I Spy With My Little Pixel

Hayes Choy
z5258816
W19A - COMP6841

April 13, 2022

Abstract

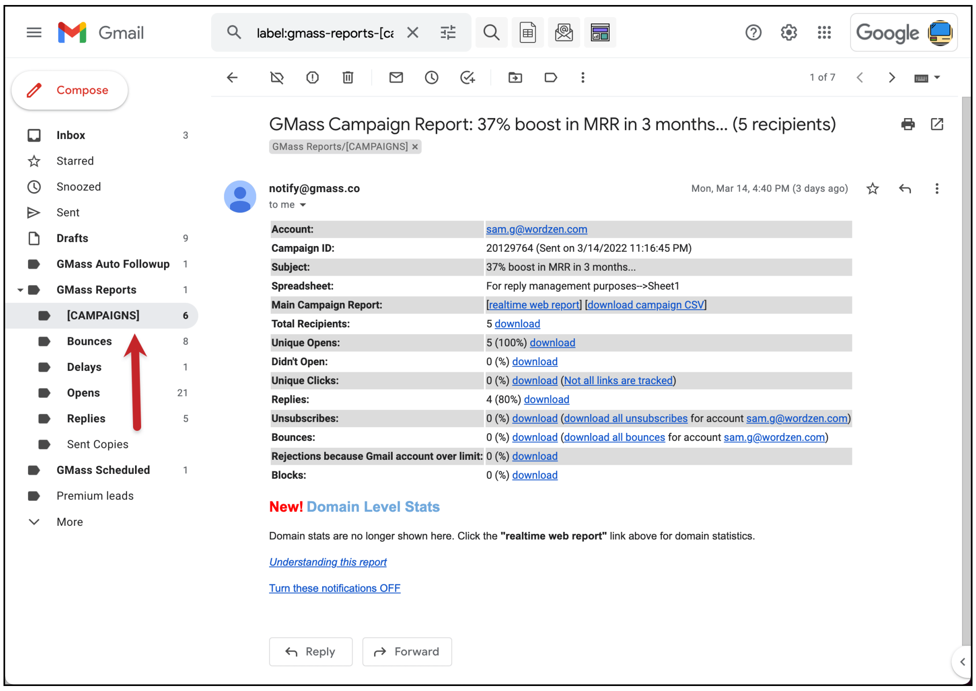
My project is a proof of concept on how spy pixels can be used to track when the recipient of an email opens an email, and the user's metadata. I have successfully created a backend to log all reads of a spy pixel, a database to store this data, and a frontend to create new spy pixels and view logs in a readable format. Furthermore, as a side project, I have learnt how HTML is used in emails, and investigated its differences with HTML in web development!

My spy pixel service is deployed at `https://spy-pixel.netlify.app` with the spy pixel itself being deployed at `https://spy-pixel.ts.r.appspot.com/spy-pixel.gif`

1 Context and motivation

1.1 Email tracking

If you've used a mass email service before, chances are the service also has a feature showing which recipients have read your email. Late last year and early this year, I was required to use GMass and was shocked when I could not only see the time they read my email, but also the number of unique clicks, replies and bounces.



GMass Campaign Report: 37% boost in MRR in 3 months... (5 recipients)

GMass Reports/[CAMPAIGNS] ×

notify@gmass.co
to me

Mon, Mar 14, 4:40 PM (3 days ago)

Account:	sam.g@wordzen.com
Campaign ID:	20129764 (Sent on 3/14/2022 11:16:45 PM)
Subject:	37% boost in MRR in 3 months...
Spreadsheet:	For reply management purposes-->Sheet1
Main Campaign Report:	[realtime web report] [download campaign CSV]
Total Recipients:	5 download
Unique Opens:	5 (100%) download
Didn't Open:	0 (%) download
Unique Clicks:	0 (%) download (Not all links are tracked)
Replies:	4 (80%) download
Unsubscribes:	0 (%) download (download all unsubscribes for account sam.g@wordzen.com)
Bounces:	0 (%) download (download all bounces for account sam.g@wordzen.com)
Rejections because Gmail account over limit:	0 (%) download
Blocks:	0 (%) download

New! Domain Level Stats

Domain stats are no longer shown here. Click the **"realtime web report"** link above for domain statistics.

Understanding this report

Turn these notifications OFF

Figure 1: GMass feature demo.

Like Slack or old text messages, I had always assumed emails were a safe place where read receipts would never be implemented, and I could safely "seen" emails and procrastinate a reply. Clearly, something was sent along with the email invisibly, and for my project I wanted to understand how they worked, and perhaps, what other information I might be sending.

After a few Google Searches, I learnt that this feature was made possible thanks to spy pixels.

1.2 What is a spy pixel?

A spy pixel is a 1x1 pixel transparent image that is sent alongside email content, with its source as an http request to a server. When the email is loaded on the recipient's side, the transparent pixel image is sent from the server, and the request is logged by the server. It looks something like this:

```
<img width="1" height="1" border="0" src="http://www.host.com/spy-pixel.png" />
```

The server is able to parse through the request headers and extract information about the user, such as their browser, device, OS, architecture and their location.

1.3 Where are they used?

Spy pixels are used **everywhere!**. On 10 March I opened up my inbox and inspected the elements of each email I received. Every newsletter seemed to contain a spy pixel embedded, to track whether I read it or not. Screenshots can be found in Appendix 6.1 as examples.

