POLITECNICO DI TORINO

Master Degree course in Cybersecurity

Master Degree Thesis

Email Spoofing and Domain Impersonation: A Technical and Experimental Study of DMARC Enforcement and Risk Assessment Frameworks

Supervisors
Prof. Andrea Atzeni
Prof. Paolo Dal Checco

Candidate
Nicola Seidita

Academic Year 2025-2026

Acknowledgements

I am thankful for the support and guidance provided by my supervisors Prof. Andrea Atzeni and Prof. Paolo Dal Checco, whose precise advice has been invaluable to this work. I would like to continue the rest of my acknowledgements in Italian, in order to make people better understand what I want to exprime.

Desidero ringraziare sinceramente i miei genitori, senza i quali non sarei qui. Durante tutti questi anni hanno sempre messo da parte i loro impegni e hanno fatto tanti sacrifici per potermi permettere di arrivare fino a qui. Senza il loro supporto e la loro fiducia questo traguardo non sarebbe stato possibile.

Un ringraziamento speciale va a mio fratello, che è sempre stato il mio primo sostenitore e non ha mai smesso di credere in me e nelle mie capacità. Con la sua presenza e il suo incoraggiamento è sempre stato per me un punto di riferimento e una fonte di motivazione nei momenti più difficili e in quelli più importanti.

Inoltre volevo ringraziare tutti i miei amici, che ci sono sempre stati, sia nei bei momenti che non. Il vostro supporto mi ha fatto crescere e mi ha fatto capire che potrò sempre contare su di voi, perché siete la mia seconda famiglia.

Abstract

Email spoofing and domain impersonation represent persistent threats to digital communications, enabling Business Email Compromise (BEC), financial fraud, and reputational damage. Despite the availability of authentication standards such as SPF, DKIM, and DMARC, adoption remains incomplete and enforcement policies are frequently misconfigured, reducing their preventive effectiveness.

This thesis investigates email authentication from a multidisciplinary perspective, integrating technical analysis, forensic methodologies, and legal accountability frameworks. First, the work examines the operational interplay between SPF, DKIM, and DMARC, highlighting common configuration weaknesses and their security implications. A technical and comparative evaluation of existing DMARC compliance and forensic tools is conducted, with particular focus on configuration-based exposure analysis and evidentiary preservation.

Building upon these foundations, the thesis introduces two original contributions. The first is a tool designed to evaluate domain-level spoofing exposure through lookalike domain detection, Internationalized Domain Name (IDN) homoglyph analysis, subdomain-based deception signals, and DNS authentication posture assessment. A risk scoring model is proposed to provide explainable and reproducible exposure metrics. The second contribution consists of a controlled multi-node email simulation environment that reproduces legitimate and malicious scenarios, including direct spoofing and third-party impersonation attacks. Through systematic header analysis and the correlation of authentication result, the experimental framework demonstrates how different configuration choices affect message delivery, rejection, and forensic traceability.

The results show that effective spoofing mitigation requires not only correct DMARC enforcement but also continuous monitoring, transport hardening, and domain risk analysis. Finally, the thesis argues that authentication configurations can serve as objective forensic artifacts in legal disputes, linking technical posture to domain owner responsibility.

Contents

Chapter 1

Introduction

Every day, billions of emails are sent around the world, facilitating business transactions and personal communications. Most people assume that when they receive an email from a bank, a colleague, or a trusted company, the sender is who they claim to be. Unfortunately, this assumption is often wrong.

1.1 The threat of email spoofing

Email spoofing is the practice of disguising an email to make it appear as if it comes from someone else. This has become one of the most prevalent and damaging forms of cybercrime. Criminals use spoofed emails to trick victims into revealing passwords, transferring money, clicking malicious links, or downloading malware. These attacks cost businesses and individuals billions of dollars annually and can damage organizational reputations. According to the FBI's Internet Crime Complaint Center (IC3), Business Email Compromise attacks resulted in losses exceeding $2.7 billion in 2024 alone [1]. Organizations face not only direct financial losses but also regulatory penalties, erosion of customer trust and operational disruptions.

The common mechanisms used to protect email are usually: spam filters, blacklists, and reputation-based systems. However, they do not really protect from email spoofing and business email compromise attacks, which are still very effective. The main purpose of these mechanisms is to find unsolicited bulk emails or known malicious patterns, but they do not to check if the sender is who they say they are. Modern spoofing and BEC attacks frequently lack malicious attachments or URLs, relying instead on social engineering and impersonation techniques that exploit inherent weaknesses of the SMTP protocol. As a result, spoofed emails often appear legitimate and bypass content-based and reputation-based filtering mechanisms, reaching recipients' inboxes without triggering security alerts.

1.2 DMARC as a solution for domain authentication

In order to address the structural limitations of traditional email security mechanisms, domain-based authentication protocols (such as DMARC) have been introduced.

Domain-based Message Authentication, Reporting, and Conformance (DMARC) is an email security protocol that helps organizations protect their domain name from being misused to send fraudulent emails. DMARC is built on other systems, called SPF and DKIM, which already help verify if an email is genuinely sent from an authorized source.

In practical terms, DMARC allows organizations to publish a set of rules ("policy") that tells email providers what to do if they receive a suspicious email or fail security checks (e.g., whether to deliver it, move it to spam, or block it altogether). Furthermore, DMARC requires email providers to send regular reports to the organization, helping them to see if anyone is trying to misuse their domain and letting them monitor which emails are being sent on their behalf.

In general, DMARC strengthens trust in email communication by ensuring that legitimate emails are more likely to be delivered, while malicious or fake emails are detected and stopped before reaching the recipient. This makes DMARC an essential tool not only for cybersecurity professionals, but also for organizations that want to protect their reputation and contacts from fraud [2].

The significance of adopting this protocol is illustrated by the following images (Figure 1.1). The figure on the left represents the percentage of DMARC usage in the EU in 2023. The countries with the highest support rates were Austria, Belgium, the Netherlands, and the Czech Republic, with rates exceeding 70%. Italy followed closely behind, with approximately 60% of the population in support.

The figure on the right presents the support rate for the year 2025. A marked increase in the utilization of DMARC has been observed. Indeed, the overwhelming majority of EU countries have demonstrated support rates in excess of 80%, with certain nations attaining rates of up to 100%.

1.3 Scope and objectives

The technical objective of this work was the design and the analysis of a DMARC verification tool, that is capable of: record syntax validation, assessing policy enforcement levels, identifying misconfigurations, and supporting forensic interpretation of email authentication data.

Beyond technical validation, this thesis explores the role of DMARC configuration as forensic evidence in legal disputes related to email fraud. Furthermore, the examination of how authentication records may support attribution and liability assessment is analyzed.

The contribution of this thesis lies in the integration of technical DMARC analysis with forensic and legal perspectives, thus providing a practical framework for assessing both security posture and potential domain owner responsibility.

1.4 Research questions (Email spoofing: motivation, risks, examples)

The practice of email spoofing poses a persistent and escalating threat in the context of contemporary digital communications. This threat is predicated on the exploitation of fundamental vulnerabilities inherent in protocols that were designed in an era preceding

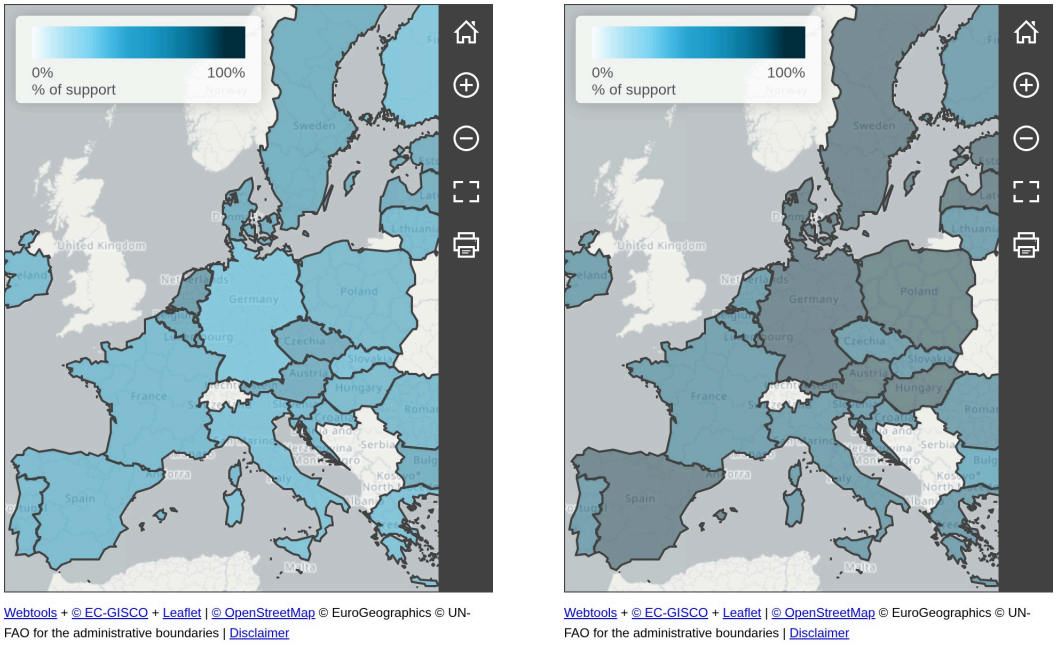

Figure 1.1: DMARC 2023 and 2025 Support Rate (source: ec.europa.eu)

the advent of widespread security concerns. The ease with which malicious actors can forge sender identities creates substantial risks for organizations and individuals. However, despite DMARC's demonstrated effectiveness in preventing domain impersonation, adoption rates remain concerningly low. Global statistics reveal that only 18.2% of the top 10 million domains have implemented DMARC records, and merely 7.6% of domains with DMARC have progressed to enforcement-level policies ("*p=quarantine*" or "*p=reject*") [3]. The support rate percentage of "*p=strict*" policy divided by country remains at a relatively low level, as illustrated in the Figure 1.2. The mean support rate percentage hovers around 30-40%, with Italy exhibiting a below-average rate of 25%. This predicament is particularly salient for the numerous companies that are susceptible to spoofing attacks without implementing countermeasures.

The impact of a successful email spoofing attack is best illustrated through real-world incidents, where attackers exploit weaknesses in email authentication and organizational trust mechanisms. Between April and May 2025, a bioscience company based in India suffered a fraud incident. Attackers impersonated a U.S.-based executive by falsifying the sender's email address and registering a domain name almost the same as the legitimate supplier. Through this impersonation, the Indian company was deceived into transferring ₹33.5 (≈ €400.000) to a fraudulent bank account. Thanks to the intervention of law enforcement authorities, a significant portion of the money was then recovered. [4]

Many years before, in 2016, the Austrian aerospace company *FACC* was hit by a

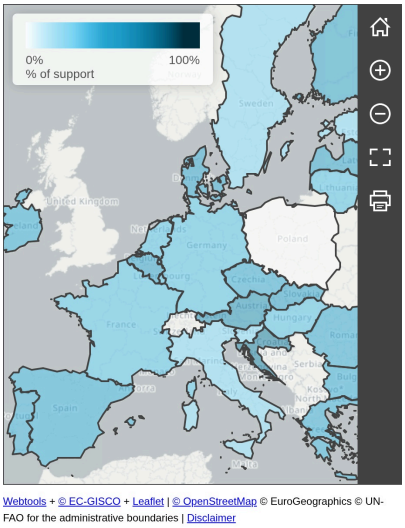

Figure 1.2: DMARC Strict Policy Support Rate - Q3 2025

cyber fraud that costed around €42 millions. Attackers acted by posing as the CEO of the company in an email. This fraudulent email asked an employee to transfer money for a fake project. The company did not disclose the manner in which the CEO violated his security duties; however, it was stated that, of the €42 million that was lost, only €11 million was able to be blocked from being fraudulently transferred. [5]

These cases highlight how the absence of of effective email authentication mechanisms can lead to substancial financial losses, despite the avilability of well-established technical countermeasures.

