4th Year Project Report
ImageAware+ - Phishing Detection & Educational Platform

Name	**Lorcan Kelly Zazzera**
Student ID	**C00288941**
Course	**Bachelor of Science (Honours) in Cybercrime and IT Security**
Supervisor	**Mark Cummins**

Abstract

ImageAware+ is a hybrid phishing detection and educational platform developed as a Final Year Project for the BSc (Hons) in Cybercrime and IT Security at South East Technological University, Carlow. The project addresses a recognised and growing gap in conventional phishing detection: most existing tools analyse email text, URLs, and metadata in plain form, but are entirely blind to phishing content delivered inside image files or rendered within HTML email bodies. A fake Geek Squad invoice, a PayPal billing alert, or a DocuSign impersonation email constructed as a graphic bypasses text-based filters completely while appearing entirely convincing to a human recipient.

The system combines multi-pass Optical Character Recognition (OCR) using Tesseract with OpenCV-based image preprocessing, QR code detection and decoding, HTML href extraction, and a 29-indicator rule-based scoring engine to produce explainable, forensically traceable risk assessments for submitted image files and .eml email files. Threat intelligence integration with VirusTotal, URLScan.io, and PhishTank provides external URL reputation data, while a comprehensive email header analysis module covers SPF, DKIM, DMARC authentication failures, display name spoofing, and reply-to domain mismatches.

Formal evaluation on 300 labelled samples, comprising 150 phishing emails from the Nazario 2025 corpus and 150 legitimate emails from the TREC 2007 ham corpus, demonstrated 80.95% precision and a 2.67% false positive rate on the email pipeline at the Medium detection threshold. Image pipeline evaluation on 22 samples achieved 100% precision with 0% false positive rate, a result that honestly surprised me given how variable OCR quality can be on real phishing graphics. The system is deployed as a publicly accessible platform at imageawareplus.onrender.com, integrating the forensic analysis tool with comprehensive phishing awareness educational content.

Acknowledgements

I would like to thank my project supervisor for their guidance and feedback throughout the development of this project. I would also like to acknowledge Jose Nazario for maintaining the publicly available phishing corpus used in the evaluation dataset, and the TREC organisers for the 2007 spam track public corpus. The open-source communities behind Tesseract OCR, OpenCV, Flask, and ReportLab provided the foundational tools without which this system could not have been built.

Table of Contents

1. Introduction

1.1 Background

Phishing has been around since the mid-1990s. It is still, arguably, the most effective attack method in widespread use, and that combination, old, well-understood, and still working, is what drew me to this topic. The scale alone is hard to genuinely internalise: the Anti-Phishing Working Group (APWG) estimates roughly 3.4 billion phishing emails are sent every single day, accounting for around 1.2% of all global email traffic [1]. The Verizon 2023 Data Breach Investigations Report found phishing contributing to 36% of all data breaches, making it the single most common initial attack vector across industries [2]. The FBI's Internet Crime Complaint Center put financial losses from phishing and business email compromise at over $4.57 billion in 2023 alone [3]. Those are not numbers that suggest a problem anywhere close to being solved.

Traditional email security tools, spam filters, secure email gateways, antivirus scanners, all essentially do the same thing: they read text. They check email bodies, scan sender domains against blacklists, test URLs against threat intelligence feeds, and apply heuristic rules built up over years of pattern recognition. These approaches are genuinely good at what they do. The problem is they only work on content they can actually see. And attackers figured that out.

The workaround is almost elegant in its simplicity. Render the entire phishing content as an image. A fake Geek Squad invoice, a PayPal billing alert, or a DocuSign signature request, constructed as a high-resolution PNG, looks completely convincing to a human reader. From the perspective of a text-based scanner, though, the email body is either empty or contains only a few lines of harmless text. The actual payload, the brand logo, the fake invoice number, the urgency language, the suspicious phone number, all of it is locked inside a graphic that the scanner simply cannot read. It scores zero and gets delivered.

The rise of QR codes as an attack vector, sometimes called quishing (QR phishing), adds a further dimension. By embedding a malicious URL within a QR code image, an attacker can redirect the victim to a phishing site through an out-of-band channel that most email security gateways do not inspect. The recipient scans the QR code with

their mobile phone, which takes them directly to the malicious page, completely bypassing any URL scanning the email gateway might apply.

1.2 Problem Statement

The core problem is simple to state: existing phishing detection tools are blind to content that lives inside images. A system that reads only the text of an email will give a Geek Squad invoice scam a score of zero, because from its perspective the email body is essentially empty. It misses the brand impersonation, the fake invoice number, the urgency language telling the recipient to call immediately, the toll-free number at the bottom, and the malicious URL embedded in the image as text, all of which would be obvious the moment you apply OCR and actually read what is in the graphic. This was the problem I wanted to solve.

There was also a second problem I did not fully anticipate until I started testing real phishing emails. When a phishing email arrives as a .eml file, the actual malicious URL is frequently not in the plain text body at all, it is buried inside an HTML href attribute. A parser that only reads the plain text part simply never encounters it. This is not an edge case. Looking through real DocuSign and PayPal impersonation emails from the Nazario corpus, it is a consistent pattern: the clickable link the victim is supposed to follow exists only in the HTML, and the plain text body contains nothing actionable. Extracting those href URLs became one of the more important parts of the implementation.

A third challenge is explainability. Security analysts and incident responders who investigate phishing reports need more than a binary verdict. They need to understand why a system classified something as phishing, which specific indicators were present, and what evidence supports each indicator. A black-box classifier that returns "phishing: 94% confidence" is of limited forensic value without the underlying evidence.

1.3 Research Objectives

The following research objectives were defined at the outset of the project:

- Develop an OCR-based pipeline capable of extracting readable text and URLs from phishing images, enabling content-based analysis of image-based phishing attacks.

- Build a comprehensive email analysis pipeline that handles .eml files, extracts URLs from both plain text and HTML parts, and analyses email headers for authentication failures and spoofing indicators.
- Integrate external threat intelligence APIs (VirusTotal, URLScan.io, PhishTank) with caching to enable URL reputation checking without excessive API quota consumption.
- Design and implement a fully explainable rule-based scoring engine with named indicators, evidence lists, and a structured breakdown that enables forensic documentation of phishing verdicts.
- Conduct a formal evaluation of the system against a labelled dataset to measure precision, recall, and false positive rate at multiple detection thresholds.
- Deploy the system as a publicly accessible platform with educational content to make phishing awareness accessible to a general audience.

1.4 Research Questions

The following research questions guided the development and evaluation of the system:

- Can OCR-based content analysis detect phishing indicators in image files that bypass text-based email security filters?
- How effective is a rule-based heuristic scoring engine at identifying phishing emails when analysing extracted body text and email headers, without reliance on machine learning?
- What are the primary failure modes of content-based phishing detection and how can they be characterised systematically?
- Can a hybrid image and email analysis pipeline achieve acceptable precision and false positive rates on a labelled evaluation dataset?

1.5 Project Scope

The system includes the following capabilities:

- Image file analysis (.png, .jpg, .jpeg, .bmp, .webp) using OCR, QR code detection, and URL extraction.
- Email file analysis (.eml) covering MIME parsing, plain text and HTML body extraction, href URL recovery, and header analysis.
- Threat intelligence enrichment via VirusTotal, URLScan.io, and PhishTank APIs with SQLite caching.
- A 29-indicator rule-based explainable scoring engine covering eight attack categories.
- Structured JSON and PDF forensic report generation.
- A publicly deployed web platform with analysis tool and educational content.

The system explicitly does not include: analysis of email attachments (PDF, Word, Excel), real-time email gateway integration, multilingual indicator detection beyond English, or machine learning classification.