From a marketing perspective, this makes sense. Marketers would be able to adapt their approach to their subscribers based on the activity collected from an email. If an email was able to prompt the user to read the email, and then click on its links, and then later lead to a purchase, then the email was successful. As Richard Buckland says, "if data can be collected then it will be collected", and it seems that all digital newsletters now have this feature to better their interactions with customers.

1.4 How to stop spy pixels

The scary thing is that you *can't* stop spy pixels from being embedded into emails. Spy pixels take advantage of the way HTML `img` tags fetch data from a host if the source isn't local, and this feature isn't always bad. This feature is what allows emails to have variety with its graphics, and removing this feature from email clients would force every email to be a boring black and white text.

Another point of concern is that the use of spy pixels are a grey area legally since the user isn't aware of the data being collected. However, that has never stopped companies from collecting data.

The only way to really stop these spy pixels from happening is to download an extension that pre-parses your email before rendering, and removes the spy pixel from HTML before showing it to the user. Examples include PixelBlock or Tracker Blocker, but these are limited to web-based email clients and don't prevent spy pixels from appearing on your phone. They also tend to only work on popular email clients such as GMail or Outlook.

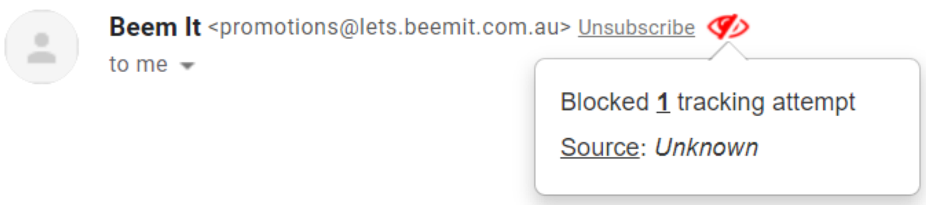

Beem It <promotions@lets.beemit.com.au> Unsubscribe
to me
Blocked 1 tracking attempt
Source: *Unknown*

Figure 2: PixelBlock blocking a spy pixel in an email from Beem It

While creating my own spy pixel service, I also discovered GMail's native approach to spy pixels. Each image rendered in an email is sent through GoogleImageProxy, which anonymises the user data. This prevented my spy pixel from detecting any of the user's metadata. So while a spy pixel is able to detect when a user has read an email, it is restricted to only do just that.

Other email clients such as Outlook unfortunately don't go through the effort of anonymising user metadata.

It's important to note that it's the process of loading an image that makes a user trackable. Spy pixels have become the industry standard since they are super small, meaning they minimally affect load times, and are transparent, meaning the user won't notice them. However, any image that loads off an external server can be used, including email signatures or header banners! These aren't detected by the above Chrome extensions, so overall, it's virtually impossible to prevent the collection of your metadata as you browse your emails.

1.5 What about clicks?

Clicks are tracked using URL forwarders, such as Bitly or other URL shorteners. When a request is sent to a server, it redirects the browser to the actual link and logs the redirect. This is a fairly simple idea, and a service that is readily available, so this feature is not explored in this project's scope.

2 Implementation of my spy pixel service

2.1 Backend

My backend was created using Flask in Python 3 and is deployed on a Google Cloud App Engine instance. The main code responsible is in Appendix 6.2.

The pixel is generated in the last three lines of the function. Since the image is of 1x1 pixel size, we can encode the image and store it as a hard-coded string. Thanks to the website https://png-pixel.com, it gave me all the strings I needed. It turns out a GIF saves a few bytes, which would increase load performance, which is why I have decided on a /spy-pixel.gif route.

```
gif_str =base64.b64decode('R0lGODlhAQABAIAAAP///////yH5BAEKAAEALAAAAAABAAEAAAICTAEAOw==')
```

When a request is called at the /spy-pixel.gif route, it first checks if the request came from the deployed service copying the spy pixel to the user's clipboard or is from when the user copy pastes it into their email. This is because both of these operations trigger a call to the backend, causing it to be logged. We don't want to log our own views, so we disregard either calls.

```
if request.headers.get("Referer") !="https://mail.google.com/" and request.headers.get("Referer") !=
                                              "https://spy-pixel.netlify.app/":
```

The neat thing about http requests is that you can also provide query parameters. The route checks for a campaign ID and an email to identify the user. Campaign IDs are compulsory so that it has a point of reference and can summarise data from multiple spy pixels. Email identifiers are optional, but allow the sender to differentiate between recipients if they wish. If there is a campaign ID present in the request, then we log the data to the database.

```
<img width="1" height="1" border="0"
    src="https://spy-pixel.ts.r.appspot.com/spy-pixel.gif?campaign=CAMPAIGNID&email=E@MAIL" />
```

2.2 Database

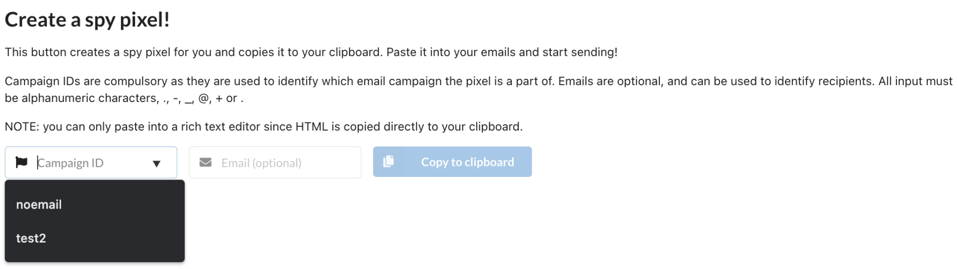
I used Firestore as my database. I chose Firestore since I've had experience using it before in my JavaScript projects and knew that the general functionality was easy to use. I also wanted to learn how to integrate it into Python projects, which I had never done before.