Organizations cite multiple barriers to DMARC enforcement, such as complexity of fragmented email infrastructures involving numerous third-party services, fear of blocking legitimate communications, lack of internal expertise.

This thesis will investigate how DMARC and supporting authentication technologies can effectively prevent email spoofing while examining the technical, organizational, and legal factors that either facilitate or impede comprehensive implementation.

Central research questions include:

- What technical and operational approaches most effectively detect and prevent email spoofing?
- What legal frameworks establish domain owner responsibilities for email authentication?
- What forensic methodologies enable attribution of spoofed emails?

1.5 Thesis structure

- Chapter 2 - Background: this section aims to provide the basic background knowledge needed to understand the main topic of this thesis work. It begins with an overview of Email fundamentals necessary to contextualize the problem, then SPF, DKIM and DMARC are explored in order to understand how they work together and how they should then be correctly configured.
- Chapter 3 - State of the Art: in this section, the state-of-the-art for DMARC implementation and configuration is analyzed.
- Chapter 4 - Legal Responsibility of Domain Owners: this section analyzes the legal accountability for domain owners and the risks they may face in case of violation.
- Chapter 5 - Insights of existing DMARC checker and Forensic tools: in this chapter, a first analysis of already commercial existing tools is executed.
- Chapter 6 - DMARC Forensic tool technical analysis, this section provides a deeper technical analysis of the tool published by Prof. Dal Checco, considering what features can be added, leading to the design of the domain analyzer tool later developed.
- Chapter 7 - Impersonation Risk assessment tool: in this chapter the first practical contribution is presented, providing examples and results.
- Chapter 8 - Experimental Email testbed for DMARC enforcement and forensic evidence collection: in this section is presented the second practical contribution, an experimental laboratory which simulates the effectiveness of domain-based authentication protocols in a controlled email environment.
- Chapter 9 - Conclusion: this final chapter summarises the obtained results and provides some analysis and ideas for future features integrations.

Chapter 2

Background

This chapter aims to provide an essential foundation for a better understanding of the technical and practical components of email and the mechanisms used to combat the problem of spoofing. The discussion then foregrounds the role of DMARC within the broader context of email security. An analytical synthesis of domain fundamentals, major authentication protocols and the rules behind their work is provided, then followed by exploration of configuration vulnerabilities and practical real-world cases. The objective is to deliver readers a depth of conceptual and operational knowledge necessary for advanced investigation into email spoofing prevention.

2.1 Domain and Email Fundamentals

The Internet Mail protocol is predicated on the collaboration between various components: Mail User Agent (MUA), Mail Transfer Agent (MTA), and the Domain Name System (DNS) infrastructure. DNS provides a distributed naming infrastructure that maps human-readable domain names to IP addresses, but it does not inherently provide authentication or assurance regarding the legitimacy of entities using a given domain name. As a result, domain identity in email communication is fundamentally logical rather than cryptographic, relying on conventions and cooperative behavior between mail servers.

SMTP is the protocol underlying most email delivery. In particular, SMTP distinguishes between the envelope sender specified during the `MAIL FROM` command and the header sender indicated in the `From:` field of the message headers; these identities are independent and are not required to match. The inherent openness of SMTP facilitates the use but also introduces exposure to malicious manipulation; specifically, it does not enforce sender authentication by default. This structural weakness is the crucible in which various spoofing and impersonation attacks are forged, as attackers can craft messages that claim arbitrary sender domains without violating the SMTP protocol. For this reason, additional authentication layers become indispensable for organizational email security [6].

In the context of email communication, several actors are involved. The **Author** of the message is responsible for its composition and the specification of the **Recipient**'s

address. Upon transmission, the message is relayed through an outgoing MTA, propagated through multiple intermediate MTAs and subsequently delivered to the recipient's inbox via an inbound MTA.

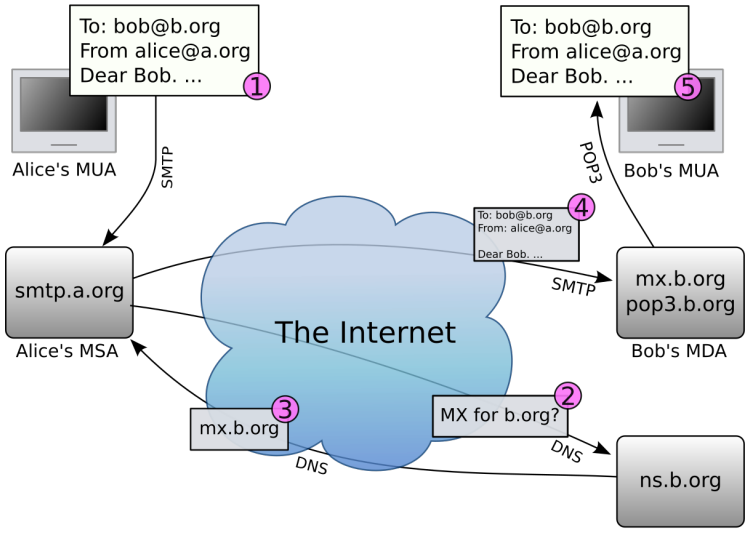

Figure 2.1: Email Architecture Scheme (source: Wikipedia)

2.2 The problem of Spoofing

Email spoofing represents a fundamental weakness of the email ecosystem rooted in the original design of SMTP, which lacks native mechanisms for sender authentication. As a result, attackers can forge email header fields(most notably the "From" address) to impersonate trusted domains, organizations, or individuals. Unlike traditional spam campaigns, modern spoofing attacks are often highly targeted and leverage social engineering rather than malicious payloads. For these reasons, these attacks nowadays frequently appear indistinguishable from legitimate business communications, allowing them to bypass content-based filters and reach recipients without triggering any security alerts.

In Business Email Compromise (BEC), adversaries exploit impersonation techniques to induce victims into performing fraudulent financial transactions. In such scenarios, attackers either spoof executive identities or compromise legitimate email accounts, thus observing ongoing conversations to inject fraudulent instructions at critical moments.

One of the most effective attacks happened in 2015, where the company "Ubiquiti Networks" disclosed a large-scale financial fraud, resulting in losses of approximately $46.7 million [7]. The forensic reconstruction reported that the attackers conducted a targeted spoofing campaign impersonating senior executives and trusted external partners. Emails were crafted to appear as much legitimate as possible, leading the finance department of the company to execute a series of international bank transfers. The attack exploited email header spoofing combined with social engineering techniques, leveraging

the lack of enforced sender authentication and domain alignment controls. The spoofed emails exhibited forged header fields, particularly within the "`From:`" and "`Reply-To:`" headers, which visually matched legitimate executive identities. However, the SMTP envelope sender (`MAIL FROM`) and the originating IP addresses were not cryptographically bound to the claimed sender domain. In the absence of enforced DMARC policies, receiving mail servers accepted these messages despite potential misalignment between the header "`From:`" domain and the authenticated identifiers. Since the messages did not contain vindictive attachments, embedded URLs, or abnormal content, content based spam filters and malware detection engines were not triggered. Traditional email security mechanisms, such as blacklist-based filtering and reputation scoring, were then ineffective, as the sending infrastructure was either newly registered or indistinguishable from legitimate mail servers. This case demonstrates how spoofed emails that are syntactically valid and contextually plausible can bypass classical defenses without exploiting any software vulnerability.

2.3 SPF (Sender Policy Framework)

Sender Policy Framework is an email authentication protocol that helps protect email senders and recipients against spoofing and phishing attacks. It allows a receiving mail server verify whether the sending IP address is authorized to send email on behalf of a given domain. SPF verifies that incoming emails originate from authorized mail servers by checking IP addresses against DNS records. An SPF record is a *TXT* record published in the Domain Name System (DNS). This DNS *TXT* record contains a list of all the IP addresses that are authorized to send emails from a given domain [8].

2.3.1 SPF record

Now let's see what an SPF record is composed of.

```
v=spf1 +mx a:colo.example.com/28 -all
```

Figure 2.2: SPF record example (source: RFC 7208)

SPF records use some specific Mechanisms to determine which IP ranges, hostnames and subdomains are allowed to send mails. Among these Mechanisms there are: "*ip4*", "*ip6*", "*mx*", "*include*", and "*exists*". Qualifiers, such as "+" (pass), "-" (fail), "∼" (softfail), "?" (neutral), are then used to determine how strictly these policies are enforced.

This SPF record (Figure 2.2) has:

- "`v=spf1`" - version of SPF
- "`+mx`" - mechanism that authorizes all hosts listed in the domain's MX records to send email; "`+`" qualifier indicates a "`Pass`" result; basically if the sending IP matches any IP address associated with the domain's MX hosts, the SPF check returns Pass.

- "a:colo.example.com/28" - the "a" mechanism authorizes IP addresses; associated with the IPv4 or IPv6 record of the specified hostname (colo.example.com); "/28" is the CIDR mask that limits authorization to a subnet of 16 IPv4 addresses;
- "-all" - "all" mechanism matches any IP address not previously matched, "-" qualifier represents "Fail": any sending IP that does not match one of the preceding mechanisms is explicitly unauthorized.

2.3.2 SPF functioning

At network level, when an email arrives at the recipient's mail server, it performs the following steps:

1. TCP connection establishment: the sending MTA initiates a TCP connection to the receiving MTA, without authentication;
2. SMTP handshake: client sends "HELO" command, identifying himself;
3. MAIL FROM command: the sending MTA transmitsS the SMTP command "MAIL FROM:<user@example.com>". This command defines the "Return-Path" address; SPF validation will then be based on the domain extracted from this field, authenticating only the envelope sender domain.
4. The receiving MTA extracts, through a DNS TXT query, the domain name from the "MAIL FROM" or "Return-Path" address in the email header:

   ```
   example.com. IN TXT "v=spf1 ip4:x.x.x.x/24 include:_spf.provider.com -all"
   ```

5. IP authorization check: the receiving MTA compares the source IP address of the SMTP connection with the mechanisms specified in the SPF record.
6. The SPF evaluation produces one of several results:
   - **SPF Pass**: The sending IP matches an authorized IP; message accepted
   - **SPF Fail**: The sending IP is explicitly unauthorized; message potentially rejected
   - **SPF Softfail**: The sending IP is probably unauthorized but treated leniently
   - **SPF Neutral**: No definitive authorization or rejection
   - **SPF None**: No SPF record found for the domain

   These results are then recorded internally and often surfaced in the message header.

2.4 DKIM (DomainKeys Identified Mail)

DomainKeys Identified Mail (DKIM) is a protocol that uses public key cryptography to verify the authenticity of emails and to detect any tampering attempt during transmission. DKIM is a digital signature technology that confirms the authenticity of an email's sender. Unlike SPF, DKIM operates at the message level - rather than the transport level - and remains valid across message forwarding [9].

2.4.1 DKIM Signature

Looking at the Figure 2.3, let's analyze the various fields of a DKIM Signature:

```
DKIM-Signature: v=1; a=rsa-sha256; d=example.net; s=brisbane;
     c=relaxed/simple; q=dns/txt; i=foo@eng.example.net;
     t=1117574938; x=1118006938; l=200;
     h=from:to:subject:date:keywords:keywords;
     z=From:foo@eng.example.net|To:joe@example.com|
       Subject:demo=20run|Date:July=205,=202005=203:44:08=20PM=20-0700;
     bh=MTIzNDU2Nzg5MDEyMzQ1Njc4OTAxMjM0NTY3ODkwMTI=;
     b=dzdVyOfAKCdLXdJOc9G2q8LoXSlEniSbav+yuU4zGeeruD00lszZ
              VoG4ZHRNiYzR
```

Figure 2.3: DKIM Signature example (source: Wikipedia)

- "`v=1`" represents the DKIM version
- "`a=rsa-sha256`" is the cryptographic algorithm used
- "`d=example.net`" is the Signing Domain Identifier (SDID), ensures that mail has been signed by that specific address;
- "`s=brisbane`" is the selector, it is used to locate the public key to adopt and allows key rotations
- "`c=relaxed/simple`" is the canonicalization, represents how header and body have been normalized before the hash
- "`t`" and "`x`" are the timestamp and expiration date, useful against replay attacks;
- "`h`" represents all the headers that are included in the signature, if one of them changes, DKIM check fails;
- "`bh`" is the hash of the body, which is compared from the recipients;
- "`b=dzdV...`" is the signature.