1.6 Dissertation Structure

Chapter 2 reviews the academic and technical literature on phishing attacks, image-based phishing evasion techniques, email authentication mechanisms, and threat intelligence platforms. Chapter 3 describes the system design, architecture, and technology stack. Chapter 4 documents the implementation of each major component. Chapter 5 presents the formal evaluation methodology and results. Chapter 6 discusses the findings, strengths, and limitations. Chapter 7 identifies directions for future work. Chapter 8 presents conclusions. References and appendices follow.

2. Literature Review

2.1 Phishing Attacks

Phishing as an attack category dates to the mid-1990s, when attackers impersonating AOL staff sent instant messages requesting account credentials. The term "phishing" is attributed to a 1996 Usenet post and deliberately evokes the image of casting a wide net to catch unwitting victims. Over the following three decades, phishing evolved from crude mass-spam campaigns to highly targeted spear phishing, business email compromise, and multi-stage social engineering operations.

Modern phishing is characterised by psychological manipulation rather than technical exploitation. Attackers leverage authority (impersonating banks, government agencies, or technology companies), urgency (claiming that accounts will be suspended or charges applied unless immediate action is taken), and familiarity (using logos, branding, and language indistinguishable from legitimate communications) to deceive recipients into taking actions against their own interests. The FBI's IC3 identifies business email compromise as the costliest form of cybercrime, with losses exceeding $50 billion between 2013 and 2022 [3].

Academic research on phishing detection has traditionally focused on URL analysis, machine learning classifiers, and email header inspection. Fette et al. (2007) proposed PILFER, one of the earliest ML-based phishing email classifiers, using ten features derived from email content and headers. Subsequent work expanded feature sets and classification approaches, with ensemble methods and deep learning models achieving high accuracy on benchmark datasets. However, as detection systems improved, attackers adapted their techniques, leading to an arms race in which new evasion methods continuously emerge.

2.2 Image-Based Phishing

Image-based phishing represents a specific and well-documented evasion technique. By encoding phishing content as an image rather than text, attackers render text-based analysis ineffective. A 2006 study by Fumera et al. documented early image spam using randomly generated backgrounds and distorted text to evade OCR-based spam filters. While their context was spam rather than phishing specifically, the evasion principle is identical.

More recent documentation of image-based phishing includes reports from Proofpoint, Cofense, and Cisco Talos describing campaigns where the entirety of a fake invoice, credential harvesting page screenshot, or notification alert is rendered as a single embedded image. Verizon's 2023 DBIR notes that phishing emails increasingly avoid detectable text patterns, with the payload delivered visually or through embedded links pointing to external pages [2].

QR code phishing (quishing) emerged as a significant threat from 2022 onwards. Security researchers at Hoxhunt and Abnormal Security documented campaigns in 2023 where QR codes embedded in email images directed victims to credential harvesting pages, bypassing URL scanning entirely because the malicious URL was encoded within the QR image rather than present as a clickable link. The challenge of quishing is compounded by the fact that recipients typically scan QR codes with mobile devices, which may lack the security controls present on corporate workstations.

2.3 Email Security Mechanisms

Email authentication standards provide the primary technical defence against sender spoofing. Sender Policy Framework (SPF), defined in RFC 7208, allows domain owners to publish a list of authorised mail servers in DNS. When a receiving mail server checks SPF for an incoming message, it verifies that the sending server's IP address appears in the sender domain's SPF record. A failure indicates potential spoofing.

DomainKeys Identified Mail (DKIM), defined in RFC 6376, uses public-key cryptography to allow domain owners to sign outgoing messages. The receiving server verifies the signature against the public key published in DNS. A valid DKIM signature provides cryptographic assurance that the message was sent by an authorised party and was not modified in transit.

Domain-based Message Authentication, Reporting and Conformance (DMARC), defined in RFC 7489, builds on SPF and DKIM by specifying what action receivers should take on messages that fail authentication. DMARC also provides a reporting mechanism that allows domain owners to monitor authentication failures. Despite being available since 2012, DMARC adoption remains incomplete, and many

legitimate email senders do not implement it, meaning its absence alone is not a reliable phishing indicator.

In practice, the absence of SPF, DKIM, and DMARC records is common in both legitimate and phishing email, particularly for older messages and smaller organisations. This limits the reliability of authentication-based indicators when used in isolation and necessitates their combination with content-based signals in a multi-indicator scoring system.

2.4 Threat Intelligence Platforms

Threat intelligence platforms aggregate reputation data on URLs, IP addresses, and domains from multiple sources, providing a lookup service that security tools can query in real time. VirusTotal, acquired by Google in 2012, is the most widely used platform, aggregating results from over 70 antivirus engines and URL scanners. Its API allows programmatic submission of URLs for analysis and retrieval of historical scan results, making it practical for integration into automated detection pipelines.

URLScan.io is a specialised URL scanning service that performs live browsing of submitted URLs and captures screenshots, DOM content, network requests, and certificates. Unlike VirusTotal's primarily static analysis, URLScan captures dynamic behaviour, which is valuable for detecting phishing pages that only reveal their content after JavaScript execution. PhishTank is a community-driven database of confirmed phishing URLs, maintained by volunteers who manually verify submissions. Its lookup API allows querying whether a specific URL has been previously reported as phishing.

The practical limitation of all threat intelligence platforms for phishing detection is latency. New phishing campaigns use freshly registered domains with no existing reputation. A URL that was registered within the last 24 hours and used in a targeted phishing campaign will return clean results from VirusTotal because no scanner has yet analysed it. This limitation, commonly known as the "zero-day phishing" problem, means that threat intelligence is most useful as a corroborating signal rather than a primary detection mechanism.

2.5 Optical Character Recognition for Security

OCR's application to security problems has been documented primarily in the context of spam filtering and CAPTCHA breaking. Kanich et al. (2011) demonstrated that OCR-based spam detection could be evaded by distorting text in specific ways, leading to an arms race between OCR improvement and deliberate OCR evasion in spam content. This history is directly relevant to phishing image analysis: attackers who are aware of OCR-based scanning may deliberately stylise text in their phishing images to reduce OCR accuracy.

Tesseract OCR, originally developed at HP Labs and later maintained by Google, is the most widely used open-source OCR engine. Version 4 introduced a Long Short-Term Memory (LSTM) neural network model alongside the legacy pattern-matching engine, significantly improving accuracy on natural-language text. However, Tesseract's performance degrades significantly on images with coloured or textured backgrounds, very small or very large fonts, unusual typefaces, and low DPI. These are precisely the characteristics of many phishing images, which use marketing-style layouts with brand colours, logos, and decorative fonts.

Multi-pass OCR strategies, where multiple preprocessing variants and page segmentation modes are applied and the best result is selected by confidence scoring, are documented in the document analysis literature as a means of improving robustness. This approach was adopted in the implementation of ImageAware+, where nine OCR variants (three preprocessing methods times three PSM modes) are evaluated per image.

2.6 Limitations of Existing Detection Systems

Commercial email security products from vendors such as Proofpoint, Mimecast, and Microsoft Defender for Office 365 use multi-layer approaches combining signature-based URL blacklisting, sandboxed link detonation, ML classifiers, and user reporting. Despite their sophistication, these systems are not designed primarily to analyse embedded images, and their OCR capabilities where they exist are not publicly documented or evaluated.

Academic phishing detection datasets, including the ISCX-URL2016 and PhishTank-based collections, focus almost exclusively on URL features rather than image content. This reflects the research community's historical focus on URL-based

detection and the relative novelty of systematic image-based phishing as a studied phenomenon. The scarcity of labelled phishing image datasets is itself a barrier to research progress.

The key gap that motivated this project is the absence of a publicly available, open-source, deployable system that specifically targets image-based phishing content through OCR analysis, combined with email header analysis and threat intelligence, and provides explainable output suitable for forensic documentation. This is the space ImageAware+ occupies.