2.3 Frontend

The frontend (deployed at https://spy-pixel.netlify.app) has two types of pages: a dashboard, containing a spy pixel creator and ongoing campaigns, and a campaign details page.

2.3.1 Spy Pixel Creator

The spy pixel creator is the frontend interface to creating a spy pixel. A spy pixel can be easily created manually by writing some HTML and using a img tag, but I wanted to make this user friendly for not just coders! A spy pixel can take in two query parameters: the campaign ID and an optional email identifier. The copy to clipboard button is disabled until a valid campaign ID is entered. Also, since campaign IDs will be reused often, I added a datalist containing all the current campaign IDs so that the user can easily click on them if reusing a campaign ID.


Create a spy pixel!

This button creates a spy pixel for you and copies it to your clipboard. Paste it into your emails and start sending!

Campaign IDs are compulsory as they are used to identify which email campaign the pixel is a part of. Emails are optional, and can be used to identify recipients. All input must be alphanumeric characters, ., -, _, @, + or .

NOTE: you can only paste into a rich text editor since HTML is copied directly to your clipboard.

Figure 3: Spy pixel creator with pre-loaded datalist

Upon clicking the copy to clipboard button, the image is rendered invisibly and then copied into the clipboard. This can then be pasted into a rich text editor such as an email draft, which would sit invisibly in the body content! A side effect of this method is that it triggers a render of the spy pixel, and sends a request to my backend. This is why my backend requires an `if` statement to ignore any requests referred from the frontend site. Unfortunately, this was the only way to get it to work since other methods would copy the HTML as plaintext and not as HTML.

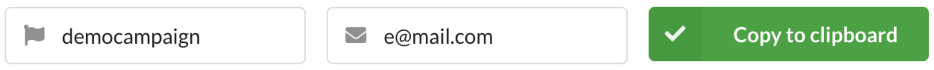


Figure 4: Responsive copy spy pixel to clipboard button

I also realised that since I was passing in a parameter straight into HTML, I was vulnerable to a XSS attack! If JavaScript was injected into the parameters, then the process of loading the image and copying it into the clipboard could trigger the script and run malicious code. To prevent this from happening, I disabled the copy to clipboard button if it detected characters that weren't on the whitelist, and visually showed which parameter was causing this error by highlighting the input box red.

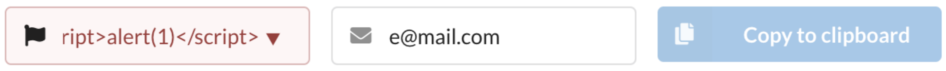


Figure 5: Input boxes turn red on invalid input

2.3.2 List of campaigns

Campaigns that have views will appear in the list of campaigns. Each campaign will appear as a card, and clicking on them will take you to its details. Campaigns also contain a short summary describing how many views it has.

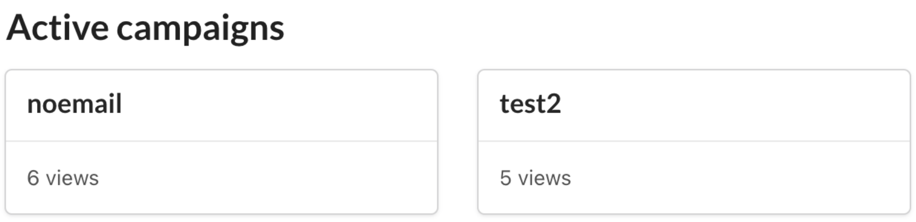
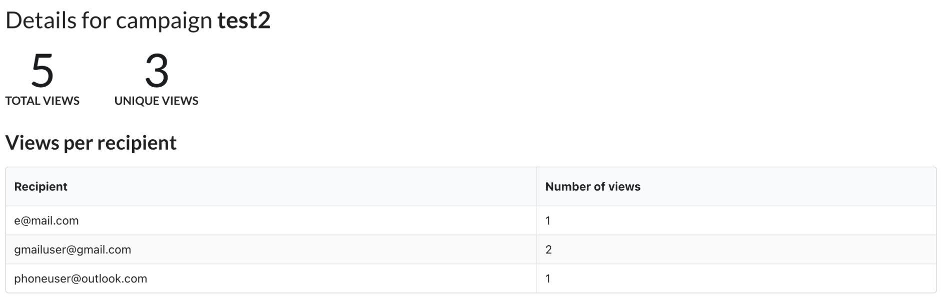


Figure 6: List of active campaigns

2.3.3 Campaign Details Page

This page summarises the data collected about a certain campaign based on a campaign ID. It shows the total views, unique views (per email identifier that is passed in as a query parameter), and also the views per recipient.