This entire signature ensures that the domain "`example.net`" is the one who signed the message at the time "`t`", the headers "`h`" have not been modified and the body respects the integrity checks.

2.4.2 DKIM functioning

The first step performed by the sender domain to use DKIM is to generate an asymmetric key pair, one public and one private, that will then be used in the authentication process. The private key will be securely stored in the sending MTA, while the public key will be published in DNS as a TXT record:

```
selector1._domainkey.example.com. IN TXT "v=DKIM1; k=rsa; p=MIIBIjANBgkq..."
```

Once the message is submitted, the MTA sender computes hashes on header fields using canonicalization algorithms ("simple" or "relaxed"). Subsequently, the sender's private DKIM key is used to encrypt the hash, thus producing the DKIM signature. Before the email transmission, the digital signature is appended and incorporated into the DKIM Signature field header of the email. The signed message is then transferred using the SMTP "`DATA`" command. Looking at packet level, DKIM signature is part of the email payload, for this reason it will not be altered by intermediate MTAs.

The recipient's MTA, once it receives the email, extracts the DKIM parameters ("`d`", "`s`", "`h`") and performs a DNS query for retrieving the public key. After this, hashes are recomputed using the same canonicalization rules and the retrieved public key will be used for the cryptographic validation of the signature. The result will be recorded in the authentication metadata:

```
"Authentication-Results: dkim=pass header.d=example.com"
```

Once the receiving server has verified the presence of a valid DKIM signature on an email, it can be deduced that the integrity of the email has been maintained. It is important to note that, in the majority of cases, DKIM signatures are not visible to end-users, and the validation process occurs at the server level. This entire process ensures that the integrity of the email has been maintained since its transmission.

2.5 DMARC (Domain-based Message Authentication, Reporting and Conformance)

Domain-based Message Authentication, Reporting, and Conformance (DMARC) is an email authentication protocol designed to limit email spoofing and domain impersonation attacks. DMARC is built on SPF and DKIM, and the combined work of these protocols helps strengthen the effectiveness by addressing authentication gaps that neither protocol can resolve independently. DMARC was originally developed as an email security protocol at the DNS level, designed in 2012 by Microsoft, Google and Yahoo [2].

When an email arrives, the recipient's server performs a DNS lookup searching for a DMARC record (after conducting SPF and DKIM checks). DMARC checks that the authenticated identifiers match the domain contained in the "`From`" header. For SPF alignment, DMARC checks if the domain from the "*envelope from*" address (used in SPF) matches the "`From`" header domain. However, for DKIM alignment, it verifies whether the "`d=`" domain in the DKIM signature matches the "`From`" header domain. Alignment can be configured as: "`strict`", meaning that it requires the exact domain match; or "`relaxed`", thus allowing matching base domains with different subdomains. The DMARC check is satisfied if either SPF or DKIM passes authentication controls with respect of the alignment test, one is already sufficient. Domain owners have the possibility, with DMARC, to publish their policies that specify how receivers should handle messages that failed authentication. There are three policy levels:

1. **p=none**: no action taken for unauthenticated emails, but DMARC records are sent to the domain owner;

2. **p=quarantine**: failed messages are sent to SPAM inbox;
3. **p=reject**: messages are not delivered to the recipient inbox.

Usually, domain owners start with `p=none` in order to monitor the authentication requests and then go into more restrictive policies.

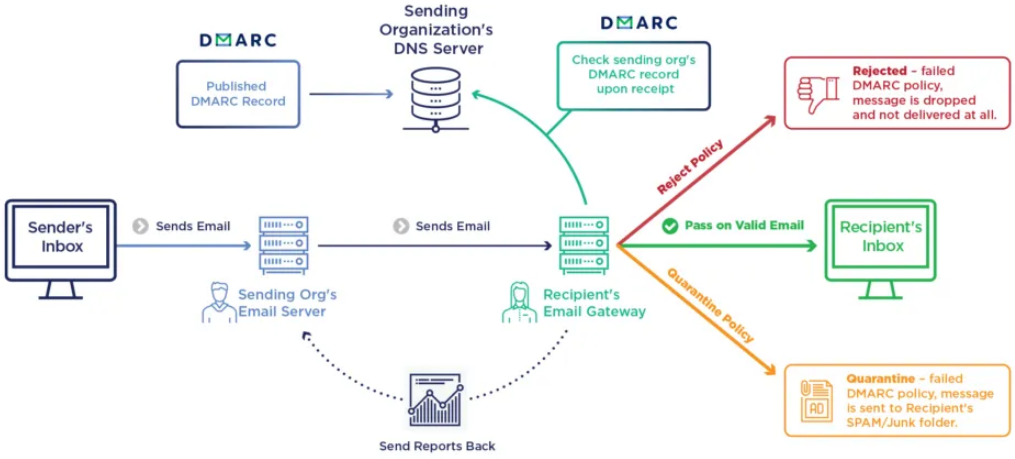

Figure 2.4: DMARC Architecture (source: emailauth.io)

There is an example of DMARC TXT record:

```
"v=DMARC1; adkim=r; aspf=s; p=none; pct=100; rf=afrf; ri=86400;
                ruf=mailto:forensics.example.com;"
```

Figure 2.5: DMARC record (source nist.gov)

Let's analyze the tag present:

- **v=DMARC1** - specifies that this TXT is a DMARC record;
- **adkim=r** - says that the DKIM domain must be evaluated with a relaxed basis, so the organizational domain in the record must be the same as the one in the "From" domain;
- **aspf=s** - specifies that the SPF domain must be evaluated in a strict way;
- **p=none** - DMARC policy not applied on failing messages;
- **pct=100** - percentage of messages which policies are applied to;
- **rf=afrf** - specifies the individual forensic report format (AFRF in this case);
- **ri=86400** - interval in seconds to deliver summary reports;

- **ruf=mailto:forensics.example.com** - address where the forensic report (which contains messages that fail DMARC checks) should be sent.

For domain owners, DMARC implementation reduces spoofing risks, provides unprecedented visibility into email ecosystems, and improves deliverability by signaling rigorous authentication practices to major email providers. For recipients, DMARC offers enhanced protection against phishing and business email compromise attacks by ensuring only authenticated emails from verified sources reach the inbox. The protocol's evolution from voluntary best practice to industry requirement reflects its proven effectiveness in combating email-based threats.

2.6 How SPF, DKIM and DMARC work together

The joint effort between these three protocols helps strengthen protection against the threat of email spoofing, adding an additional layer of security. SPF asserts the authenticity of the sending infrastructure, while DKIM evaluates the cryptographic identity of the message. Although SPF and DKIM are often deployed together, their independent validation does not inherently prevent domain impersonation. SPF authenticates the SMTP envelope sender, while DKIM authenticates the domain accountable for applying the cryptographic signature. In the absence of a definitive alignment mechanism, a message may successfully pass SPF and DKIM checks while still presenting a deceptive sender identity in the "`From`" header. Consequently, relying on SPF or DKIM alone (or even both) without correlation does not provide sufficient protection against spoofing attacks targeting the domain identity perceived by the end user. Meanwhile, DMARC integrates these two mechanisms, ensuring that the authentication results and the domain presented to the end-users match.

The workflow is typically the following:

1. The recipient's server performs SPF and DKIM controls;
2. DMARC policy is consulted to determine if at least one check between SPF or DKIM is satisfied;
3. based on DMARC conditions, the recipient performs the preferred action
4. optionally, reporting mechanisms allow administrators to review authentication statistics.

2.6.1 Authentication results correlation and data flow

These three protocols, when combined, establishe a layered defense model, thus reducing opportunities for attackers to exploit isolated weaknesses. The combined operation of SPF, DKIM, and DMARC occurs within a single SMTP transaction, during which multiple authentication mechanisms are evaluated at different stages of message processing and their results are subsequently correlated. Rather than introducing an additional authentication layer, DMARC acts as an orchestrator, consuming the outputs of SPF and DKIM and enforcing domain-aligned policy decisions.

From a protocol-level perspective, during the SMTP envelope phase SPF evaluation is triggered, specifically after receiving the "`MAIL FROM`" command. At this stage, the receiving MTA extracts the envelope sender domain and performs a DNS TXT query to retrieve the corresponding SPF record, comparing the source IP address of the SMTP connection to the authorized sending mechanisms defined in the record, as specified in RFC 7208 [8]. The evaluation result (pass, fail, softfail etc.) will be then determined before the message transmission and it is bound to the envelope sender.

On the other hand, DKIM is performed after SMTP "`DATA`" command, thus after the full message (including header and boy) has been received. The recipient MTA extracts the DKIM-Signature header, retrieves the public key through a DNS TXT query and then verifies cryptographically both message integrity and domain authorizaion, according to RFC 6376 [9]. DKIM validation is thus independent from the sending IP address and it is kept valid across message forwarding, but DKIM alone does not asseverate that the signing domain corresponds to the sender identity that arrives to the user.

After SPF and DKIM results are available, DMARC evaluation is executed. Through a DNS TXT query the DMARC record is retrieved, then the domain is extracted from the visible "`From`" header. DMARC does not re-evaluate SPF or DKIM, but instead it correlates their authentication results with the domain shown to the recipient and verifies the alignment conditions, requiring that either the SPF-authenticated domain or the DKIM signind domain match the "`From`" domain. After this, the receiving system applies the policy specified by the domain owner [2].

During the entire process, the receiving MTA performs multiple DNS lookups within the same authentication context, including the SPF records for the envelope sender, DKIM key records for the signing domain and DMARC policy records for the header sender. The results of all these evaluations are put in the "`Authentication-Results`" header. This header documents SPF, DKIM and DMARC results and provide a strong forensic trace of all the authentication decisions applied to a given message during its transmission [10].

This integrated workflow addresses the structural weaknesses of SMTP's trust model and establishes a deterministic mechanism for preventing domain impersonation and large-scale spoofing attacks.

2.7 Additional relevant technologies (MTA-STS, BIMI)

MTA-STS (Mail Transfer Agent Strict Transport Security) and BIMI (Brand Indicator for Message Identification) extend the email architecture, securing the transport layer and providing visual authentication indicators to recipients. MTA-STS allows domain owners to publish policies via DNS and HTTPS, specifying that the recipient's mail server must support TLS encrypted connections, in this way malicious users are prevented to perform TLS downgrade attacks. However, if security requirements cannot be met, the message is rejected rather than transmitted in plaintext. Three policies are offered by MTA-STS: **none** (no enforcement at all), **testing** (messages are monitored but not blocked), **enforce** (full compliance is mandatory).

BIMI, on the other hand, works as a visual trust signal. It allows verified companies to

display the sender's logo in the recipient's inboxes. Only those emails that pass rigorous DMARC controls (with quarantine or reject policy), are eligible to display logos. BIMI is implemented by companies through the publication of a DNS record that specifies the location of the logo and links it to a certificate (Verified Mark Certificate or Common Mark Certificate), cryptographically binding the ownership of the logo to that organization. Despite this multi-layered approach reduces exposure to spoofing, phishing, and man-in-the-middle attacks, the actual adoption of MTA-STS and BIMI remains incomplete due to implementation complexity, cost, and incomplete support across all email clients.

2.8 Possible configuration errors

The effectiveness of SPF, DKIM and DMARC heavily relies on correct configuration, errors can directly undermine security or reduce communications's reliability.