3. System Design

3.1 System Overview

ImageAware+ is designed as a multi-input forensic analysis system with a web-based interface. It accepts image files (PNG, JPG, JPEG, BMP, WEBP) and email files (.eml) and produces a structured risk assessment comprising a numerical score from 0 to 100, a risk classification (Low, Medium, or High), an itemised breakdown of every indicator that contributed to the score, and downloadable forensic artefacts including a PDF report, annotated image, and OCR text file.

The system is designed around three core principles: explainability (every score point is traceable to a named indicator with supporting evidence), dual-input capability (both image and email files are first-class inputs processed through a unified scoring engine), and modularity (each component is independently testable and replaceable without affecting others).

3.2 System Architecture

The system is organised into three layers: the extraction layer, the intelligence layer, and the scoring and reporting layer.

The extraction layer is responsible for converting raw file input into structured data that the intelligence and scoring layers can process. For image files, this involves loading the image with OpenCV, applying preprocessing transformations, running multi-pass OCR, detecting QR codes, and extracting URLs from OCR text. For email files, this involves MIME parsing to extract headers, body text from both plain and HTML parts, inline images, and href URLs from HTML parts.

The intelligence layer enriches the extracted URLs with external reputation data. Extracted URLs are first filtered through the email cleaner to remove known safe domains, tracking pixels, and CDN links. Remaining URLs are queried against the VT cache (which may return cached results or make live API calls), URLScan.io for behavioural reputation, and PhishTank for confirmed phishing database matches. Domain age lookups via WHOIS are performed for image mode analysis.

The scoring and reporting layer receives all extracted and enriched data and applies the 29-indicator engine to produce a final score. The score, breakdown, and all

evidence are serialised to a structured JSON report. The PDF generation module then renders a formatted forensic report from the JSON. The web interface presents the results to the user and provides download links for all artefacts.

3.3 Technology Stack

The technology choices were driven by availability, community support, and fitness for purpose.

- Python 3.11 - Core application language. Python's extensive library ecosystem for image processing, email parsing, and web development made it the natural choice.
- Flask 3.x - Lightweight web framework for the HTTP interface. Chosen over Django for its simplicity and because the project does not require Django's ORM or admin interface.
- Tesseract OCR 5.x - OCR engine. The most capable open-source OCR engine available. Accessed via the pytesseract Python wrapper.
- OpenCV 4.x (headless) - Image preprocessing and computer vision. The opencv-python-headless variant was used for server deployment to avoid GUI dependencies.
- pyzbar - QR code and barcode detection and decoding library.
- ReportLab - Python PDF generation library used for forensic report production.
- SQLite (via Python sqlite3) - Local caching store for VirusTotal results, preventing redundant API calls.
- python-whois - Domain age detection via WHOIS queries, used in image mode only.
- gunicorn - Production WSGI server for deployment.
- Docker - Containerisation for reproducible deployment on Render.com.
- VirusTotal, URLScan.io, PhishTank APIs - External threat intelligence providers.

3.4 System Workflow

When a file is submitted through the web interface, the following workflow is executed. For image files: the image is loaded and preprocessed using OpenCV; multi-pass OCR is run on three preprocessing variants using three PSM modes; the best OCR result is selected by confidence scoring; QR codes are detected and decoded; URLs are extracted from OCR text using the repair pipeline; URLs are filtered and queried against threat intelligence; and the scoring engine evaluates all collected data against 29 indicators to produce a final score and breakdown.

For email files: the .eml is parsed using Python's BytesParser with the modern policy.default parser; headers are extracted and normalised; HTML parts are processed first to extract href and action URLs; plain text body is extracted and combined with any HTML-derived URLs; extracted images are decoded; the email analysis module checks SPF/DKIM/DMARC, display name, and reply-to; body text is scored by the scoring engine; if embedded images exist, each is also analysed through the image pipeline and the higher of body score and image score is taken as the final result.

In both cases, the result is serialised to a JSON report, a PDF is generated, and the results are rendered in the web interface. For the production deployment on Render, the analysis runs in a background thread and the user sees a processing page that polls for completion every three seconds, avoiding HTTP timeout issues with long-running analysis.

4. Implementation

4.1 Image Preprocessing

Getting Tesseract to reliably read text out of phishing images turned out to be considerably harder than I initially expected. Phishing images are not clean scanned documents, they are marketing-style graphics with coloured backgrounds, brand fonts, drop shadows, and decorative layouts, all of which Tesseract struggles with out of the box. The preprocessing pipeline applies three transformations to every input image, producing three variants that feed into the multi-pass OCR strategy.

4.1.1 Image Loading and Colour Conversion

Images are loaded using OpenCV's imread function, which handles all common raster formats. The loaded image is immediately converted to grayscale using cv2.cvtColor with the COLOR_BGR2GRAY conversion code. Grayscale conversion reduces the three-channel BGR image to a single intensity channel, which is the format Tesseract expects for single-channel input and which removes colour noise that can interfere with character boundary detection.

4.1.2 Adaptive Threshold Binarisation

Adaptive thresholding applies a different threshold value to each local region of the image, computed from the mean or Gaussian-weighted sum of neighbouring pixel values. This makes it particularly effective on images with uneven illumination or background gradients, which are common in phishing images that use coloured or shaded backgrounds. The implementation uses cv2.adaptiveThreshold with the ADAPTIVE_THRESH_GAUSSIAN_C method and a block size of 11 pixels.

4.1.3 Otsu Threshold Binarisation

Otsu's method computes a global threshold value by minimising the intra-class variance between the foreground (text) and background pixel classes. Unlike adaptive thresholding, Otsu uses a single threshold across the entire image, making it most effective on images with a consistent background. The implementation uses cv2.threshold with the THRESH_BINARY + THRESH_OTSU combination. Where adaptive thresholding performs better on complex backgrounds, Otsu often performs better on high-contrast, clean images.

4.2 OCR Text Extraction

4.2.1 Multi-Pass OCR Strategy

Rather than running Tesseract once and hoping for the best, the system runs it nine times per image: three preprocessing variants combined with three PSM modes (PSM 4 for a single column of variable-sized text, PSM 6 for a uniform text block, PSM 11 for sparse text scattered across the image). Nine runs, nine candidate results. The winner is selected by combining mean confidence score with word count, candidates with no text are filtered out first, then sorted by confidence with word count as a tiebreaker. In practice this approach made a substantial difference. A Geek Squad invoice that returned mostly noise under grayscale PSM 6 would come back as cleanly extracted text under adaptive threshold PSM 4. Running all nine and picking the best was the right call.

4.2.2 Confidence Filtering

Tesseract assigns each word a confidence score from 0 to 100 via the image_to_data output mode. Words below the threshold, set at 35 by default, are discarded. This step exists because Tesseract will confidently return complete nonsense when pointed at a coloured background or a logo; without filtering, the OCR output is littered with garbled character sequences that the scoring engine would never meaningfully interpret. The threshold of 35 was arrived at through a fairly tedious process of running the same images at different values and eyeballing the results. Lower than 35 and too much noise crept in. Higher and genuinely readable low-contrast text started getting dropped. 35 was the sweet spot, at least for the sample set I was working with.

4.2.3 Line Structure Reconstruction

Rather than joining all OCR words with spaces into a single string, the implementation reconstructs line structure by preserving Tesseract's block_num and line_num values from the image_to_data output. Words within the same block and line are joined with spaces; a newline is inserted between lines; a blank line is inserted between blocks. This is particularly important for invoice-style phishing images with two-column layouts, where a single-string OCR output would collapse both columns into an unreadable run of words.

4.3 URL Extraction and Normalisation

4.3.1 Robust URL Pattern Detection

URL extraction from OCR text must handle both well-formed URLs (beginning with http:// or https://) and bare domains (beginning with www.). The extract_urls_robust function applies a two-pattern regex: the first matches full URLs with scheme, the second matches www.-prefixed bare domains. Both are then passed through a cleaning pipeline that strips trailing punctuation characters commonly attached by OCR (periods, commas, parentheses, quotes).