Details for campaign **test2**

5 TOTAL VIEWS

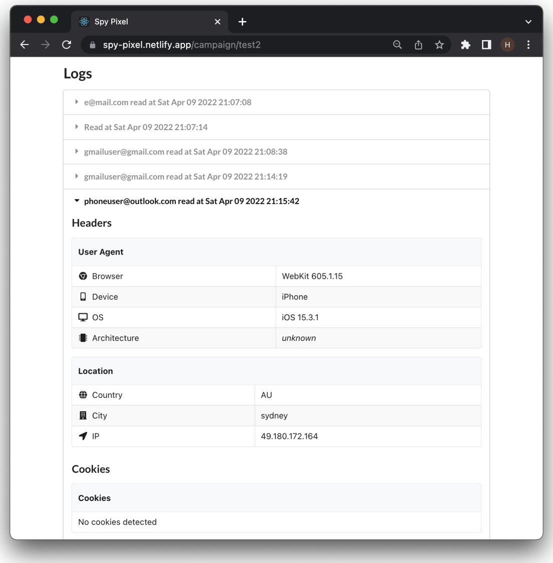
3 UNIQUE VIEWS

Views per recipient

Recipient	Number of views
e@mail.com	1
gmailuser@gmail.com	2
phoneuser@outlook.com	1

Figure 7: Summary of all metadata collected from spy pixels with campaign id `test2`

The logs are listed in chronological order, and can be expanded to show more information regarding the metadata collected when the spy pixel was loaded. Metadata includes the browser, device, OS and architecture extracted from the user agent, and also the country, city and IP address. Cookies are also scraped however I haven't encountered an email client that allows cookies to be scraped (this feature does work however if spy pixel is loaded in a new tab).



Logs

- e@mail.com read at Sat Apr 09 2022 21:07:08
- Read at Sat Apr 09 2022 21:07:14
- gmailuser@gmail.com read at Sat Apr 09 2022 21:08:38
- gmailuser@gmail.com read at Sat Apr 09 2022 21:14:19
- **phoneuser@outlook.com read at Sat Apr 09 2022 21:15:42**

Headers

User Agent	
Browser	WebKit 605.1.15
Device	iPhone
OS	iOS 15.3.1
Architecture	*unknown*

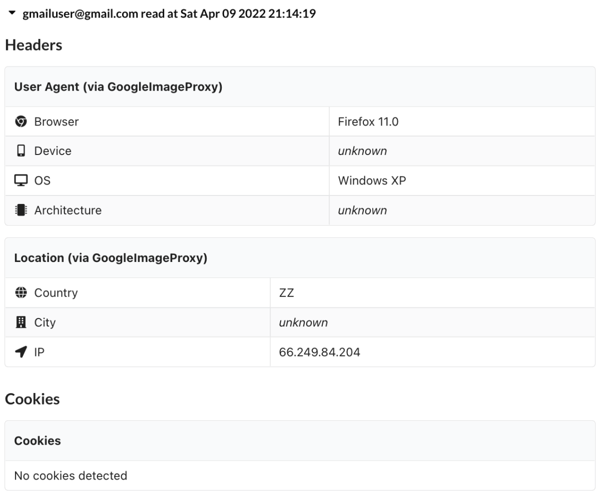
Location	
Country	AU
City	sydney
IP	49.180.172.164

Cookies

Cookies
No cookies detected

Figure 8: Metadata shown in an easy-to-read table

If the spy pixel was loaded through a proxy, the proxy name is shown and can help explain nonsensical metadata (such as a ZZ country!).



▾ gmailuser@gmail.com read at Sat Apr 09 2022 21:14:19

Headers

User Agent (via GoogleImageProxy)	
Browser	Firefox 11.0
Device	*unknown*
OS	Windows XP
Architecture	*unknown*