Among the common missteps there are:

- **Incomplete or outdated SPF records** - if all the legitimate mail sources are not specified, this will cause a lot of false positives;
  - From a technical perspective, this error manifests when the source IP address of the SMTP connection is not included in the domain's SPF record. Typical results include "*SPF fail*" when the record terminates with "`-all`", or "`SPF softfail`" when "∼all" is used. Furhtermore, complex SPF records that rely on multiple "`include`" mechanisms may exceed the RFC-mandated limit of 10 DNS lookups, resulting in a "`permerror`".
- **DKIM key erroneous management** - rotating keys without updating DNS record entries can cause verification failures;
  - These failures usually arise when the public key published in the DNS does not match the private key used to sign messages. This may happen during key rotation if the DNS record is not updaated, or if the selector referenced in the "`DKIM-Signature`" ("`s`" in the Figure 2.3) does not exist. Moreover, if the message is modified after signing or if strict canonicalization ("`c=simple/simple`") is adopted, may conduct to a signature invalidation.
- **Misalignment of DMARC policies** - not enforcing strict alignment may lead to insufficient protection;
  - Misalignment failures occur when the domain authenticated by SPF or DKIM does not match the one present in the "`From`" header. Due to this failure, an email may pass SPF authentication using a third-party envelope sender while presenting a different organizational domain to the user (e.g. "`From: example.com`" and "`MAIL FROM: mail.thirdparty.com`"). Additionally, when subdomains are involved, strict alignment (like "`asps=s`" or "`adkim=s`") may increase failure likelihood.

- **Inadequate monitoring** - overlooking reports can allow unnoticed attacks or misconfigurations to persist.
  - If DMARC policies are set to "`p=none`", monitoring is enabled but it does not enforce any action against unuthorized messages. As a result, spoofed emails may be delivered as well, making DMARC controls ineffective.

The mitigation of all these errors necessitates a combination of technical expertise and institutional processes that are oriented towards ongoing maintenance, monitoring, and periodic validation of authentication frameworks.

2.9 Practical implications and case studies

This framework of SPF, DKIM and DMARC finds a lot of critical application in real-world scenarios, since implementation decisions carry measurable security and operational consequences.

It is evident that public-access tools such as Emkei and AnonyMailer empower individuals, even without technical expertise, to formulate communications that appear being originated from arbitrary email addresses, including potentially sensitive institutional accounts. This accessibility underscores the necessity of robust authentication infrastructure. Absent proper configuration of SPF, DKIM and DMARC at the DNS and mail server levels, receiving mail servers possess no technical means to distinguish legitimate correspondence from spoofed impersonations.

In a direct spoofing scenario (typical of the CEO fraud scenario), the attacker establishes an SMTP session with a receiving mail server and forges the `From` header to impersonate a trusted domain. If SPF `fail` or `pass` is not aligned and DKIM is not present (`dkim=none` or `d` $\neq$ `From`) and if the target domain lacks a DMARC record or publishes a permissive policy (`p=none`), the receiving server will accept the message [2], [8]–[10].

```
Authentication-Results:
  spf=fail smtp.mailfrom=attacker.com;
  dkim=none;
  dmarc=fail (p=none)
```

More sophisticated attacks leverage legitimate third-party email infrastructure (e.g., marketing platforms or cloud mail relays) to bypass basic authentication checks. In such cases, SPF may pass for the envelope sender, while the visible sender identity remains misleading [11]. Technical detail:

- SPF: `pass` (third-party is authorized)
- DKIM: `pass` (`d=thirdparty.com`)
- `From: ceo@victim.com`
- DMARC: if relaxed alignment is set, it will fail the check; if `p=none`, the mail si going to be delivered.

The header could be:

```
spf=pass smtp.mailfrom=mailer.thirdparty.com;
dkim=pass header.d=mailer.thirdparty.com;
dmarc=fail header.from=victim.com
```

The ability of receiving servers to detect and reject counterfeited mails depend entirely on correct configuration choices.

The practical forensic significance of DMARC configuration emerges in the investigation of email-based fraud (Man-In-The-Mail) [12]. In such cases, attackers intercept organizational communications to redirect financial transfers to their accounts. These fraudulent messages are injected into existing threads, often reusing original message headers and subjects to evade detection. A verification that a domain maintained strict DMARC enforcement, valid DKIM signatures, and appropriate SPF restrictions at the time of the disputed communication establishes that spoofed messages impersonating the organization would have been technically rejected by properly configured receiving systems. On the other hand, the presence of a permissive (or the total absence) of authentication configuration demonstrates the organization vulnerability to spoofing. The heterogeneity of institutional authentication maturity demonstrates that email security is a field where organizational choices are crucial in real-world security outcomes.

Chapter 3

State of the Art

Contemporary approaches to email spoofing prevention and forensic identification have evolved substantially in the last ten years. In this chapter the current state of the art in email security will be examined, exploring both theoretical frameworks and practical implementations designed to address the persistent challenge of email spoofing with particular emphasis on forensic identification and evidence preservation.

3.1 Enhanced Email Security approaches

Since email threats evolved from simple spam and phishing attempts to sophisticated social engineering attacks, a need for robust defense mechanisms arose.

Nowadays, email security architectures implement multilayered defense strategies, extending beyond the traditional authentication protocols to incorporate behavioral analysis, content inspection and real-time threat intelligence. These enhanced approaches recognize that email threats are not simple spam and phishing anymore, but they evolved to sophisticated social engineering attacks.

Modern email security gateways, in order to provide an SMTP-level enforcement, started to perform pre-delivery analysis during the SMTP transactions, before the message acceptance. At this tage, decisions are made based on transport-layer metadata such as: IP source address, `HELO/EHLO` identity, reverse DNS resolution, TLS parameters and historical reputation scores. All of these deicions are taken before the `DATA` SMTP command, analyzing: IP reputation, geolocation and SMTP command timing. The results could lead to: 550 `reject` or 421 `tempfail` [13].

After message acceptance, advanced security systems perform fine-grained parsing of email headers, reconstructing the full SMTP relay chain from the sequence of `Received` fields. All those anomalies involving inconsistent timestamps, non-resolving hostnames, forged `Message-ID` formats, and mismatches between envelope and header domains are used as indicators of spoofing or fraud. Detailed header-level analysis techniques are further examined in 3.3.

Contemporary email security platforms integrates advanced threat protection with sandbox analysis, real-time attachment and URL scanning, machine learning-based anomaly

detection, and policy-based message authentication. Sandbox technology represents a significant advancement in the field, offering isolated environments in which to analyze emails and attachments without exposing organizational networks to potential threats. These sandboxes employ emulation tools and behavioral analysis to identify zero-day exploits, polymorphic malware, and targeted attacks that are even able to evade signature-based detection systems. These types of analysis are usually based on dynamic tracing rather than signature matching [14].

Next-generation email security solutions integrate conventional threat intelligence with behavioral anomaly detection, meticulously analyzing an extensive array of data points. These mechanisms also encompass linguistic patterns, tone, sentiment, sender profile characteristics, and historical communication behaviors, thus facilitating the identification of sophisticated threats that rely exclusively on social engineering, thereby circumventing the use of malicious payloads or links [15].

3.2 Machine learning and AI in spoofing detection

The use of AI, which has been growing exponentially in recent years, has also found application in phishing and spoofing detection mechanisms, enabling systems to identify threats through pattern recognition and behavioral analysis rather than reliance on known signatures alone. The study [16] demonstrates that machine learning models trained on extensive datasets of legitimate and malicious emails can recognize subtle textual cues, structural anomalies, and behavioral deviations indicative of spoofing attempts.

An arXiv 2022 paper [13] by Beaman and Isah focused on anomaly detection in emails using machine learning and header information. The result of this research demonstrated that email header information is sufficient to reliably detect spam and phishing emails. Supervised learning algorithms such as Random Forest, SVM, MLP and their ensembles achieved high accuracy scores of 97% for phishing and 99% for spam emails. One-class classification with One-Class SVM achieved accuracy scores of 87% and 89% with spam and phishing emails respectively. This approach is particularly relevant for DMARC integration, as it demonstrates that authentication metadata can be highly effective for threat detection.

In 2020 [17] Kaspersky implemented a combination of DMARC technology with machine learning in order to minimize false positives, while protecting against phishing. When fail DMARC checks messages are received, Kaspersky's AI technology analyzes header sequences and content using neural networks trained on approximately 140 million messages (40% spam). This combination ensures an effective user protection from phishing attacks while minimizing false positives.

3.3 Email Header analysis and technical forensics

The analysis of email headers is a fundamental forensic technique used to investigate email authenticity, track message origins and reconstruct transmission pathways. The importance of the headers is given by their structured metadata fields that document the complete history of a message, even including the sequence of MTAs through which it

passed, originating IP addresses, timestamps, authentication results and message identifiers.

Email headers play an important role in the identification of sender and receiver (Figure 3.1), and through their analysis some information can be collected, such as:

- From, To, CC, BCC
- Reply-to, Content Type
- Message ID, Important Level
- Network Path, X-Mailer
- MIME Info
- Email client information
- Time stamp, SMTP information
- Encoding details

In particular, the *"Received"* header chain provides chronological documentation of each server hop during message transmission. Forensic investigators meticulously analyze this chain to identify anomalies such as unexpected geographic routing, inconsistent timestamps, suspicious intermediate relays, or IP addresses associated with known malicious infrastructure. The process of reverse DNS lookups on IP addresses extracted from headers facilitates the identification of hostnames associated with sending servers. This, in turn, enables investigators to determine the alignment between these hostnames and the domains claimed by the servers.

Other critical header field for forensic analysis is the Message-ID, which provides a unique identifier for each email and it is useful to reveal forged messages when the format differs from known patterns for specific mail system. The Return-Path or envelope-from address indicates where delivery failure notifications should be sent, and especially in spoofed messages they are different from the visible From address. The authentication result headers are also useful because they document SPF, DKIM and DMARC validation outcomes performed by receiving servers.

The process of content examination serves as a complementary analysis to header analysis. It involves the investigation of email body text, embedded objects, and attachments. This methodical approach seeks to identify any potential indicators of malicious intent. Forensic investigators meticulously analyze linguistic patterns, urgency indicators, and social engineering tactics employed within message content. They then correlate these elements with header anomalies to construct comprehensive threat assessments.

3.4 Autorship attribution

Stylometric analysis is defined as the quantitative study of writing style. Autorship attribution leverages stylometric analysis to identify or verify the authors of anonymous disputed email messages. This is particularly useful in cybercrime investigations where

Figure 3.1: Email Header Fields example (source: Forensicsware)

determining the identity of individuals behind malicious email campaigns can enable prosecution and deterrence. In practical implementations, authorship attribution relies on the extraction of multi-level stylometric features, including: lexical features (word frequency distributions and vocabulary richness), character-level features (character n-grams, punctuation usage), syntactic features (part-of-speech sequences, sentence length distributions), and structural features such as greeting formulas, signature patterns, and formatting conventions.

More advanced Authorship attribution methodologies incorporate Machine and Deep Learning techniques that automatically learn stylistic features from training corpora. Neural network architectures such as Convolutional or Recurrent Neural Networks can model sequential dependencies in text, capturing write style representations that differs from hand-crafted feature sets [18]. These models have been developed to learn distributed representations of words that are enriched with linguistic features, including topical, lexical, syntactic, and character-level features. These features are designed to replicate human sentence composition processes. The effectiveness of stylometric authorship attribution depends critically on several factors such as the availability of sufficient comparison texts from candidate authors, consistency of writing style across documents, and absence of deliberate obfuscation attempts. One of the most used tool is JGAAP, an open source tool that through machine learning techniques helps with authorship attribution problems. JGAAP is a modular authorship attribution framework that implements a configurable pipeline for stylometric feature (such as lexical frequences, character and

word n-grams, punctuation patterns) extraction and classification, enabling probabilistic attribution and similarity analysis across textual datasets. All the extracted features are then converted into "event sets", thus numerical representations of the author's style, making it possible to compare texts in a quantitative way. For the final result JGAAP applies classification and distance algorithms in order to attribute a text to a specific author or to evaluate the similarity between multiple texts [19].