4.3.2 OCR URL Repair

A dedicated url_repair.py module handles common OCR corruptions of URLs. Missing dots in domain names are recovered by detecting space-separated tokens that match domain-like patterns. Common OCR character substitutions (1 for l, 0 for o) are normalised. Missing http:// schemes are added for domain patterns that lack them. Concatenated URLs, where two URLs are joined without separation, are split at the occurrence of a second http:// or www. pattern within a single token.

4.3.3 Email URL Extraction from HTML

For email files, a critical additional extraction step recovers URLs from HTML href and action attributes. This is implemented in email_parser.py by processing all HTML MIME parts with Python's email.policy.default parser (which correctly handles quoted-printable and base64 encoding) and applying a regex to the decoded HTML content. The extraction occurs regardless of whether a plain text part also exists, and extracted URLs are appended to the body text so they reach the threat intelligence pipeline. HTML entities (e.g., &) are decoded before URL extraction. Redirect chain URLs containing multiple http:// occurrences are split to extract only the outer (attacker-controlled) domain.

4.4 QR Code Detection

QR code detection is implemented using the pyzbar library, which provides a Python wrapper around the ZBar barcode detection library. The detect_qr_codes function in qr.py loads the image with OpenCV, converts it to grayscale, and passes it to pyzbar's decode function. Detected QR codes return their data content (typically a URL), bounding box coordinates, and barcode type.

If a QR code is detected and its data contains a URL, the URL is added to the list of URLs for threat intelligence processing, contributing the qr_data indicator. The bounding box coordinates are used to draw a highlighted rectangle on an annotated version of the image, which is included in the forensic report as visual evidence of the QR code's presence and location. This annotation makes QR code evidence immediately visible in the PDF report without requiring the analyst to re-examine the original image.

4.5 Threat Intelligence Integration

4.5.1 VirusTotal Integration

VirusTotal is queried via its public API v3. Extracted URLs are submitted to the /urls endpoint, which returns analysis results from over 70 security vendors. The VTCache module (vt_cache.py) wraps the API call with a SQLite-backed cache: each URL is stored by its SHA-256 hash, and cache hits return the stored result without consuming API quota. The returned data includes the number of malicious and suspicious vendor detections, which are mapped to the vt_malicious and vt_suspicious indicators in the scoring engine. The free tier allows 500 requests per day, which necessitates the URL filtering step that removes known safe domains before any API queries are made.

4.5.2 URLScan Integration

URLScan.io is queried using its search API, which returns existing scan results for a URL without triggering a new live scan. This approach avoids the latency and privacy implications of live scanning (which would involve URLScan browsing the submitted URL) while still providing historical reputation data. The returned data is stored in the threat intelligence results for presentation in the report.

4.5.3 PhishTank Integration

PhishTank is queried against its confirmed phishing URL database. A match in PhishTank is strong evidence of phishing, as each entry has been manually verified by community volunteers. During development, PhishTank closed new registrations, which prevented obtaining an API key for the production deployment. The integration is present in the codebase and functions correctly with a valid key, but is disabled with a placeholder in the production deployment.

4.5.4 Email URL Cleaning

Before any URLs are sent to threat intelligence APIs, the email_cleaner.py module filters out known safe and non-relevant domains. A curated blocklist of email service provider domains, CDN domains, tracking pixel patterns, and one-click unsubscribe URL patterns is maintained. This prevents the VT quota from being consumed by legitimate infrastructure (e.g., apple.com, linkedin.com, mailchimp.com) and reduces false positive contributions from marketing email URLs that would otherwise inflate scores.

4.6 Threat Intelligence Caching System

The caching system was a critical design requirement given VirusTotal's 500 request per day free tier limit. The VTCache class maintains a SQLite database (vt_cache.sqlite) in the outputs directory. Each cache entry stores the URL, its SHA-256 hash as the primary key, the full API response JSON, the verdict (malicious/suspicious/clean), individual signal counts, and a timestamp.

The get_or_query(url) method first checks whether the URL hash exists in the cache. On a cache hit, it returns a CachedVTResult object populated from the stored data without making any API call. On a cache miss, it calls the VirusTotal API, stores the result in the database, and returns the result. The caching approach means that once a URL has been analysed, any subsequent analysis involving the same URL costs zero API quota. For batch evaluation or repeated analysis of similar phishing campaigns (which often reuse the same malicious domains), this provides substantial quota savings.

4.7 Email Analysis Engine

The email analysis engine (email_analysis.py) performs header-based analysis independent of the body content scoring. It receives the parsed email headers and body text as inputs and returns a structured result containing a score and list of indicators.

4.7.1 Email Parsing

Email files are parsed using Python's BytesParser with policy.default, which correctly handles all MIME structure variants including multipart/mixed, multipart/alternative,

multipart/related, and nested multipart structures. Each MIME part is examined: text/plain parts contribute to the body text, text/html parts contribute both to body rendering and href extraction, and image/* parts are decoded and saved as embedded images for the image pipeline. The part's Content-Disposition header is checked to skip attachment parts that are not inline body content.

4.7.2 Header Analysis

Headers are normalised to lowercase with hyphens replaced by underscores for consistent lookup. The analysis checks the following: Reply-To domain differing from the sender domain (12 points, indicating potential misdirection of replies); sender domain using a high-risk TLD from a monitored list including .xyz, .top, .click, .tk, .ml, .ga (10 points); brand name in the display name but not in the sender domain (display name spoof, 12 points with surname suppression to avoid false positives from personal names matching brand terms).

4.7.3 Email Authentication

SPF is checked by inspecting the Received-SPF and Authentication-Results headers. A fail or neutral result adds 8 points. An absent SPF record adds 8 points. DKIM is checked against DKIM-Signature and Authentication-Results headers; a failure or absence adds 8 points. DMARC is checked against Authentication-Results; a failure or absence adds 8 points. The email authentication score is capped at 20 points to prevent email security signals from dominating the total score.

4.7.4 Lookalike Domain Detection

The detect_lookalike_domain function extracts the base domain (removing subdomains and TLD) and computes its similarity to each monitored brand domain using Python's SequenceMatcher. Additionally, character substitution normalisation (0 for o, 1 for l, 3 for e, @ for a) is applied before comparison to catch homograph attacks. A similarity score above 0.80, or a containment relationship between the brand root and the sender domain root, triggers the lookalike indicator. The legitimate brand domain is excluded from matching to prevent false positives from authentic communications.

4.8 Explainable Risk Scoring Engine

The scoring engine (scoring.py) is the core analytical component of ImageAware+. It receives OCR text, keyword hits, URLs, QR data, threat intelligence results, email analysis results, and OCR confidence metrics as inputs, and returns a ScoredResult containing a final score, risk level, list of reason strings, and a detailed breakdown dictionary mapping each indicator name to its contribution, label, and evidence list.

4.8.1 Indicator Design

Each indicator is implemented as a self-contained scoring block that evaluates one specific signal. The _add_factor() helper function is called at the end of each block to register the indicator's contribution in the breakdown dictionary. The helper stores the indicator name, human-readable label, number of matching items, per-item weight, total contribution, and the list of matching evidence strings. This structure means the full evidence for every indicator is available for report generation.

4.8.2 Risk Indicator Categories

The 29 indicators are organised across eight categories. Threat Intelligence indicators (vt_malicious, vt_suspicious) contribute up to 30 points based on VirusTotal detection counts. URL indicators (url_present, multi_url, qr_found, qr_data, credential_url) contribute up to 31 points based on URL presence, count, QR code detection, and credential-harvesting URL path patterns. Domain Intelligence indicators (domain_impersonation, young_domains, display_name_spoof) contribute up to 40 points for suspicious domain patterns, newly registered domains, and display name spoofing. Content Layout indicators (invoice_layout, invoice_table) contribute up to 18 points for invoice-style vocabulary and table structures. Social Engineering indicators (urgency_indicators, legal_threat, sextortion, delivery_scam, job_scam, bec) contribute up to varying maximums for vocabulary matching specific attack types. Brand and Identity (brand_impersonation, email_security) contribute for brand name detection and authentication failures. Technical indicators (phone_numbers, hidden_link, ocr_mean_conf, keyword_hits) cover phone number detection, hidden hyperlink patterns, OCR confidence anomalies, and general phishing keyword density.