Location (via GoogleImageProxy)	
Country	ZZ
City	*unknown*
IP	66.249.84.204

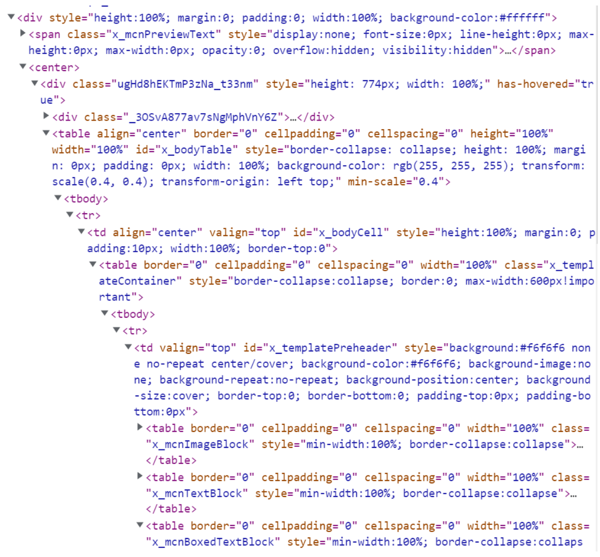
Cookies

Cookies
No cookies detected

Figure 9: Detection of GoogleImageProxy

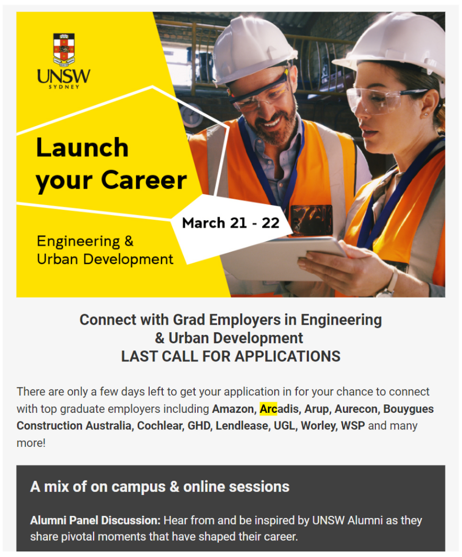
3 An aside - HTML in Emails

While inspecting the HTML in emails, you might notice two odd things: inline styling is used everywhere, and data table tags `table`, `tbody`, `td` and `tr` are used in place of `div`s and other HTML elements. Below is a UNSW newsletter email with its HTML shown next to it.



```
▾<div style="height:100%; margin:0; padding:0; width:100%; background-color:#ffffff">
  ▸<span class="x_mcnPreviewText" style="display:none; font-size:0px; line-height:0px; max-
  height:0px; max-width:0px; opacity:0; overflow:hidden; visibility:hidden">…</span>
  ▾<center>
    ▾<div class="ugHd8hEKTmP3zNa_t33nm" style="height: 774px; width: 100%;" has-hovered="tr
    ue">
      ▸<div class="_3OSvA877av7sNgMphVnY6Z">…</div>
      ▾<table align="center" border="0" cellpadding="0" cellspacing="0" height="100%"
      width="100%" id="x_bodyTable" style="border-collapse: collapse; height: 100%; margi
      n: 0px; padding: 0px; width: 100%; background-color: rgb(255, 255, 255); transform:
      scale(0.4, 0.4); transform-origin: left top;" min-scale="0.4">
        ▾<tbody>
          ▾<tr>
            ▾<td align="center" valign="top" id="x_bodyCell" style="height:100%; margin:0; p
            adding:10px; width:100%; border-top:0">
              ▾<table border="0" cellpadding="0" cellspacing="0" width="100%" class="x_templ
              ateContainer" style="border-collapse:collapse; border:0; max-width:600px!impo
              rtant">
                ▾<tbody>
                  ▾<tr>
                    ▾<td valign="top" id="x_templatePreheader" style="background:#f6f6f6 non
                    e no-repeat center/cover; background-color:#f6f6f6; background-image:no
                    ne; background-repeat:no-repeat; background-position:center; background
                    -size:cover; border-top:0; border-bottom:0; padding-top:0px; padding-bo
                    ttom:0px">
                      ▸<table border="0" cellpadding="0" cellspacing="0" width="100%" class=
                      "x_mcnImageBlock" style="min-width:100%; border-collapse:collapse">…
                      </table>
                      ▸<table border="0" cellpadding="0" cellspacing="0" width="100%" class=
                      "x_mcnTextBlock" style="min-width:100%; border-collapse:collapse">…
                      </table>
                      ▾<table border="0" cellpadding="0" cellspacing="0" width="100%" class=
                      "x_mcnBoxedTextBlock" style="min-width:100%; border-collapse:collaps
```

(a) Email's HTML with inline styling



Connect with Grad Employers in Engineering & Urban Development
LAST CALL FOR APPLICATIONS

There are only a few days left to get your application in for your chance to connect with top graduate employers including **Amazon, Arcadis, Arup, Aurecon, Bouygues Construction Australia, Cochlear, GHD, Lendlease, UGL, Worley, WSP** and many more!

A mix of on campus & online sessions

Alumni Panel Discussion: Hear from and be inspired by UNSW Alumni as they share pivotal moments that have shaped their career.

(b) Rendered email

Figure 10: HTML contains tables, yet the original email has no data tables visible

It turns out that emails have to use HTML like this because of the restrictions of email clients. Emails are rendered statically, not dynamically like a React website. This means it isn't able to respond to user input and change its content. Due to the multitude of screen sizes, browsers, devices and email clients there are available, there are over 15000 ways an email can be rendered (an accuracy check can be found in Appendix 6.4). In other words, there are 15000 ways to parse an HTML document, and so HTML must be written in a way that even the most basic email client can render an email exactly as the sender intended.

For example, some email clients strip away `<head>` when rendering the email, taking away with it the style sheets. Not only does this explain the overbearing presence of inline styling being, but also why emails don't tend to use non-browser friendly fonts that you would typically be able to load with a CSS style sheet.

Some email clients also don't render `div` blocks properly, and can only render `tables`. To ensure the layout is preserved across all email clients, senders must use `table`, `tr` and `td` to format columns and rows.

HTML is also rendered from top to bottom, just like HTML in the web. To ensure email's content is loaded quickly, the spy pixel should be placed at the bottom of the email.

4 Reflection

I'm really grateful to have the opportunity to learn what I did! It's not everyday that my university projects spark so much curiosity and freedom for me to explore and learn what I want to explore and learn.

As explained in section 1, I chose this topic because I noticed privacy concerns when using the mass email service GMass and I was able to see each time my recipients viewed my email. Initially I thought emails were simple forms of communication, much like a digital letter that only contained its content, but now that I've discovered the existence of spy pixels, it's clear to me that even emails have been "weaponised" as tools to track and collect data on users, whether they like it or not.

I consider myself very lucky to have chosen this topic to research on, because I am now quite wary of spy pixels and have taken strategies to prevent their effect. Whereever possible, I try and use Chrome on my laptop or PC to browse my emails since I have downloaded the Chrome extension PixelBlock which is most effective at preventing spy pixels. When I'm out, I only use GMail on my phone since it uses GoogleImageProxy to anonymise my data.

The learning experience has also changed how quickly I reply to my emails. Recently, I've been applying for internships, and chances are that if they use an automated email service to send me emails, they've probably seen that I've read the email already. To show companies I am an attentive person, I reply as quickly as I can!

I faced a multitude of challenges during the completion of my assignment. I work full time meaning I needed to balance my study sessions and university courses around my 9-5 work schedule. Another major external factor that implicated me was having to travel two hours each day to Gosford Hospital from weeks 6 to 8.