In email investigations, stylometric attribution is enriched by domain-specific indicators, such as reply structure, quotation style, use of subject prefixes (e.g., "*Re:*", "*Fwd:*"), time-of-day sending patterns, language switching, and consistency with historical organizational communication norms.

The Forensic application of autorship attribution include the use of clustering techniques to group anonymous emails by likely author. This enables investigators to link disparate messages to common sources, even when the absolute identity of the author cannot be established. This capability is valuable in the context of spam and spoofing campaigns, since clustering by autorship reduces the investigative workload by grouping related messages for a collective analysis rather than individual examination. Adversaries may deliberately alter writing style or employ AI-assisted text generation to evade stylometric detection. As a result, authorship attribution must be treated as probabilistic evidence and corroborated with technical indicators such as authentication metadata, routing paths, and infrastructure reuse.

3.5 Integration with security operations (SIEM integration)

The integration of email security systems with Security Information and Event Management platforms signifies a pivotal advancement in organizational threat detection and incident response capabilities. SIEM systems are designed to aggregate, normalize, and correlate security event data from diverse sources across IT infrastructure. This capability enables holistic threat analysis, which identifies attack patterns that are invisible when examining individual systems in isolation. [20]

SIEM systems function as the primary investigative tools in the realm of cybersecurity. They help in the detection of phishing and spoofing through:

- **Centralized Monitoring**: SIEM consolidates logs from multiple sources, providing a comprehensive overview of network activities, facilitating the identification of anomalous patterns.

- **Correlation of Events**: SIEM tools are designed to analyze patterns by correlating events that appear to be unrelated to one another across the network. For example, the receipt of an email that appears to originate from an internal executive address, followed by multiple failed authentication attempts from an external IP, could be indicative of an email spoofing attempt.

- **Automated Alerts and Responses**: SIEM systems enable the immediate identification of potential spoofing attacks.

- **Real-Time Threat Intelligence**: Integration with threat intelligence feeds enables SIEM systems to remain updated on the latest spoofing tactics and malware signatures.

SIEM	Data ingested	Detection logic
Splunk Enterprise Security	Email gateway logs; parsed headers (From, Return-Path, Authentication-Results); DMARC reports; DNS telemetry; user-reported phishing.	Correlation of DMARC failures with delivered messages; detection of domain impersonation via sender mismatch; behavioral baselining of sender domains; campaign-level time correlation.
Elastic Security	SMTP/MTA logs; email gateway logs; full header parsing (Received chain, Authentication-Results); DMARC reports; DNS logs with enrichment.	Query-based detection of SPF/DKIM failures and misalignment; correlation of spoofed domains with anomalous routing paths; forensic timeline reconstruction.

Table 3.1: Comparison of SIEM-based email spoofing detection

The table 3.1 summarizes SIEM approaches for email spoofing detection [2], [8]–[10], [21], [22]. While both SIEM platforms ingest comparable email-related telemetry, Splunk Enterprise Security emphasizes rule-based correlation and behavioral analytics for alert generation, whereas Elastic Security provides greater flexibility for custom query development and forensic timeline reconstruction.

SIEM has been shown to enhance phishing and spoofing detection capabilities by examining elements beyond the content of the email. Furthermore, SIEM correlates email activity with other threat indicators, such as login anomalies or external web traffic, to detect phishing attempts that bypass standard filters. This ability to provide deep detection across multiple vectors gives SIEM an advantage over standalone email filtering solutions. Practical implementations demonstrate measurable improvements in security posture: financial institutions integrating DMARC monitoring with Splunk SIEM have achieved real-time alerting on unauthorized sender attempts, automated correlation of authentication failures with threat actor infrastructure, and substantial reductions in manual DMARC report analysis workload [23].

3.6 Emerging technologies

In response to evolving email security challenges, novel technological approaches continue to emerge, particularly those focused on detecting AI-generated phishing attempts and

implementing decentralized authentication architectures.

3.6.1 AI-generated phishing and spoofing detection

The proliferation of generative AI tools, which can produce human-quality text, and deepfake technologies capable of creating convincing audio and video impersonations, has introduced sophisticated attack capabilities accessible to low-skill adversaries.

Nowadays Generative AI serves as both the threat vector and a defensive tool. New generation AI-driven security solutions analyze email content using multimodal analysis that examines text, visual elements, metadata and behavioral patterns, thus making it possible to detect AI-generated phishing emails. In addition to such generic content-based detection, many studies and commercial offerings use AI models that have been trained with email metadata, authentication outcomes and communication patterns to recognize spoofing and impersonation attacks. These systems blend the results of linguistic analysis with protocol-level signals (SPF/DKIM/DMARC verdicts, sender reputation, and historical interaction history) to detect social-engineering attacks when there are no malicious payloads [13], [15], [16].

3.6.2 Blockchain-based email systems

A decentralized approach to email authentication is offered by the Blockchain technology, which addresses vulnerabilities in traditional centralized email infrastructures. With Blockchain-based architectures, the email infrastructure is distributed across peer-to-peer networks, eliminating centralized control and enabling direct user-to-user message routing. In such type of communications, each email transaction is protected by a cryptographic signature, thus ensuring that only authorized recipients are allowed to read the sent emails. Several research proposals explore blockchain-based mechanisms to redesign email trust and identity binding (e.g., smart-contract-based email protocols and certified email without a conventional TTP). While these approaches can, in principle, reduce spoofing by strengthening identity guarantees, they typically require architectural changes and face interoperability and deployment constraints in SMTP ecosystems [24].

Blockchain email platforms allow to reduce the spoofing problem, integrating with decentralized domain registries. With the verification of the ownership of the domains on the blockchain, it is possible to drastically reduce spoofing while giving users full control on their identities and accounts. Finally, this decentralized paradigm offers an alternative to the classic centralized communications but, at the same time, ensuring security, privacy, ownership and availability. Despite theoretical advantages documented in technical literature, blockchain email systems face significant adoption challenges including scalability limitations that introduce latency in high-volume environments, implementation costs associated with blockchain infrastructure, interoperability difficulties communicating with traditional email systems, and user experience complexity requiring cryptographic key management [25].

3.7 Legal and Forensic Challenges (evidence collection and preservation, attribution and investigation)

The use of email evidence in legal proceedings and forensic investigations presents complex challenges spanning authentication, preservation, attribution, and admissibility requirements that vary across jurisdictions, as documented in forensic and legal literature.

3.7.1 Evidence Collection and Preservation

Rigorous protocols are required to maintain authenticity of digital evidences and prevent spoliation throughout collection, analysis, and presentation phases according to forensic standards. Best practices require the use of forensically sound collection methods and specialized tools to capture emails without altering the metadata or header information. This is typically achieved through server-level extraction rather than user-initiated copying, which could cause the compromission of the integrity of the evidence. Creating master and working copies ensures that the original evidence is preserved while enabling analysis of duplicates, for this reason Master copies must be secured under strict access controls.

A critical evidentiary requirement in forensic literature is the established Chain of custody documentation, which provides chronological records tracking every individual who accessed evidence, the date and duration of access, the purpose of examination, and any actions performed. This meticulous documentation allows courts to verify that evidence remained unaltered between collection and presentation, establishing authenticity and defeating challenges regarding tampering or contamination. Failure to maintain complete chain of custody records can result in evidence exclusion regardless of its probative value, as demonstrated by legal precedents [26] [27].

3.7.2 Attribution and Investigation

Attribution involves determining the identity of individuals or entities responsible for malicious email activities. It is a fundamental challenge in cybercrime investigations, as documented in law enforcement research. Techniques and technologies that enhance anonymity such as the use of VPNs, Tor networks, and proxy services, complicate attribution by obscuring the real IP addresses and locations. When investigators even successfully trace emails to specific IP addresses, obtaining subscriber information from Internet Service Providers requires appropriate legal process including subpoenas, search warrants, or court orders. Attribution assessments use systematic methods to give each evidentiary artifact a reliability and credibility score. This enables the evaluation of confidence levels in attribution conclusions.

A research on email spoofing attacks [11] demonstrated that there are significant vulnerabilities in authentication chains involving multiple protocols, roles, and services, where any failure can break chain-based defenses. The analysis of 30 popular email services and 23 email clients found that all of them were vulnerable to certain types of attacks capable of bypassing SPF, DKIM, DMARC and user-interface protections. These

findings underscore attribution challenges when sophisticated attackers exploit protocol inconsistencies to obscure their identities.

Furthermore, cross-border investigation complexities arise when email servers, sender locations, and victim jurisdictions span multiple countries: These investigations require international legal cooperation through mutual legal assistance treaties and adherence to diverse legal frameworks governing data access and privacy protection. For this reason investigators must navigate legal tensions introduced by conflicting privacy regulations, such as the European Union's General Data Protection Regulation, which can prohibit evidence collection methods that are permissible in other jurisdictions.

3.7.3 Legal Admissibility Challenges

The legal admissibility in legal proceedings of email evidence is contingent upon the satisfaction of authentication, relevance, and hearsay exception requirements under applicable evidentiary rules as established in legal scholarship.

Authentication necessitates substantiation that the emails intended as evidence originate from the stated senders and have remained unaltered. This is typically established through digital forensic analysis of metadata, header examination, and hash verification. Consequently, it is frequently essential to solicit the testimony of computer forensic experts who have collected and analyzed the evidence to establish its authenticity and explain the technical collection methodologies to the courts. Forensic experts must prepare for challenges addressing collection methodology, preservation integrity, and analysis reliability, since it is mandatory the presentation of original emails or properly authenticated duplicates, with forensically preserved copies including complete metadata satisfying this requirement according to evidentiary standards. Furthermore, privacy law compliance, particularly regarding employee email monitoring and third-party service provider data access, requires careful navigation to avoid exclusion of improperly obtained evidence.

The evidentiary value of email forensics extends beyond criminal prosecutions to encompass civil litigation, employment disputes or intellectual property cases. For this reason, in each context, adherence to forensically sound collection and preservation practices proves essential to ensure evidence withstands admissibility challenges and contributes meaningfully to case resolution.

Chapter 4

Legal Responsibility of Domain Owners

The use of email as a critical infrastructure component for business communications has created a complex legal framework governing domain owner responsibilities regarding email security. As spoofing attacks demonstrate increasing sophistication and financial impact, legal systems worldwide are grappling with questions of liability allocation, duty of care standards, and the evidentiary role of authentication records. This chapter examines the regulatory landscape, establishing legal responsibilities, case law trends, and practical compliance frameworks that define the contemporary legal environment surrounding email spoofing prevention.

4.1 Regulatory frameworks

4.1.1 GDPR

The General Data Protection Regulation (GDPR) establishes comprehensive data protection obligations that encompass email communications security, creating duties for domain owners to implement appropriate technical and organizational measures. The "accountability" principle, specified in Article 5(2), requires the implementation and demonstration of data protection "by design" and "by default". Encryption and pseudonymization are explicitly cited as technical measures organizations can employ to minimize potential damage in the event of unauthorized access [28].

GDPR does not explicitly mandate specific email protocols such as DMARC, but instead establishes an implicit protection to protect personal data transmitted via email. Article 32 [29] specifies that if spoofing leads to a personal-data breach (e.g., staff fooled, credentials stolen, systems compromise), regulators may judge that failing to adopt widely recommended controls (like DMARC/SPF/DKIM, monitoring of DMARC reports, etc.) fell short of "appropriate security", exposing to regulatory action and damages claims.