4.8.3 Email Mode vs Image Mode

The scoring engine operates in two modes depending on whether the input is from an email body or an image. Email mode applies different calibration to certain indicators. WHOIS domain age lookups are disabled in email mode due to performance

considerations. URL scoring applies a cap of six effective URLs to prevent legitimate newsletters with many links from scoring disproportionately highly on URL volume signals. The email_analysis result, containing header-based scores, is passed in as a separate parameter and merged with the content-based score. In image mode, all 29 indicators are active and WHOIS lookups are performed.

4.8.4 Overlap Caps

Without overlap management, indicators that share vocabulary would cause score inflation when multiple related signals fire simultaneously. Two overlap caps are implemented. The OVERLAP_BANKING_CAP limits the combined contribution of financial_lure, support_scam, and bank_cluster to 20 points, regardless of how many individual terms match. These three categories share significant vocabulary and detecting the same underlying financial theme through three indicators should not triple-count the same signal. The OVERLAP_SOCIAL_ENG_CAP limits the combined contribution of sextortion, delivery_scam, bec, and job_scam to 25 points. If the combined raw total exceeds the cap, each indicator's contribution is scaled proportionally to sum to the cap.

4.8.5 Risk Classification

The final numerical score is classified into three risk levels. A score below 35 is classified as Low, indicating that the content does not exhibit sufficient phishing indicators to warrant further investigation. A score between 35 and 69 inclusive is classified as Medium, indicating significant phishing indicators that warrant caution and potentially further investigation. A score of 70 or above is classified as High, indicating strong multi-indicator evidence of phishing. These thresholds were established empirically through live testing and evaluation, with the Medium threshold at 35 providing the best balance of precision and recall for the evaluation dataset.

4.9 Report Generation

The system produces three output artefacts for each analysis. First, a structured JSON report is written to the outputs directory, containing all metadata, OCR results, threat intelligence data, email analysis results, and the complete scored breakdown. This JSON report is the canonical data source from which all other outputs are derived. Second, the pdf_report_rl.py module reads the JSON report and generates a

formatted PDF using ReportLab, including executive summary, email metadata (for .eml inputs), OCR text, extracted URLs, threat intelligence results, and the full indicator breakdown table with evidence. Third, for image inputs, an annotated PNG is saved with QR code detection boxes, extracted URL annotations, and risk level overlaid on the image.

The PDF generation required careful layout management using ReportLab's Platypus framework. Key challenges included managing content overflow across pages, preventing indicator tables from splitting across page boundaries using KeepTogether elements, and maintaining consistent visual hierarchy across the varying amount of content produced by different analysis types.

4.10 Web Interface and Deployment

The Flask web application provides the user interface for file submission and results presentation. The interface was redesigned from a single-page upload form into a multi-page educational platform with four sections: Home (phishing awareness and statistics), Learn (educational content on seven attack types), About (system architecture and evaluation results), and Analyse (the upload and results interface).

The upload form accepts drag-and-drop in addition to file browser selection, with immediate client-side feedback showing the selected filename. Form submission triggers an asynchronous analysis workflow: the Flask route handler saves the uploaded file, creates a job ID, starts the analysis in a background thread using Python's threading.Thread, and immediately returns a processing page. The processing page uses JavaScript to poll a /status/<job_id> endpoint every three seconds; when the job completes, the user is automatically redirected to the /result/<job_id> page showing the full results.

Deployment uses Docker on Render.com. The Dockerfile uses python:3.11-slim as the base image and installs tesseract-ocr, tesseract-ocr-eng, libzbar0, libgl1, and libglib2.0-0 as system packages before installing Python dependencies. This containerised approach ensures that the Tesseract system dependency is always present and correctly configured in the production environment. The production server runs gunicorn with one worker and a 120-second timeout. Automatic deployment from

the GitHub main branch is configured, meaning a git push triggers a Render redeploy within minutes.

5. Experimental Evaluation

5.1 Dataset Selection

Selecting appropriate datasets for evaluation required careful consideration. The primary requirement was that the phishing dataset contain real phishing emails in their original .eml format, as synthesised or simplified samples would not exercise the full email parsing pipeline including MIME structure handling, embedded image extraction, and href URL recovery.

The Nazario 2025 Phishing Corpus was selected as the phishing dataset. Jose Nazario has maintained a publicly available collection of real phishing emails since the mid-2000s, with regular updates from contemporary phishing campaigns. The 2025 collection was obtained from monkey.org/~jose/phishing/ and converted from .txt archive format to individual .eml files using a custom conversion script. 150 samples were selected randomly from the available corpus. This corpus is cited in multiple academic phishing detection papers and represents a well-established research resource.

Choosing the benign dataset caused me more trouble than I expected. My first instinct was the SpamAssassin 2002 ham corpus, it is widely used in the academic literature and straightforward to obtain. I ran the evaluation, looked at the false positive rate, and it was terrible. Every single one of the 150 benign emails was triggering the authentication absence indicators. It took me an embarrassingly long time to realise why: emails from 2002 predate SPF (RFC published 2003), DKIM (RFC published 2004), and DMARC (RFC published 2012) entirely. Of course they all lack authentication records. The corpus was the wrong tool for evaluating a system that uses authentication signals, and the high false positive rate was a dataset artefact rather than a genuine failure of the scoring engine. I switched to the TREC 2007 ham corpus, extracted from the publicly available Kaggle dataset. 2007 is early enough that SPF was beginning to appear in legitimate email, making it a considerably more realistic baseline.

5.2 Test Environment

The evaluation was conducted on a Windows 11 workstation with a Python 3.12 virtual environment. The evaluation script (evaluation.py) was designed to be deterministic

and reproducible: VirusTotal, URLScan, and PhishTank API calls were disabled entirely during evaluation, WHOIS lookups were disabled in email mode, and the evaluation ran purely on content-based indicators. This design ensures that the evaluation measures the intrinsic capability of the scoring engine rather than the availability and response quality of external APIs.

The evaluation computed metrics at three detection thresholds simultaneously: High (score >= 70), Medium (score >= 35), and Low+ (score >= 25). For each sample, three binary predictions were computed alongside the continuous score and risk level. At the end of the run, precision, recall, F1, accuracy, and false positive rate were computed for each threshold using the standard definitions.

5.3 Evaluation Methodology

The evaluation followed the standard binary classification evaluation methodology. True Positives (TP) are phishing samples correctly classified as phishing. False Positives (FP) are benign samples incorrectly classified as phishing. True Negatives (TN) are benign samples correctly classified as benign. False Negatives (FN) are phishing samples incorrectly classified as benign. Precision is defined as TP / (TP + FP), measuring what fraction of positive predictions are correct. Recall is defined as TP / (TP + FN), measuring what fraction of actual phishing is detected. F1 is the harmonic mean of precision and recall. False Positive Rate (FPR) is FP / (FP + TN), measuring what fraction of legitimate email is incorrectly flagged.

For a forensic tool, precision and FPR are the most critical metrics. A high false positive rate erodes analyst trust and creates alert fatigue. A system that incorrectly flags a significant proportion of legitimate email cannot be used in practice regardless of its detection capability. Recall against the evaluation corpus, while important, is secondary to these concerns.

5.4 Detection Performance

The email pipeline evaluation results on 300 samples (150 phishing, 150 benign) are presented in the table below.

