
              lastReceived = header[j]["value"];
            }
          }
        }

        if (subject === '') {
          subject = "(no␣subject)";
          subjectSet = true;
        }

        var dateAdded = '␣(added␣by␣';
        if (date.includes(dateAdded)) {
          date = date.substring(0, date.indexOf(dateAdded));
        }

        if (hops < 2 || lastReceived === '?') { //ignore message with no received fields
          continue;
        }
        else {
          var regex = / * by mx\.google\.com * with (.*) id /g;
          var protocol = regex.exec(lastReceived); //SMTP, SMTPS, ESMPT, ESMPTS, etc
          var secure = /s$/i; //case insensitive, ends with an 's'
          try {

            totalReceived++; //if an exception occurs processing this e-mail, totalReceived will not increament

            var link = 'https://mail.google.com/mail/u/0/#all/' + emailId;

            var domain;
            if (/^from .*? \((.*?) \[/g.test(lastReceived) == false) {
              //desired regex failed
              domain = /^from (.*?) \(/g.exec(lastReceived);
            }
            else {
              domain = /^from .*? \((.*?) \[/g.exec(lastReceived);
            }

            var dns_search = "https://dns.google.com/resolve?name=" + domain[1] + "&type=PTR";

            let response = await fetch(dns_search).catch((error) => {
              APPEND_STATISTICS('Some error occured trying to fetch DNS information from ' + dns_search + ': ' + error);
              return;
            });
            if (!response.ok) {
              let responseStatus = response.status;
              APPEND_STATISTICS('HTTP ' + responseStatus + ' error trying to fetch DNS information from ' + dns_search);
              return;
            }

            var dom = await response.json().catch((error) => {
              APPEND_STATISTICS('Some error occured trying to fetch DNS information from ' + dns_search + ': ' + error);
              return;
            });;

            var thisDomain = '';

            if (!dom.hasOwnProperty("Authority") || !dom["Authority"].length>0 || !dom["Authority"][0].hasOwnProperty("name")) {
              if (!dom.hasOwnProperty("Answer") || !dom["Answer"].length>0 || !dom["Answer"][0].hasOwnProperty("name")) { //has neither authority nor answer property
                //this can be a result of a dns status 2 f.i., where the server refused the query
                thisDomain = domain[1];

              }
```

```
  else { //doesn't have authority property but has answer property
    thisDomain = dom["Answer"][0]["name"];
  }
}
else { //has authority property
  thisDomain = dom["Authority"][0]["name"];
}

var domStatus = dom["Status"];
var protocolSecurity = '';

var domainId = -1; //will remain -1 if the domain was secure

if (!secure.test(protocol[1])) {
  totalUnencrypted++;

  //update unencrypted domains list
  var foundDomain = false;
  for (var d=0; d<unsecureDomains.length; d++) {
    if (unsecureDomains[d].domain === thisDomain) {
      foundDomain = true;
      (unsecureDomains[d].count)++;
      domainId = unsecureDomains[d].id;
      break;
    }
  }
  if (!foundDomain) {
    domainId = unsecureDomains.length;
    unsecureDomains.push({id: domainId, domain: thisDomain, count: 1});
  }
  pendingAppends.push({label: scanLabel, from: from, subject: subject, date: date, protocol: protocol[1], link: link, provider: thisDomain});
  protocolSecurity = ':UNSECURE';

}
else {
  protocolSecurity = ':SECURE';
}

const hashDomain = await digestMessage(thisDomain);

from = from.replace("<", "");
from = from.replace(">", "");
var fromTok = from.split("␣"); //this has length at least 1
var senderDomain = "?";
var fi;
for (fi=0; fi<fromTok.length; fi++) {
  if (fromTok[fi].includes("@")) {
    var mailTok = fromTok[fi].split("@");
    if (mailTok.length < 2) break; //make sure mailTok[1] exists
    senderDomain = mailTok[1];
    break;
  }
}

resultsAsObjects.push({ id: domainId,
          hash: hashDomain.toString(),
                        dnsStatus: 'dns␣status␣' + domStatus,
                        date: date,
                        security: protocol[1] + protocolSecurity,
                        hops: hops + '␣hops',
                        lastDomain: thisDomain,
                        mimeType: thisMimeType,
                        sizeEstimate: thisSizeEstimate,
                        senderDomain: senderDomain});

          } catch(error) { //something went wrong trying to process given header
            APPEND_ERROR_PARAGRAPH('Some␣error␣occured␣trying␣to␣process␣the␣header␣of␣
                e-mail␣\"' + link + "\":␣" + error, false);
            errorsCount++;
          }
        }

        processedTotal += total;

      }

        while (pendingAppends.length > 0) {
        HIDE_LOADER();
        //display unsecure e-mails' information on the screen
        var pendingAppend = pendingAppends.shift();
        APPEND_LINK(pendingAppend.label, pendingAppend.from, pendingAppend.subject,
            pendingAppend.date, pendingAppend.protocol, pendingAppend.link,
            pendingAppend.provider);
      }

      inboxProcessing(); //recursive function call until all messages are processed or
          authToken expires

    }

    inboxProcessing(); //first function call with this authToken

  });
}

async function beginProcessing() {
  APPEND_CENTER_PARAGRAPH("Your␣e-mail␣processing␣just␣began.␣Do␣not␣close␣this␣tab␣
      until␣processing␣is␣completed.");
  document.body.appendChild(document.createElement("br"));
  APPEND_LOADER();

  chrome.identity.getAuthToken({'interactive': true},
  async function (token) {
    authToken = token;

    //fetch profile info
    var url = "https://gmail.googleapis.com/gmail/v1/users/me/profile?access_token=" +
        authToken;

    let response = await fetch(url).catch((error) => {
      APPEND_ERROR_PARAGRAPH('Some␣error␣occured␣trying␣to␣fetch␣your␣profile␣
          information:␣' + error, true);
      return;
    });

    if (!response.ok) {
      let responseStatus = response.status;
      var error = 'HTTP␣' + responseStatus + '␣error␣trying␣to␣fetch␣your␣profile␣
          information';
      APPEND_ERROR_PARAGRAPH(error, true);
      return;
    }

    var profile = await response.json().catch((error) => {
      APPEND_ERROR_PARAGRAPH('Some␣error␣occured␣trying␣to␣fetch␣your␣profile␣
          information:␣' + error, true);
      return;
    });;

    clientHash = await digestMessage(profile["emailAddress"]);

    inboxProcessingWithNewAuthToken();
  });
}
```