The GDPR establishes a specific regulatory framework for data protection, but liability arising from spoofing-related incidents cannot be reduced exclusively to data protection law. The legal foundation of responsibility must be analyzed within the broader

framework of civil liability, both extracontractual (tort-based) and, where applicable, contractual. Under Article 82 [30], "*any person who has suffered material or non-material damage as a result of an infringement of the Regulation has the right to receive compensation from the controller or processor*". However, this article operates within a structured liability model where the claimant must demonstrate: an infringement of the GDPR, damage, and a causal link between infringement and damage. The controller may avoid liability if it proves that it is not in any way responsible for the event giving rise to the damage. Accordingly, the key legal question is not merely whether DMARC was implemented, but whether the domain owner complied with the standard of reasonable technical diligence applicable at the time of the incident.

A critical issue concerns whether failure to configure DMARC constitutes negligence per se (violation of established technical rules) or remains within the domain of discretionary technical judgment. At the moment, DMARC is standardized by RFC 7489 [2], recommended by cybersecurity agencies (e.g., ENISA), required in certain national frameworks (e.g., ACN Email Authentication Framework), but it is not universally mandated by statute in all contexts. Therefore, its legal qualification depends on the sector involved, the size of the organization, the foreseeability of spoofing risk and the availability and proportionality of the implementation. The failure to implement DMARC may increasingly be interpreted as a deviation from industry standards in sectors where email fraud risk is well-known (e.g., finance, public administration). Nevertheless, outside regulated sectors, it may still be evaluated under the flexible standard of "reasonable technical discretion."

This nuanced approach avoids overstating DMARC as an automatic minimum legal requirement while recognizing its growing normative weight.

4.1.2 ePrivacy

The ePrivacy Directive (2002/58/EC) and Regulation [31] establishes specific requirements for security of electronic communications, complementing GDPR requirements with additional protections for communications confidentiality. This directive imposes an obligation on companies that utilise electronic means of communication to implement security measures in order to protect personal data against accidental loss, destruction or damage, with a view to preventing the tampering of such communications by malicious users. ePrivacy Directive prohibites the use of false identities in electronic communications, establishing a legal basis for requiring accurate senders identification.

Furthermore, the Directive requires companies to obtain explicit consent before sending marketing communications via email and mandate implementation of unsubscribe mechanisms. These security measures must be proportionate to the risk of unauthorized access; thus, the responsibility for determining the proportionality falls on the domain owner. This approach creates a legal framework where domain owners must assess their email communication risk profiles and implement authentication measures accordingly. The combination of privacy protections and authentication requirements effectively transforms email domain ownership into a responsibility requiring active security management rather than passive acceptance of default mail server configurations.

4.1.3 Email Authentication Framework (ACN)

The Italian Agenzia per la Cybersicurezza Nazionale published a comprehensive Email Authentication Framework establishing specific requirements for email security, specifying the mandatory implementation of SPF, DKIM and DMARC. The ACN framework emphasizes that proper configuration of these protocols drastically reduces the likelihood of successful fraudulent email delivery. Furthermore, the framework positions email authentication as a prerequisite for risk reduction across all other email security domains, which means that, without proper authentication infrastructure, organizations cannot effectively implement layered security approaches. [32]

4.1.4 NIS2

The NIS2 Directive [33] is a cybersecurity framework aimed at strengthening the security standards in EU countries and organizations, thus providing a high level of security for networks and information systems. This Directive introduces cybersecurity obligations for specific categories of entities classified as: **Essential** entities and **Important** entities. First are typically large enterprises operating in highly critical sectors whose disruption could cause severe cascading effects. Sectors include energy, healthcare, banking, transport, drinking/waste water, and core digital infrastructure. The latter generally encompass medium-sized enterprises in the highly critical sectors, as well as medium and large enterprises in other critical sectors such as postal services, waste management, chemical manufacturing and food production and distribution. There are Entities falling within the Directive's scope and Organizations outside its scope.

For essential and important entities, email authentication controls such as SPF, DKIM, and DMARC can potentially be regarded as risk management measures, especially where the risk of spoofing impacts service continuity or public trust. Failure to implement these controls may potentially be regarded as non-compliance with Article 21 risk management obligations.

Article 21 [34] requires these entities to implement "*appropriate and proportionate technical, operational and organisational measures*" to manage risks to network and information systems. Risk management measures must be implemented by companies, that need also to report incidents and be compliant with cybersecurity standards. NIS2 represents the most explicit regulatory framework, mandating email security measures using specific protocols. Since NIS2 requires the implementation of security measures by design and by default, email authentication protocols (SPF/DKIM/DMARC) are a great example of security-by-design controls for communications, for this reason the implementation of DMARC can be seen as a proportionate technical measure mandated by NIS2. In particular, companies that send more than 5,000 emails per day will have to implement these protocols to maintain email deliverability and demonstrate regulatory compliance. Failure to implement these measures constitutes a breach of Article 21 risk management obligations, exposing organizations to substantial financial penalties (which go up to €7,000,000 or 1.41% of the annual global revenue for Important Entities and €10,000,000 or 2% of the global yearly revenue for Essential Entities) and management liability. Importantly, the Directive does not prescribe specific protocols. It

mandates outcome-oriented security measures. For this reason, DMARC cannot be considered legally mandatory per se under NIS2, but may represent a proportionate measure depending on the risk profile of the entity.

4.2 Duty of care: What is reasonable DNS/email configuration?

The concept of "duty of care" in the email security context requires analysis of what domain owners should implement, in terms of expected security practices. For email domain owners, industry standards are increasingly coalescing around implementation of SPF, DKIM, and DMARC protocols. The reasonable care standard encompasses not only initial implementation but also ongoing monitoring and maintenance of authentication records. Domain owners must regularly audit their SPF records to ensure that they accurately reflect authorized mail sources, rotate DKIM keys according to industry practices (typically annually), and review DMARC reports to identify unauthorized senders or configuration errors. Failure to maintain these records in response to organizational changes-such as the adoption of new email service providers, business acquisitions, or infrastructure migrations-constitutes a breach of reasonable care obligations.

4.3 Case law: Liability in email fraud (Man in the Mail)

Litigation regarding email compromise and fraudulent financial transfers provides a concrete illustration of how courts allocate liability and what evidence they examine when establishing negligence in email infrastructure.

A case of December 18, 2024 showed how liability is managed in Germany, within the GDPR rules. A contractor (plaintiff) performed work for a private customer (defendant) and later sent the final invoice by email. However, the email was intercepted and manipulated by unknown third parties, who altered the bank account details on the invoice before forwarding it to the defendant. Believing the invoice to be legitimate, the defendant transferred €15,385.78 to the fraudulent account. The fraud was discovered only after the plaintiff noticed the missing payment. The plaintiff demanded that the defendant pay the amount again, arguing that the debt had not been fulfilled under Section 362(1) of the German Civil Code (BGB), since the payment never reached his account.

The defendant, however, contended that she had already fulfilled her obligation by paying the invoice as received. Furthermore, she claimed that the plaintiff had failed to implement adequate security measures when sending bank details by unprotected email, which enabled the fraud. The defendant invoked Article 82 of the GDPR, alleging that the plaintiff's negligent handling of personal data led to the incident. [35]

When the email account of the sender of funds is compromised and fraudulent payment instructions is issued, courts examine whether the recipient exercised reasonable diligence in verifying payment instructions. The case of "Jane Group Limited v. Heritage Gas Limited" [36] established that recipients receiving apparently legitimate payment instructions from known business partners generally must pay per those instructions, even if they are originated from compromised email accounts, absent evidence that recipients

should have suspected fraud. However, courts will examine whether recipients ignored unusual payment circumstances, such as changes to established banking relationships or unusual urgency requests that deviate from normal business patterns.

4.4 Forensic evidence as leverage: Linking DMARC records and responsibility

The preservation of digital evidence and the analysis of email authentication records have become fundamental to establish liability in legal proceedings. Particular emphasis has been placed on the use of DMARC records as objective indicators of organizational security posture.

Maintaining DMARC records can serve as a form of evidence regarding the company's or victim's security intentions and implementation choices. When a domain maintained a DMARC policy of "*p=reject*" (mandatory rejection of unauthenticated emails), the domain owner can present forensic evidence that any email impersonating their domain would have been rejected by properly configured receiving mail servers. The strength of this evidence is twofold: firstly, it is objective and therefore not subject to expert interpretation; secondly, it is demonstrable without requiring access to the compromised email account or the fraudster's infrastructure.

On the other hand, when a domain operated with a DMARC policy of "*p=none*" (monitoring without enforcement) at the time of the disputed transaction, the domain owner effectively demonstrated that they prioritized email deliverability over security and accepted the risk that spoofed emails might be delivered.

Courts interpret this choice as evidence that the domain owner was aware of authentication capabilities but chose not to implement them, constituting a form of negligence even if the specific fraud attack was sophisticated.

In addition, the process of deriving DNS records through web forensics techniques facilitates the documentation of DMARC, SPF, and DKIM configuration states at particular historical points in time by enabling the retrieval of DNS historical databases, certificate transparency logs, and DMARC report collections. While this forensic capability is not universally available for all historical periods, it can establish what defensive measures were technically available at specific times and whether domain owners had knowledge of these measures through receipt of DMARC reports or other sources.

Finally, courts examining DMARC evidence are required to address questions regarding the probative value of the evidence in relation to its prejudicial effect. While a domain maintaining "*p=none*" policies may be probative evidence of inadequate security, courts must assess whether such evidence unfairly prejudices the defendant by suggesting negligence without adequate consideration of industry standards at the specific time period or the sophistication of the attack employed.

4.4.1 DMARC policy and defensive positioning

Looking at defensive positioning closer, the evidentiary value of DMARC records must be analyzed within the broader framework of civil procedure and burden of proof allocation.

In general civil litigation, the burden of proof lies initially with the claimant. The injured party must, in fact, demonstrate: the occurrence of damage, the defendant's unlawful conduct or breach of duty, and the causation. In this context, DMARC records serve not as automatic proof of negligence or diligence, but as objective technical evidence of security posture at a given time.

A domain that operates with `p=reject` must demosntrate that:

- security enforcement mechanism were active;
- receiving servers were properly configured and for this reason would have rejected misaligned messages;
- spoofing success required bypassing external receivers enforcement or exploiting different vectors.

On the other hand, if a domain has a "`p=none`" policy, this indicates that there is some form of authentication awareness but also shows there is no attempt to stop mail. But the only presence of "`p=none`" by itself is not sufficient to establish negligence. The courts will consider: industry standards at the time, technical and practical limits, proportionality, and whether the risk analysis justified a temporary monitoring mode. Therefore, the DMARC policy is just part of the overall picture in determining negligence.

4.4.2 Accountability and proof of diligence

In GDPR, Article 5(2) [37] states that controllers must be able to demonstrate compliance. This shifts practical emphasis toward documentation and monitoring. During disputes the domain owner may need to demonstrate the existence of SPF/DKIM/DMARC configuration, a progression toward enforcement, monitoring of DMARC reports (RUA ingestion) with related documented audits and corrective actions. Any failure to maintain logs or historical DNS records may weaken the defensive position, as the inability to demonstrate security posture can be interpreted unfavorably under accountability logic. For this reason, the probative value of Authentication-Results headers, DNS record snapshots, DMARC aggregate reports and SIEM logs, lies not only in proving technical facts, but in evidencing structured governance.

4.5 Mitigation strategies and defensible compliance practices

The foundational mitigation strategy involves the implementation of comprehensive email authentication aligned with the recommendations of ACN Framework and industry best practices. This encompasses:

- **SPF configuration**: publishing accurate SPF records listing all systems authorized to send emails on the domain's behalf, including mail servers, third-party marketing platforms, and any acquired companies' mail infrastructure. Regular audits ensure that SPF records remain accurate.

- **DKIM configuration and Key Management**: companies should implement DKIM signing for all outbound email; key rotation should also occur at least annually and immediately upon suspicion of key compromise. Documentation of key generation, rotation, and retirement procedures demonstrates defensible key management practices in case of litigation.