Threshold	TP	FP	TN	FN	Precision	Recall
High (70)	0	0	150	150	0.00%	0.00%

Medium (35)	17	4	146	133	80.95%	11.33%
Low+ (25)	43	43	107	107	50.00%	28.67%

At the High threshold, no emails were detected. Every phishing email in the corpus scored below 70, confirming that the High threshold is not viable for email analysis where many phishing emails score in the 35-60 range. At the Medium threshold, the system achieved 80.95% precision with a 2.67% false positive rate. Of the 17 true positives, all were invoice-style or brand-impersonation phishing with rich text content. At the Low+ threshold, recall improved to 28.67% but the false positive rate increased to 28.67%, making this threshold unsuitable for practical use.

The 11.33% recall figure is the one number in these results that looks genuinely bad, and it requires some honest explanation. Working through the 133 false negatives, four distinct patterns emerged. The largest group, 53 emails, 35% of false negatives, scored 10 to 19: minimal content with not enough recognisable vocabulary for indicators to fire at all. Another 46 emails (31%) scored 20 to 29, meaning some indicators triggered but the total did not reach the Medium threshold. 23 emails (15%) scored 0 to 9, most of which had either empty bodies or payloads entirely in PDF or Word attachments that the system does not analyse. The remaining 10 were near-misses, scoring 30 to 34. The honest interpretation is that the low recall is mostly a scope limitation rather than a flaw in the scoring logic itself. The system was built to analyse email body content and embedded images. A substantial portion of the Nazario corpus delivers its payload in attachments or uses techniques that leave essentially nothing in the body text to score.

The image pipeline evaluation on 22 samples (12 phishing, 10 benign) produced significantly stronger results.

Threshold	TP	FP	TN	FN	Precision	FPR
High (70)	0	0	10	12	0.00%	0.00%
Medium (35)	7	0	10	5	100%	0.00%

Honestly, the image pipeline results were better than I expected. 100% precision with 0% false positive rate at the Medium threshold, every image the system flagged was a genuine phishing image, and nothing benign was incorrectly marked. Given how

variable OCR quality can be on real-world phishing graphics, I was not confident those numbers were achievable. Recall came in at 58.33%, which reflects the fundamental OCR dependency. The invoice-style phishing images with clean, readable text scored between 41 and 68 out of 100 and were all correctly identified. The five missed images were a different story: a Twitch compliance scam, a Spotify recruitment scam, two images with genuinely unclear content, and a renewal scam where OCR returned almost nothing usable. Those misses are explainable, but they are still misses.

5.5 False Positive Analysis

The four false positives are worth looking at individually because they each tell you something specific about where the scoring engine needs adjustment. All four were TREC 2007 ham emails, and they shared two common characteristics: absence of modern authentication records (expected for 2007), and unusually high URL counts ranging from 12 to 65 links. Three of them were marketing newsletters or mailing list digests. The fourth was the one that genuinely surprised me, a completely legitimate personal email from someone named Scott Norton, whose surname triggered the Norton Antivirus brand impersonation indicator. The system thought it was detecting a tech support scam impersonating Norton. It was not. It was just a person’s name.

Fixing the surname issue required adding suppression logic to the display name spoof detection: if the brand keyword only appears as the last word of a multi-word display name, it is treated as a surname rather than impersonation. For the newsletter false positives, the fix was capping the effective URL count at six in email mode, so a newsletter with 50 article links does not score equivalently to a targeted phishing email containing three carefully placed malicious URLs. Both fixes were straightforward once the root cause was identified.

After these fixes, the false positive count at Medium threshold dropped from the pre-fix total to four stable false positives, all of which remain explainable as dataset characteristics (2007 emails without modern authentication) rather than genuine system errors.

5.6 Live Testing Results

In addition to the formal evaluation, a set of real-world phishing samples were tested through the web interface to verify system behaviour on contemporary attacks. These tests were conducted during development and informed several indicator calibrations.

A Geek Squad invoice scam image scored 68/100 Medium, correctly identifying invoice layout (+10), brand impersonation (+4), urgency indicators (+8), phone numbers (+6), support scam vocabulary (+8), and financial lure (+12). A fake Chase bank security alert in .eml format scored 46/100 Medium, triggering urgency indicators, credential phishing vocabulary, financial lure, and email security failures. A CAVRA DMCA legal threat .eml scored 61/100 Medium after the legal_threat indicator was added following its initial miss. A DocuSign impersonation .eml scored in the Medium range after the href extraction fix recovered the phishing URL that was previously invisible to the parser.

Confirmed benign emails tested included an Apple shipping confirmation (24/100 Low), a LinkedIn notification (16/100 Low), and a SETU university email (Low). These results confirm that the system does not over-flag legitimate communications from major brands when those brands are also monitored for impersonation.

6. Discussion

6.1 System Strengths

The thing I am most satisfied with in this project is the explainability. Every single point in the final score traces back to a named indicator with a list of the specific terms or patterns that triggered it. An analyst looking at the output can see exactly why the system called something phishing, verify that each indicator is legitimate, and document it properly for an incident report. That is actually more useful in a forensic context than a higher-accuracy black-box classifier that just hands you a number and no explanation. I made a deliberate choice early on to prioritise that property over raw classification performance, and I think it was the right call.

The dual-input capability is a genuine practical advantage. Most phishing detection research and tooling focuses exclusively on either image analysis or email header analysis. ImageAware+ integrates both in a unified pipeline that scores email body text first, then additionally analyses embedded images, taking the higher of the two scores. This means that phishing delivered through either channel is covered, and the system benefits from both signals when both are present.

The modular architecture facilitates maintenance, testing, and extension. Each module has a clear input/output contract and can be tested independently. Adding a new attack category requires only adding terms to a term list and a scoring block in scoring.py; it does not require retraining a model or restructuring the pipeline.

The educational platform component adds value beyond the forensic tool. The Learn page covering seven attack categories with real-world examples, red flags, and practical guidance represents a meaningful contribution to public phishing awareness. Making this content accessible through a free, publicly hosted platform aligns with the principle that human awareness is the most effective first-line defence against social engineering.

6.2 Limitations

The most significant limitation is the OCR dependency, and it is not one that can be engineered around within the current architecture. When phishing content is a graphic with no legible text, or when image quality is just too poor for Tesseract to get anything

useful out of it, the scoring engine has nothing to work with. Full stop. This is a genuine constraint of the approach, not something that better threshold tuning will fix. Visual brand recognition using image classification would be the appropriate technical solution, but this requires a trained model with sufficient phishing image training data, which was beyond the scope of this project.

The inability to analyse email attachments is the main reason recall is as low as it is against the Nazario 2025 corpus. Around 30% of the false negatives were emails where the entire phishing payload lived in a PDF or Word document that the system never opens. Detecting malicious attachments requires either static signature analysis, dynamic sandbox execution, or document content extraction, all of which involve significant additional system complexity.

The rule-based indicator weights were calibrated empirically through live testing rather than through systematic data-driven optimisation. While the weights produce reasonable results, they may not be optimal for all phishing categories or for phishing not represented in the live test samples. A logistic regression or gradient boosting model trained on the indicator features would likely produce better-calibrated weights for a specific evaluation corpus.

API rate limits constrain the threat intelligence capability. With 500 VirusTotal requests per day on the free tier, high-volume analysis is not practical. The caching system mitigates this for repeated URLs but does not resolve the fundamental quota constraint. Zero-day phishing domains not yet indexed by VirusTotal receive clean scores, limiting the threat intelligence signals for the most dangerous, freshest attacks.

6.3 Comparison with Existing Approaches

Compared to commercial email security gateways (Proofpoint, Mimecast, Microsoft Defender for Office 365), ImageAware+ is a research tool rather than a production security product. Commercial systems process email in real time at the gateway level, integrate with organisational identity systems, and have access to threat intelligence from millions of email flows. ImageAware+ is a manual submission tool for forensic analysis of suspected phishing samples, not an automated protection layer.