There were also a few technical challenges. Google Cloud App Engine was more overwhelming than I thought, and I had initially started on Google Cloud Functions which later turned out to be the wrong microservice to use. The process involved a lot of googling and debugging, and managed to get it working on Google Cloud App Engine on my second day of trying! If I had to do this again, I would just use Heroku since I had experience with it, but I was stubborn and wanted to save my deeply-sunk cost. At least now I have been exposed to GCP at a basic level.

Another problem I faced was React upgrading to version 18 on 29 March 2022. This was not backwards compatible with my libraries, and I went through a deep rabbit hole of reinstalling `webpack` and `npm`, before finding the root cause! Looking back, if I had just started my assignment earlier I wouldn't have ran into this problem, but another lesson that I learnt is to always inspect my `package.json` file and look carefully at my dependency versions.

I also tried to use `geoip-lite` to obtain a geolocation based on an IP address. While this seemed like a good idea, a few google searches revealed that IP addresses and physical addresses aren't as relatable as I thought. An IP address is the address of a computer in a network and not in real life, which explained why most IP addresses I tested with ended up showing the data centre in Sydney's CBD. This was a minor setback since I thought this feature would be super impressive, but had to quickly change scope once I realised I didn't have enough information to pull it off.

Overall I'm quite proud of my project! It is a fully working demo of how spy pixels work, and even has a frontend to be a service other people can use. I was also able to further my learning by finding differences between HTML in emails and HTML on the web, which revealed just how outdated emails and backwards compatible emails aim to be. If I had more time, perhaps I could have implemented a feature for users to upload their own image and have that image log views. This would make it obvious that it's not just transparent pixels we need to be worried about, but the actual process of loading any image in our emails.

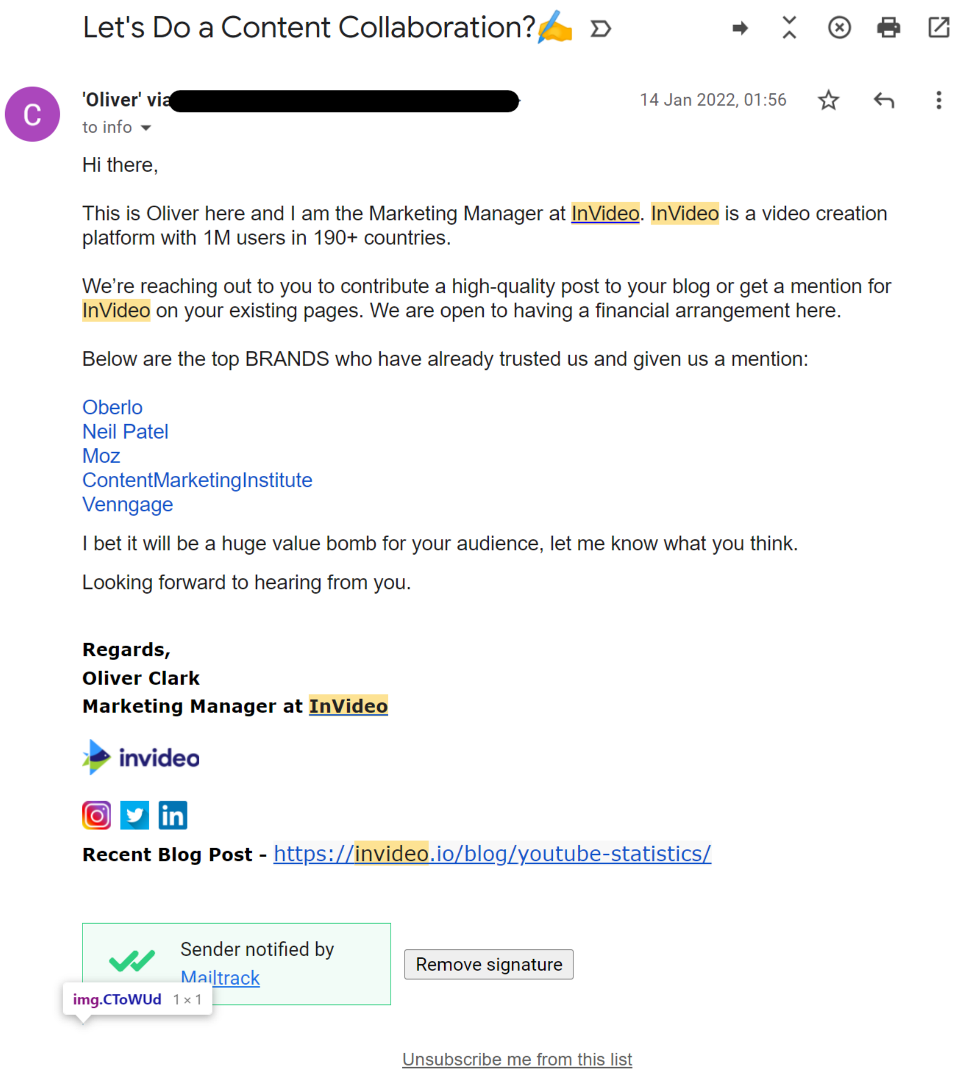
5 Conclusion

Spy pixels take advantage of the most fundamental rules of HTML image rendering to track email recipients. It's a tool that not only tracks email reads, but user metadata including their browser, device, OS, architecture, country, city, IP address, and cookies. This makes the collected data extremely valuable for marketing teams of companies who send frequent email newsletters, as they can see how effective their emails are and how changes in marketing strategies impact user interaction.