- **DMARC Policy Progression**: organizations should implement DMARC using the graduated progression recommended by ACN. Starting with policy "*p=none*" to collect authentication data and identify legitimate senders; then analysis of DMARC records useful to identify and remediate false positives. After this, the policy should be set to "*p=quarantine*" in order to redirect unauthenticated emails to spam folders. Finally, "*p=reject*" policy should be implemented, reflecting organizational risk tolerance and business impact assessment.

This graduated approach acknowledges the operational realities that organizations face while establishing that these organizations have explicit intentions to achieve enforcement-level policies rather than indefinitely maintaining permissive configurations.

4.6 Legal Defensibility through Technical Controls

Figure 4.1 depicts the relationship between legal responsibility, technical email authentication mechanisms, monitoring practices, and evidentiary readiness in the context of disputes arising from email spoofing.

The diagram represents the conceptual framework of the incremental chain of events leading to the formulation of legal responsibility, technical email authentication mechanisms, and ultimately to the formulation of defensible evidence. In the event of an invoice fraud case involving email spoofing, the organization's position relative to its duty of care is evaluated as prescribed by relevant regulatory requirements. In the European Union, the relevant regulations include Article 32 of the GDPR and Article 21 of the NIS2 Directive, which require the adoption of "appropriate technical and organizational measures" to be evaluated in the context of the state of the art and the specific risk profile. [29] [33]

Domain authentication mechanisms, including SPF, DKIM, and DMARC, form the foundation of technical email authentication mechanisms. However, the adoption of these mechanisms is only the beginning. Policy enforcement mechanisms involving an incremental chain of events leading to the formulation of the relevant policy (from `p=none` to `p=quarantine` and ultimately to `p=reject`) signify the maturity of the organization.

Most importantly, the diagram points out the need to monitor the preventive controls. This can be done through the ingestion of the aggregate reports, alerting mechanisms of anomalous senders, change management procedures, and auditing. This indicates that authentication has to be an ongoing process rather than just a static process.

The third column of the diagram points out the evidentiary requirements. This indicates that in case of any legal disputes, the organization has to be able to provide concrete evidence. This can be done through the provision of email headers with the authentication results, DNS configurations showing the enforcement of the policy at a particular time, and logs of the system.

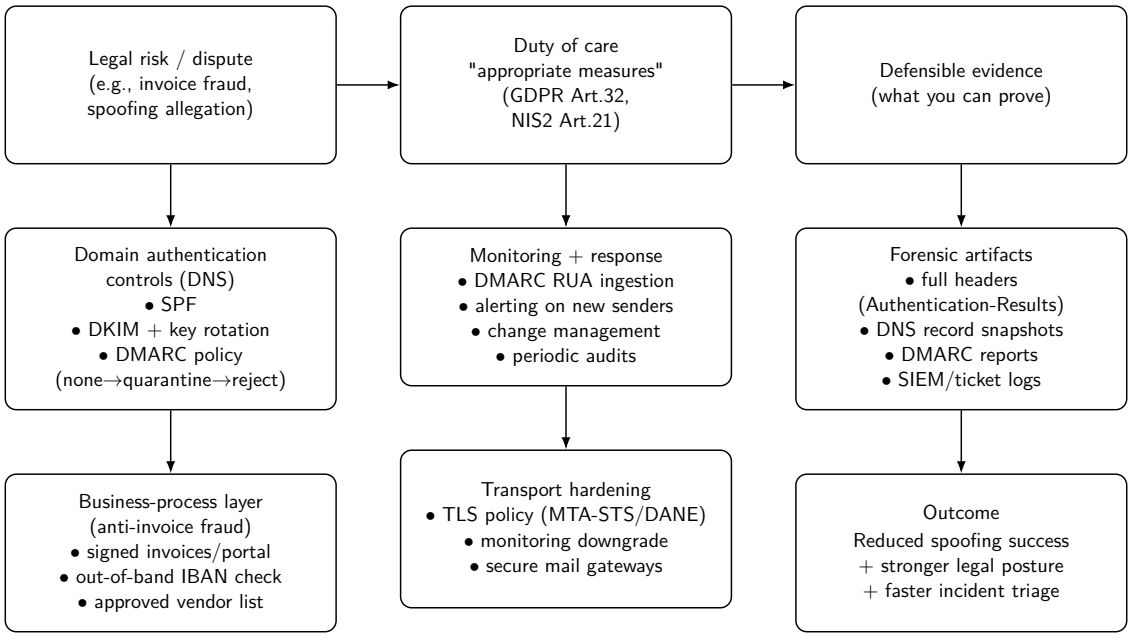

Figure 4.1: Legal responsibility, email authentication controls, monitoring, and defensible forensic evidence.

Finally, the bottom part of the diagram indicates that authentication of emails is just part of the overall security of the organization. This can be done through business process controls and transport hardening.

Ultimately, the diagram reveals that the legal defensibility of an organization in the event of email spoofing can be achieved through the integration of authentication, monitoring, governance, and evidentiary requirements.

Chapter 5

Insights of Existing DMARC Checker and Forensics Tools

The growing use of Domain-based Message Authentication, Reporting, and Conformance (DMARC) has resulted in a large number of new tools that purportedly aid domain owners in configuring, monitoring, and analyzing email authentication technologies. The tools vary in functionality, ranging from simple compliance checkers to sophisticated forensic applications that monitor, ingest, and interpret DMARC feedback reports. In this chapter, there is a technical evaluation of some available DMARC checker and forensics software, covering methodology, features, forensic capabilities, and inherent weaknesses.

5.1 Survey of DMARC compliance checkers

The primary goal of DMARC compliance checkers is to check for the correctness of DNS record authentication and gain insight into what occurs when a mail purported to have come from a certain domain is received by a mail server. Despite the variety that exists among these tools and their level of functionality, they are collectively meant to eliminate any resulting confusion with regard to domain abuse.

5.1.1 DMARC Forensics

DMARC Forensics, developed by Professor Paolo Dal Checco, is a web-based application intended to provide evaluation of DMARC compliance and spoofing risks by linking DNS-layer security controls to real-world email abusive conditions. Unlike dashboard-style applications, DMARC Forensics is based on on-demand evaluation instead of relying on previously ingested DMARC aggregate reports.

It is a configuration-based DMARC compliance and spoofing exposure analysis tool, designed to infer domain-level spoofability from DNS-published authentication policies, rather than from observed email traffic. The tool perform a series of direct DNS queries, including: `TXT` lookup for `_dmarc.<domain>`, SPF discovery via TXT lookup on the organizational domain and enumeration of DKIM selectors when inferable from known

patterns or published records. The tool does not perform interaction or mail traffic inspection, since the analysis is entirely configuration-driven. [12]

DMARC Forensics parses and evaluates:

- DMARC policy tags (`p`, `sp`, `pct`);
- Alignment modes (`adkim`, `aspf`);
- Reporting URIs (`rua`, `ruf`);
- Presence and strictness of SPF and DKIM policies
- Existence of policy contradictions (e.g., `p=none` with strict alignment)

The tool infers spoofing susceptibility demonstrateability by:

- determining whether the domain publishes an enforceable DMARC policy,
- identifying missing or permissive authentication layers,
- assessing whether a forged message would be rejected, quarantined, or accepted by a DMARC-compliant receiver.

Furhtermore, the tool does not simulate email delivery nor rely on historical abuse telemetry.

Methodology and findings

DMARC Forensics tool utilizes a methodology which is designed to be aligned specifically with the requirements of a forensic or research approach rather than automation. Such an approach is defined by:

1. Protocol faithful interpretation of DMARC aggregate reports, while preserving the semantic meaning of the SPF, DKIM, and alignment results as defined in RFC 7489.
2. Correlation between DNS state and behavior, allowing the investigator to determine whether authentication failures result from misconfiguration, third-party senders, or actual abuse.
3. Emphasis on evidentiary value to reconstruct history for domain posture and allow for conclusions regarding the maturity of law enforcement.

From the empirical research carried out using the tool, the following facts are noted: a substantial share of those websites employs DMARC in monitoring mode (p=none) for significant periods of time, making the preventive capability of DMARC less potent while still useful for telemetry. As noted throughout the argument developed within Chapter 4 of this treatise, perhaps the greatest strength of DMARC is not merely with disposition decisions but also with providing objective, verifiable proof of abuse attempts and organizational intent.

5.1.2 PowerDMARC

PowerDMARC is a commercial SaaS product, which means it includes a complete system for monitoring and managing DMARC policies. From a technological standpoint, PowerDMARC represents a simplified approach, more focusing on usability as opposed to the complexity of the protocol stack itself, particularly in an enterprise environment.

PowerDMARC is a "full-stack" email authentication service, which means it includes not only reporting but also a managed infrastructure to overcome the inherent technical limitations of the DNS and the SPF protocol.

The service is a DMARC monitoring platform based on real-time ingestion and analysis of DMARC aggregate reports (RUA) generated by receiving mail servers.[38]

PowerDMARC relies primarily on:

- SMTP-delivered XML aggregate reports sent to the rua address;
- Optional DNS queries to contextualize authentication results;
- Optional enrichment with ASN, geolocation, and reputation data.

The platform continuously parses incoming XML files and normalizes the data into an internal schema.

Analysis logic

For each reporting interval and sending source, PowerDMARC evaluates:

- Source IP address
- SPF result and domain
- DKIM result and selector
- DMARC alignment outcome
- Policy disposition applied by the receiver

These parameters are then aggregated across time windows to produce trend analysis and anomaly detection.

The core logic of the tool focuses on: the identification of unatuhroized senders based on SPF/DKIM failures, the detection of configuration drift across third-party services and the support of a safe progression from `p=none` to more enforced states. From a technical standpoint, the platform acts as a large-scale parser and correlator of DMARC feedback, rather than a protocol validator. In spite of its many features, there are some limitations to PowerDMARC, which include its dependence on receiver participation and reporting, as well as a lack of insight into attempts to spoof email blocked before DMARC evaluation.

5.1.3 MxToolbox

MXToolbox represents a lightweight, widely used diagnostic tool that focuses on real-time inspection of DNS records, including SPF, DKIM, and DMARC. Its DMARC checker validates record syntax, policy presence, and alignment indicators but does not perform longitudinal analysis or forensic correlation. As such, it is best suited for configuration verification rather than abuse investigation.

The tool performs DNS TXT lookups for SPF, DKIM, and DMARC; syntax validation of records and basic interpretation of policy directives. No historical data, reporting ingestion, or traffic observation is involved.

MxToolbox has transitioned from a suite of standalone diagnostic tools into a comprehensive "Delivery Center" designed for end-to-end deliverability management. Its technical methodology emphasizes real-time monitoring and proactive alerting based on the health of the entire email ecosystem. A component of the Delivery Center is the Adaptive Blacklist Monitoring system, which uses the telemetry extracted from DMARC RUA reports to dynamically identify every IP address sending mail on behalf of the domain. Once an IP is detected in a report, the system automatically checks it against hundreds of DNS-based Blackhole Lists (DNSBLs). If an IP used by a legitimate service (e.g., a marketing automation platform) becomes blacklisted, MxToolbox generates an emergent alert. This creates a powerful technical synergy: DMARC provides the visibility into who is sending, and the Delivery Center provides the reputation intelligence to ensure those senders remain "clean" and capable of reaching the inbox. Finally, MxToolbox provides rigorous syntax validation engines as well, analyzing the record's structure against relevant RFCs to identify common configuration errors that can lead to deliverability failures (e.g, ensuring that the `v=DMARC1` version tag is present, the policy tag p is correctly defined, and that reporting URIs are properly formatted with the `mailto:` prefix. [39]

5.2 Forensic analysis features

The forensic value of DMARC tools is primarily determined by how they ingest, interpret, and preserve feedback data defined by the DMARC specification.