However, in the specific context it was designed for (forensic analysis of suspected phishing with explainable output), ImageAware+ provides capabilities that commercial

systems do not expose. Commercial systems do not typically provide per-indicator score breakdowns with evidence lists that an analyst can use in an incident report. The explainability gap in commercial tools is a recognised problem in security operations, and systems that address it have practical value.

Compared to academic phishing detection systems in the literature, ImageAware+ is distinguished by its focus on image-based phishing specifically, its integration of OCR with email analysis in a single pipeline, its deployment as a publicly accessible system rather than a research prototype, and its educational platform component. Most academic systems are evaluated only on URL features and are not designed for or capable of image content analysis.

7. Future Work

Several directions for future development were identified during the project that were not implemented due to time constraints or scope limitations.

Visual Brand Recognition

The most obvious gap to fill is visual brand recognition. A CNN trained on brand logos would address the limitation that matters most: phishing images that contain recognisable brand graphics but essentially no readable text for OCR to extract. A ResNet or EfficientNet model fine-tuned on a dataset of brand logos could detect PayPal, Apple, or Microsoft branding in an image regardless of whether OCR can extract any text. This would particularly improve detection of login page screenshot phishing, where the visual brand identity is the primary deception mechanism.

Email Attachment Analysis

Integration with a document content extraction library such as Apache Tika would enable analysis of PDF and Word attachment content. Many phishing emails deliver their payload entirely in a PDF invoice or a Word document with embedded malicious macros. Extracting text from these documents and passing it through the existing scoring engine would address approximately 30% of the current false negative cases.

Empirical Scoring Calibration

With a larger labelled dataset (1000+ samples), logistic regression or gradient boosting could be used to learn optimal indicator weights from data rather than setting them by hand. This would likely improve precision and recall on categories where the current weights are suboptimal, and would make the calibration process reproducible and justifiable.

DKIM Signature Verification

The current email analysis checks for the presence or absence of DKIM-Signature headers but does not actually verify the cryptographic signature. Implementing true DKIM verification using the dnspython library to retrieve the public key from DNS and Python's cryptography library to verify the signature would significantly strengthen the authentication analysis. A forged DKIM header that fails cryptographic verification is strong evidence of spoofing.

Multilingual Detection

Expanding the indicator vocabulary to cover phishing in Irish, Spanish, Portuguese, French, and German would address the multilingual limitation. Many phishing campaigns targeting European users are conducted in local languages, and English-only keyword lists miss these entirely. Language detection could be applied first and the appropriate vocabulary set selected based on the detected language.

Real-Time Gateway Integration

An SMTP proxy integration that could analyse incoming email in real time before delivery would transform the system from a forensic tool into an active protection layer. This would require addressing performance requirements (analysis must complete in under a few seconds for real-time use) and false positive management (incorrectly blocking legitimate email has severe consequences).

8. Conclusion

8.1 Summary of Contributions

This project set out to build something that could actually read what is inside a phishing image, and the end result is a working system with a measurable evaluation behind it. The specific contributions are as follows. First, an open-source, publicly deployed image phishing analysis pipeline combining multi-pass OCR, OpenCV preprocessing, QR code detection, and URL extraction that specifically targets the image-based phishing evasion technique. Second, a dual-input email and image analysis framework that unifies both analysis types under a single scoring engine and always takes the higher of the two scores when both signals are available. Third, a 29-indicator explainable rule-based scoring engine with overlap caps, email-mode calibration, and per-indicator evidence lists suitable for forensic documentation. Fourth, a formal evaluation on 300 labelled samples establishing baseline performance metrics for the system. Fifth, a publicly accessible educational platform combining the forensic analysis tool with comprehensive phishing awareness content.

8.2 Research Findings

The research questions posed in section 1.4 can now be answered based on the evaluation results and development experience. OCR-based analysis can detect phishing content in images when that content includes readable text, as demonstrated by the 100% precision and 58.33% recall of the image pipeline. However, it is fundamentally limited by images where phishing content is stylised, graphical, or too low-quality for reliable OCR extraction.

A rule-based heuristic scoring engine with 29 indicators across eight attack categories achieves 80.95% precision and 2.67% false positive rate on the email pipeline at the Medium threshold against the TREC 2007/Nazario 2025 evaluation dataset. This precision level is sufficient for practical use as a forensic analysis tool, particularly given that every positive verdict is fully explainable.

The primary failure modes of content-based phishing detection are: attachment-based delivery (payload in unreadable attachments), image-only emails (no extractable text), sparse legitimate-looking content (BEC style), and near-threshold misses (scores of

30-34). These patterns are consistent with the published literature on phishing evasion techniques and provide a clear roadmap for targeted improvements.

8.3 Practical Implications

The system is immediately useful in two practical contexts. First, as a forensic analysis tool for security analysts investigating reported phishing attempts. The explainable breakdown output, structured JSON report, and PDF artefacts provide the evidence documentation needed for incident reporting, employee communication, and potential legal proceedings. Second, as an educational resource through the public platform, where the Learn pages covering seven attack categories provide accessible, practical phishing awareness content for general users.

The deployment of the system as a Docker container on Render.com, with automatic deployment from GitHub, demonstrates that a fully functional security analysis tool can be developed, formally evaluated, and deployed to production within a Final Year Project timeline. The live platform at imageawareplus.onrender.com and the public repository at github.com/Lorcan02/ImageAwarePlus are tangible deliverables that demonstrate the project's outcomes to any interested party.

The project demonstrates that explainability and detection capability are not necessarily in conflict. The 29-indicator rule-based engine achieves competitive precision and false positive rates on the evaluation dataset while providing complete transparency about every verdict. For security tools deployed in contexts where human oversight and documentation are required, this combination of explainability and performance represents a significant practical advantage over opaque ML classifiers.

9. References

[1] Anti-Phishing Working Group (APWG) (2025). Phishing Activity Trends Report, Q1 2025. Available at: https://apwg.org [Accessed: April 2026].

[2] Verizon (2023). 2023 Data Breach Investigations Report (DBIR). Available at: https://www.verizon.com/business/resources/reports/dbir [Accessed: April 2026].

[3] FBI Internet Crime Complaint Center (IC3) (2023). 2023 Internet Crime Report. Available at: https://www.ic3.gov [Accessed: April 2026].

[4] Fette, I., Sadeh, N. and Tomasic, A. (2007). Learning to detect phishing emails. Proceedings of the 16th International World Wide Web Conference (WWW 2007). ACM, pp.649-656.

[5] Fumera, G., Pillai, I. and Roli, F. (2006). Spam filtering based on the analysis of text information embedded into images. Journal of Machine Learning Research, 7, pp.2699-2720.

[6] Nazario, J. (2025). Phishing Corpus. Available at: https://monkey.org/~jose/phishing/ [Accessed: March 2026].

[7] TREC (2007). TREC 2007 Spam Track Public Corpus. Available at: https://trec.nist.gov [Accessed: March 2026].

[8] Smith, R., et al. (2007). An Overview of the Tesseract OCR Engine. Proceedings of the 9th International Conference on Document Analysis and Recognition (ICDAR 2007). IEEE, pp.629-633.

[9] Bradski, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools, 25(11), pp.120-125.

[10] Kittler, J. (1986). Minimum error thresholding. Pattern Recognition, 19(1), pp.41-47. [Otsu thresholding method].

[11] VirusTotal (2024). VirusTotal Public API v3 Documentation. Available at: https://developers.virustotal.com [Accessed: April 2026].

[12] URLScan.io (2024). URLScan.io API Documentation. Available at: https://urlscan.io/docs/api/ [Accessed: April 2026].

[13] Leontiadis, N., Moore, T. and Christin, N. (2011). Measuring and analyzing search-redirection attacks in the illicit online prescription drug trade. Proceedings of USENIX Security 2011.