Unlike other data gathering tools, spy pixels secretly collect this information without notifying the user, since they are transparent and small. They can also come in the form of large visible images, since it is the process of loading an image off an external server that allows user's metadata to be collected. Out of all the email browsers that I've tested with, I believe GMail to be the best service to use to maintain your anonymity since it loads images through a proxy, anonymising the data in the request when the email is loaded. This doesn't prevent a request from being logged and notifying the sender that you've seen their email, but makes it so that the metadata sent cannot identify the recipient.

6 Appendix

6.1 Spy pixels in my inbox



Let's Do a Content Collaboration?✍️

'Oliver' via [redacted]

to info

14 Jan 2022, 01:56

Hi there,

This is Oliver here and I am the Marketing Manager at InVideo. InVideo is a video creation platform with 1M users in 190+ countries.

We're reaching out to you to contribute a high-quality post to your blog or get a mention for InVideo on your existing pages. We are open to having a financial arrangement here.

Below are the top BRANDS who have already trusted us and given us a mention:

Oberlo
Neil Patel
Moz
ContentMarketingInstitute
Venngage

I bet it will be a huge value bomb for your audience, let me know what you think.

Looking forward to hearing from you.

Regards,
Oliver Clark
Marketing Manager at InVideo

Recent Blog Post - https://invideo.io/blog/youtube-statistics/

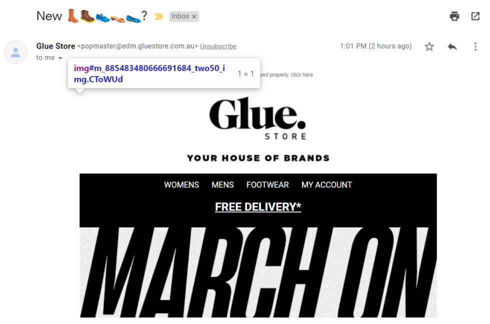
Sender notified by Mailtrack

img.CToWUd 1 × 1

Remove signature

Unsubscribe me from this list

Figure 11: Mailtrack embeds a spy pixel at the bottom. Luckily they are nice enough to tell me about it!



New 👢🥾👟👡🥿?

Inbox ×

Glue Store <popmaster@edm.gluestore.com.au> Unsubscribe

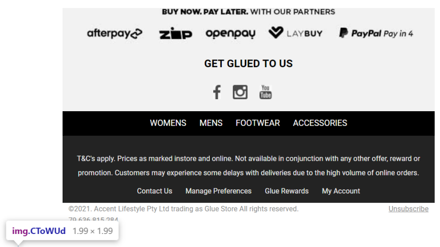
to me

1:01 PM (2 hours ago)

img#m_885483480666691684_two50_img.CToWUd 1 × 1

yed properly, click here.

(a) Glue embeds a spy pixel at the top styling



T&C's apply. Prices as marked instore and online. Not available in conjunction with any other offer, reward or promotion. Customers may experience some delays with deliveries due to the high volume of online orders.

Contact Us Manage Preferences Glue Rewards My Account

©2021. Accent Lifestyle Pty Ltd trading as Glue Store All rights reserved. Unsubscribe

img.CToWUd 1.99 × 1.99

(b) Glue embeds a second spy pixel at the bottom

Figure 12: Glue embeds two spy pixels in their email

6.2 Backend code

https://github.com/hayeselnut/spy-pixel/tree/main/backend

6.2.1 Spy pixel route

```
@app.route('/spy-pixel.gif')
def spy_pixel():
    # We don't want to log our own views
    if request.headers.get("Referer") !="https://mail.google.com/" and request.headers.get("Referer") !=
                                                            "https://spy-pixel.netlify.app/":
        # Campaign is loggable
        if request.args.get("campaign"):
            # Log to database
            log ={
                u'timestamp': datetime.datetime.now(),
                u'email': request.args.get("email"),
                u'headers': dict(request.headers),
                u'cookies': dict(request.cookies)
            }

            doc_ref =db.collection(u'campaigns').document(request.args.get("campaign"))
            doc =doc_ref.get()

            if not doc.exists:
                doc_ref.set({u'logs': []})

            doc_ref.update({u'logs': firestore.ArrayUnion([log])})

    gif_str =base64.b64decode('R0lGODlhAQABAIAAAP///////yH5BAEKAAEALAAAAAABAAEAAAICTAEAOw==')
    resp =make_response(send_file(io.BytesIO(gif_str), mimetype='image/gif'), 200)
    return resp
```

6.3 Frontend code

https://github.com/hayeselnut/spy-pixel/tree/main/frontend

6.4 Are there really 15000 ways to render an email?

6.4.1 Online email clients

There **110 common web browsers** (according to ua-parser-js): 2345Explorer, 360 Browser, Amaya, Android Browser, Arora, Avant, Avast, AVG, BIDUBrowser, Baidu, Basilisk, Blazer, Bolt, Brave, Bowser, Camino, Chimera, Chrome Headless, Chrome WebView, Chrome, Chromium, Comodo Dragon, Dillo, Dolphin, Doris, Edge, Electron, Epiphany, Facebook, Falkon, Fennec, Firebird, Firefox [Reality], Flock, Flow, GSA, GoBrowser, ICE Browser, IE, IEMobile, IceApe, IceCat, IceDragon, Iceweasel, Instagram, Iridium, Iron, Jasmine, K-Meleon, Kindle, Klar, Konqueror, LBBROWSER, Line, Links, Lunascape, Lynx, MIUI Browser, Maemo Browser, Maemo, Maxthon, MetaSr Midori, Minimo, Mobile Safari, Mosaic, Mozilla, NetFront, NetSurf, Netfront, Netscape, NokiaBrowser, Obigo, Oculus Browser, OmniWeb, Opera Coast, Opera [Mini/Mobi/Tablet], PaleMoon, PhantomJS, Phoenix, Polaris, Puffin, QQ, QQBrowser, QQBrowserLite, Quark, QupZilla, RockMelt, Safari, Sailfish Browser, Samsung Browser, SeaMonkey, Silk, Skyfire, Sleipnir, Slim, SlimBrowser, Swiftfox, Tesla, Tizen Browser, UCBrowser, UP.Browser, Vivaldi, Waterfox, WeChat, Weibo, Yandex, baidu, iCab, w3m, Whale Browser

There are at least **6 common online email clients**: GMail, Hotmail, Yahoo! Mail, ProtonMail, Outlook, iCloud Mail.

Let us assume there are **10 different screen sizes** browser windows can take. Realistically, this is almost an infinite number, so this is a conservative estimate.

Altogether, there are $\mathbf{110} \times \mathbf{6} \times \mathbf{10} = \mathbf{6600}$ different ways to render an email on an online email client.

6.4.2 Email apps on devices

There are **49 common vendors of devices** (according to ua-parser-js): Acer, Alcatel, Amazon, Apple, Archos, ASUS, AT&T, BenQ, BlackBerry, Dell, Essential, Fairphone, GeeksPhone, Google, HP, HTC, Huawei, Jolla, Lenovo, LG, Meizu, Microsoft, Motorola, Nexian, Nintendo, Nokia, Nvidia, OnePlus, OPPO, Ouya, Palm, Panasonic, Pebble, Polytron, Realme, RIM, Roku, Samsung, Sharp, Siemens, Sony[Ericsson], Sprint, Tesla, Vivo, Vodafone, Xbox, Xiaomi, Zebra, ZTE.

Let us assume that on average, each vendor listed above has at least **10 different device models** that can browse the internet. This is a conservative estimate since some vendors have many, many devices. For example, Apple has over 30 iPhone and iPad models altogether, and Samsung has over 50.

Let us assume that there are at least **20 email apps** that can be downloaded onto each device model. There are actually over 100 email apps in both App Store and Google Play, but we shall make a conservative estimation since some devices such as Blackberry and Nintendo don't have that many number of choices.

Altogether, there are $\mathbf{49} \times \mathbf{10} \times \mathbf{20} = \mathbf{9800}$ different ways emails can be natively rendered on a device.

6.4.3 Conclusion

Altogether, there are $\mathbf{6600} + \mathbf{9800} = \mathbf{16400}$ different ways to render an email, and this is using conservative estimates throughout.

6.5 Blog Posts

23 Feb 2022 Email Security

3 Mar 2022 Research

10 Mar 2022 Checking My Own Inbox

22 Mar 2022 HTML In Emails

31 Mar 2022 First Prototype

6 Apr 2022 Database Integration

7 Apr 2022 Frontend Deployed

6.6 Estimated hours of work

Activity	**Hours**	**Notes**
Deciding project	1	Blog post Email Security
Researching spy pixels	1	Blog post Research
Looking at my inbox	1	Blog post Checking My Own Inbox
Researching HTML in emails	1	Blog post HTML In Emails
First prototype	2	
Deploying prototype to Google Cloud App Engine	5	Blog post First Prototype
Firestore Database local integration	1	Blog post Database Integration
Firestore Database integration with deployed backend	4	
Configuring React	2	Unbeknownst to me, React 18 was released on 29 March 2022, which was not backwards compatible with my libraries. I went through a deep rabbit hole of reinstalling `webpack` and `npm`, before finding the root cause.
Frontend development	8	
Frontend deployment	2	Blogpost Frontend Deployed GitHub Pages didn't work, so after a bunch of had to learn and use Netlify
Debug and clean	3	Changing icons, fixing backend logging unwanted views where referer was definitely the user
Report Writing	6	
Total	**37**	

7 Further Reading

7.1 Spy pixels

- https://www.theverge.com/2019/7/3/20681508/tracking-pixel-email-spying-superhuman-web-beacon-open-tracking-read-receipts-location
- https://www.nutshell.com/blog/email-tracking-pixels-101-how-do-tracking-pixels-work
- https://www.wired.com/story/how-to-tell-which-emails-track-you/
- https://mashable.com/article/how-to-block-email-pixel-tracking
- http://engineering.curalate.com/2016/10/19/building-a-tracking-pixel.html
- https://limjiahe.com/2020/06/20/how-1x1-pixel-tracking-cookies-work-learn-by-implementing-your-own-tracking-system/
- https://www.youtube.com/watch?v=k4HImDV9S2Y

7.2 HTML in emails

- https://www.htmlemailcheck.com/check/
- https://www.smashingmagazine.com/2017/01/introduction-building-sending-html-email-for-web-developers/
- https://blog.hubspot.com/marketing/html-email
- https://mailbakery.com/blog/how-email-html-coding-differs-from-web-html-coding/
- https://www.campaignmonitor.com/dev-resources/guides/coding-html-emails/