[14] ReportLab Group (2024). ReportLab PDF Library User Guide. Available at: https://www.reportlab.com/docs/ [Accessed: April 2026].

[15] Python Software Foundation (2024). email.policy module documentation. Available at: https://docs.python.org/3/library/email.policy.html [Accessed: April 2026].

[16] RFC 7208 (2014). Sender Policy Framework (SPF) for Authorizing Use of Domains in Email. IETF.

[17] RFC 6376 (2011). DomainKeys Identified Mail (DKIM) Signatures. IETF.

[18] RFC 7489 (2015). Domain-based Message Authentication, Reporting, and Conformance (DMARC). IETF.

[19] Hoxhunt (2023). QR Code Phishing Report 2023. Available at: https://hoxhunt.com [Accessed: April 2026].

[20] Render (2024). Render Web Services Documentation. Available at: https://render.com/docs [Accessed: April 2026].

Appendix A: System Architecture Reference

The following table summarises the primary modules, their locations, and their responsibilities.

Module	Location	Responsibility
report.py	modules/	Pipeline orchestration for image analysis
scoring.py	modules/	29-indicator rule-based scoring engine
email_parser.py	modules/	MIME .eml parsing, href extraction
email_analysis.py	modules/	Header analysis, SPF/DKIM/DMARC, spoofing
email_cleaner.py	modules/	URL filtering before threat intelligence
ocr_enhanced.py	modules/	Multi-pass OCR with confidence selection
ocr.py	modules/	Single-pass OCR utility
url_repair.py	modules/	URL reconstruction from OCR text
vt_cache.py	modules/	SQLite-backed VirusTotal result cache
urlscan.py	modules/	URLScan.io API integration
phishtank.py	modules/	PhishTank API integration
pdf_report_rl.py	modules/	ReportLab PDF forensic report generation
qr.py	modules/	QR code detection and URL extraction
app.py	web/	Flask routes, async job handling
Dockerfile	/	Docker build configuration
render.yaml	/	Render.com deployment configuration

Appendix B: Evaluation Results Tables

Email Pipeline Evaluation (n=300)

Threshold	TP	FP	TN	FN	Precision	Recall	F1	FPR
High (70)	0	0	150	150	0.00%	0.00%	0.000	0.00%
Medium (35)	17	4	146	133	80.95%	11.33%	0.199	2.67%
Low+ (25)	43	43	107	107	50.00%	28.67%	0.364	28.67%

Image Pipeline Evaluation (n=22)

Threshold	TP	FP	TN	FN	Precision	Recall	F1	FPR
High (70)	0	0	10	12	0.00%	0.00%	0.000	0.00%
Medium (35)	7	0	10	5	100%	58.33%	0.737	0.00%

False Negative Score Distribution (Email, n=150 phishing)

Score Range	Count	Percentage	Interpretation
0-9	23	15%	Empty body / attachment-based / image-only payload
10-19	53	35%	Minimal content, insufficient vocabulary
20-29	46	31%	Some indicators present, below threshold
30-34	10	7%	Near-misses
35+ (detected)	17	11%	Correctly detected at Medium threshold

Appendix C: 29-Indicator Scoring Engine Reference

Indicator Name	Category	Max Pts	Description
vt_malicious	Threat Intelligence	20	VirusTotal malicious engine count
vt_suspicious	Threat Intelligence	10	VirusTotal suspicious engine count
url_present	URL Signals	2	URLs detected in content
multi_url	URL Signals	4	Multiple URLs detected
qr_found	URL Signals	4	QR code present in image
qr_data	URL Signals	6	QR code contains URL
credential_url	URL Signals	15	Credential harvesting URL path patterns
domain_impersonation	Domain Intelligence	20	Brand name in suspicious sender domain
young_domains	Domain Intelligence	8	Newly registered domain (WHOIS, image mode only)
display_name_spoof	Domain Intelligence	12	Brand in display name, not sender domain
invoice_layout	Content Layout	10	Invoice structure vocabulary cluster
invoice_table	Content Layout	8	Structured table pattern detected
urgency_indicators	Social Engineering	12	Urgency and pressure vocabulary
legal_threat	Social Engineering	20	Legal threat / DMCA scam vocabulary
sextortion	Social Engineering	15	Sextortion / extortion vocabulary
delivery_scam	Social Engineering	16	Delivery / parcel scam vocabulary
job_scam	Social Engineering	12	Job scam / recruitment fraud vocabulary
bec	Social Engineering	15	Business Email Compromise vocabulary
brand_impersonation	Brand & Identity	4	Brand name in OCR text
email_security	Email Security	20	SPF/DKIM/DMARC failures and header anomalies
keyword_hits	Keywords	15	Phishing keyword density score
financial_lure	Financial	12	Financial and banking lure vocabulary

credential_phish	Financial	16	Credential harvesting vocabulary
support_scam	Financial	12	Tech support scam vocabulary
bank_cluster	Financial	12	Banking keyword cluster (threshold: 2 hits)
phone_numbers	Technical	6	Phone numbers detected in content
hidden_link	Technical	4	Hidden hyperlink pattern detected
ocr_mean_conf	Technical	4	Low OCR confidence indicating obfuscation
header_correlation	Email Security	0	Reserved (disabled in current version)

Glossary

The following terms are used throughout this dissertation.

- BEC (Business Email Compromise): A phishing attack category in which an attacker impersonates an executive, finance department, or trusted vendor to authorise fraudulent wire transfers or change payment details.
- DKIM (DomainKeys Identified Mail): An email authentication method using public-key cryptography to verify that a message was sent by an authorised server and was not modified in transit.
- DMARC (Domain-based Message Authentication, Reporting and Conformance): An email authentication policy that builds on SPF and DKIM, specifying what action receivers should take on authentication failures.
- EML: The standard file extension for email message files stored in MIME format.
- False Positive: A legitimate item incorrectly classified as malicious. In the context of this system, a benign email or image classified as phishing.
- False Negative: A malicious item incorrectly classified as benign. A phishing email or image not detected by the system.
- F1 Score: The harmonic mean of precision and recall, providing a single metric that balances both. Ranges from 0 (worst) to 1 (best).
- FPR (False Positive Rate): The proportion of benign items incorrectly classified as malicious. Calculated as FP / (FP + TN).
- Href: The HTML attribute specifying the target URL of an anchor element. In phishing emails, the malicious URL often appears only in href attributes rather than as visible text.
- MIME (Multipurpose Internet Mail Extensions): The standard for formatting non-ASCII email content including HTML bodies, attachments, and embedded images. MIME structures can be nested and multipart.
- OCR (Optical Character Recognition): The process of extracting machine-readable text from raster images. Used in this system to read phishing content from image files.
- Phishing: A social engineering attack in which an attacker impersonates a trusted entity to deceive the victim into revealing credentials, making payments, or taking other harmful actions.
- Precision: The proportion of positive predictions that are correct. Calculated as TP / (TP + FP).
- PSM (Page Segmentation Mode): A Tesseract OCR configuration parameter specifying how the input image's layout should be interpreted (e.g., as a single block of text, a sparse collection of text, or a single column).
- QR Code: A two-dimensional barcode that can encode URLs and other data. Used in quishing attacks to embed malicious URLs in email images.
- Quishing: Phishing conducted via QR codes embedded in email images, directing victims to malicious URLs through an out-of-band channel.

- Recall: The proportion of actual positive cases correctly identified. Calculated as TP / (TP + FN).
- SPF (Sender Policy Framework): An email authentication method allowing domain owners to specify which mail servers are authorised to send email on their behalf.
- WHOIS: A query and response protocol for querying databases containing registration information about domain names, including registration date.
- VirusTotal: A free online service that aggregates multiple antivirus scanner and URL analysis results, accessible via API for programmatic URL reputation queries.