5.2.1 Aggregate vs Forensic reports

There are two reporting mechanisms defined by the DMARC protocol:Aggregate Reports (RUA) and Forensic Reports (RUF) [2]. Aggregate Reports (RUA) are periodic statistical summaries of the authentication results, categorized by sending IP address, authentication result, and policy decision. Aggregate Reports are privacy-preserving and are the core of most DMARC reporting tools.

Forensic Reports (RUF) offer near-real-time feedback for individual authentication failures, although due to privacy and regulatory issues, RUF is currently deprecated and rarely implemented. The PowerDMARC and DMARC Forensics tools mainly rely on Aggregate Reports due to current best practice and regulatory constraints.

5.2.2 Export, Logging and Evidence Preservation

From a forensic point of view, the ability to export raw data and retain historical records is essential. DMARC Forensics allows direct access to parsed aggregate data and places strong emphasis on the interpretability of individual report fields (such as source IP, SPF/DKIM result, and alignment mode). Commercial solutions usually include export capabilities in CSV and/or JSON format and often include integration with SIEM systems to retain data over time.

5.2.3 Multi-domain support

In an enterprise environment, managing multiple domains and subdomains may be necessary, including the possibility of delegating DMARC reporting addresses between domains. Advanced solutions include:

- Aggregation from multiple domains
- Subdomain policy analysis
- Reporting authorization

This is a critical requirement to properly evaluate abuse within complex email infrastructures.

5.3 Privacy, usability, and limitations

The usefulness of the tools based on the DMARC analysis process is marred by the inherent limitations of the process, which are linked to the DMARC protocol. The aggregate reports do not include message content, subject, or the identifier of the message recipient. Although the decision not to include the message content, subject, or the identifier of the message recipient is in line with the privacy by design approach, it affects the ability of the system to perform forensic attribution of infrastructure.

The commercial tools, on the other hand, have been designed to offer ease of use, whereas the tools based on the DMARC analysis process, such as DMARC Forensics, offer access to raw or lightly processed data, which requires high expertise on the part of the user.

Some limitations of DMARC-based tools are present, they indeed cannot:

- detect attacks from compromised legitimate sender accounts,
- prevent or identify content manipulation such as invoice tampering,
- attribute malicious intent beyond authentication failures.

The use of the DMARC forensics process should, therefore, be complemented by the use of additional security controls such as the DKIM key management, transport security, as well as business process controls, as discussed in the foregoing chapters.

Chapter 6

DMARC Forensics Tool Technical Analysis

DMARC Forensics is implemented as a single-page client-side web application that performs on-demand DNS interrogation and evaluates DMARC/SPF/DKIM compliance locally in the browser. The application does not require authentication, does not transmit uploaded domain lists to an application backend, and does not ingest email traffic or DMARC aggregate (RUA) telemetry; instead, it derives conclusions from real-time DNS resolution and deterministic parsing logic.

In the server-side model proposed by other tools, the vendor acts as an intermediary, possessing full knowledge of the investigator's targets. In this tool, a client-side model is used, where the vendor is removed from the loop entirely. The browser acts as the direct agent of the investigator, interacting with the DNS infrastructure without passing data through the tool developer's systems.

6.1 System architecture and workflow

From an architectural standpoint, the tool follows a browser-executed execution model in which DNS resolution is performed via DNS-over-HTTPS (DoH) using the Google Public DNS JSON API endpoint:

```
https://dns.google/resolve?name=<QNAME>&type=<RRType>
```

In this way, the correctness of the analysis depends on the resolver's view of DNS and on the integrity of the HTTPS communication channel between the client and the resolver. The privacy implications of this design are also explicitly acknowledged in the application's policy, as DNS queries are transmitted to Google's infrastructure and are subject to its data handling practices.

The operational workflow can be summarized as:

1. User input acquisition and normalization (domain list, optional DKIM selector).
2. Domain syntax validation.

3. Parallel DNS resolution for SPF, DMARC, and DKIM-related records.
4. Deterministic parsing and compliance evaluation.
5. Report generation and optional export of structured results and raw query logs.

Notably, the tool performs no SMTP-level simulation, no DKIM signature validation, and no analysis of actual message flows. Its conclusions are derived exclusively from DNS-published authentication policies.

6.2 Functionalities

6.2.1 Client-side DNS quesrying

All DNS resolution logic is performed entirely within the browser through asynchronous HTTP requests to the Google Public DNS DoH JSON service. TXT and CNAME records are dynamically retrieved. For the evaluation of DMARC records, the tool first attempts to resolve the record `_dmarc.<domain>` as a CNAME record. If the CNAME record is present, the process will follow the result and subsequently resolve the TXT record of the effective name.

The function `analyzeDmarc` has been designed to execute a specific test for CNAME Redirection. The function begins by querying `_dmarc.[domain]` with a query type of "CNAME". The function then proceeds to follow the target to retrieve a response, provided a response is found (i.e., DNS type is 5). The function is vital in auditing domains that delegate DMARC authority to a third-party service provider. It appears that the analysis process is limited to a single-level CNAME follow, without any explicit inclusion of a redirection chain. It is, therefore, evident that DNSSEC authentication is left to the external DNS resolver service.

DKIM evaluation is implemented through heuristic selector enumeration. Recognizing that DKIM records are published at unpredictable selectors (e.g., `selector._domainkey`), the tool implements a brute-force discovery algorithm. The source code contains a hard-coded dictionary of common selectors: (e,g. ['default', 'mail', 'dkim', etc...]). The tool then iterates through this list, generating speculative queries to identify active keys without requiring prior knowledge of the selector. Finally, this process confirms selector existence but cryptographic key length is not validated, the `"p="` parameter is not inspected and signature verification is not performed. On the other hand, SPF analysis is limited to record discovery and basic structural assessment.

6.2.2 Record parsing and validation

Once TXT records are retrieved, the tool parses them according to a tag-based tokenization process. The tool checks if `v=DMARC1` is present, checks validation of policy values (none, quarantine, reject), parses the optional tags (such as `sp`, `pct`, `aspf`, `adkim`, etc.) and then detects if multiple DMARC TXT records or malformed tag structures are present. As a result, domains using experimental or extension tags may be incorrectly classified as non-compliant, representing a methodological strictness that should be explicitly acknowledged in technical evaluation.

6.2.3 External reporting authorization verification

The tool implements verification logic for external DMARC reporting destinations. When a `rua` or `ruf` address references a domain different from the analyzed domain, the tool checks for the presence of a TXT record at:

`<requesting-domain>._report._dmarc.<report-domain>`

If a corresponding "`v=DMARC1`" TXT record is present, it will be interpreted as valid delegation. If the record is missing, it flags the error: "External reporting to [domain] is not authorized."

6.3 Comparative evaluation with other tools

When compared to other tools in the DMARC ecosystem, DMARC Forensics occupies a distinct operational niche. Unlike enterprise monitoring platforms that ingest DMARC aggregate reports (RUA) and correlate authentication outcomes over time, DMARC Forensics performs no telemetry ingestion. Its analysis is entirely static and configuration-based.

While other tools (such as "dmarcian") excels at visualizing XML aggregate reports over time, DMARC Forensics is engineered for infrastructure auditing. Its use of "dns.google" provides a "neutral ground" truth, free from the caching artifacts that often plague server-side tools. The client-side execution ensures that the "observer effect" is minimized; the investigator does not leave a footprint on a third-party forensic vendor's logs.

Compared to lightweight DNS diagnostic services, it provides deeper protocol interpretation, including:

- CNAME-following logic for DMARC records,
- External reporting authorization checks,
- Alignment mode assessment,
- Structured compliance classification.

Even if it lacks other capabilities, such as abuse detection visibility, source IP telemetry and SIEM integration capabilities. The tool is therefore most appropriate for point-in-time configuration audits, compliance demonstrations, and educational or investigative contexts where deterministic DNS state evaluation is sufficient.

6.4 User experience: strengths and weaknesses

From a usability and security engineering perspective, several design characteristics deserve to be mentioned.

Among the pros there are:

- **Forensic Transparency**: tool provides raw DNS Response (JSON format) in the logs. This transparency is vital for explaining findings in court (e.g., proving a record existed but was malformed).

- **Speed**: the use of `Promise.all` allows for checking 20 domains in nearly the same time as checking one, limited only by the browser's maximum concurrent connection limit to dns.google.
- **Heuristic Reconnaissance**: the automated DKIM selector search effectively "guesses" hidden configurations, a feature often absent in free tiers of commercial tools.
- Secure DOM manipulation using `textContent`, reducing XSS exposure risk.
- Structured export capabilities including XLS and detailed TXT logs.
- Transparent display of actionable remediation guidance.

On the other hand, there are some limitations as well:

- **Volatile State**: with a client-side SPA, refreshing the page clears all evidence. There is no persistent database.
- **Rate Limiting**: there is a hard dependency on a single external resolver (Google public DNS), thus an heavy usage (e.g., checking hundreds of domains) may trigger Google's API rate limits (`HTTP 429`), as the requests originate from the user's IP.
- Strict classification model that may over-report non-compliance.
- **Absence of full SPF and DKIM semantic evaluation**.
- **No Historical Data**: The tool captures the state now, so it cannot prove what the record was yesterday (unlike Passive DNS databases).

6.5 Integration of forensic evidence

One of the most technically significant aspects of the tool is its evidentiary potential. The application records: a timestamp for each analysis, the exact DoH query URL used for resolution and the raw JSON block returned by the resolver. The code function `downloadTxtFile` generates a file named `dmarc-log-$timestamp.txt`. The log does not merely save the final table, it instead appends the raw transaction data during the analysis loop (`textLog+=...`). The timestamp follows an ISO 8601 format (e.g., `2023-10-27T10:00:00.000Z`). Queries integrity are kept by recording the exact URL queried (e.g., `https://dns.google/resolve?name=_dmarc.example.com...`). The full JSON response from Google is recorded, including the `Answer` array, `TTL` (Time To Live), and `Status` codes.

These elements can be exported and preserved. This enables what may be termed configuration crystallization: the ability to capture a reproducible snapshot of a domain's authentication posture at a specific point in time.

As previously mentioned, digital evidence must be acquired in a manner that ensures its integrity and repeatability. The concept of "crystallization" refers to freezing the digital scene to prevent alteration. Such crystallization can support the demonstration of

enforcement level (e.g., p=reject vs p=none), the verification of reporting configuration and a proof of DNS state at analysis time.

In conclusion, the DMARC Forensics tool should be formally classified as:

A browser-executed, DNS-based DMARC compliance and spoofing exposure analyzer that performs deterministic policy evaluation using DNS-over-HTTPS queries, without ingesting runtime email authentication telemetry.

Its methodological strengths lie in transparency, configuration strictness, and evidentiary export capability. Its limitations derive from its stateless design and its exclusive reliance on DNS-published policy data.

Chapter 7

Impersonation Risk Assessment Tool

7.1 Motivation and Scope

While previous chapters have analyzed the theoretical underpinnings of email authentication protocols (SPF, DKIM, DMARC) and the legal frameworks governing their adoption, practical risk assessment often remains fragmented. Organizations typically employ disparate tools for DNS compliance checking and separate services for brand protection or typosquatting detection. The tool developed in this thesis aims to bridge this gap by providing a unified, automated engine that correlates domain syntax anomalies with email authentication posture.

The scope of the tool is to perform a "black-box" assessment of a target domain or FQDN (Fully Qualified Domain Name). Unlike internal audit tools that require privileged access to mail servers, this system operates from an external perspective, simulating the reconnaissance phase of an attacker or the due diligence phase of a security analyst. It evaluates two primary vector categories: deceptive syntax (homoglyphs, typosquatting, and misleading subdomains) and permissive authentication (weak or missing DMARC/SPF policies), combining them into a single quantifiable risk score.

7.2 Threat Model and Design Assumptions

The system is designed based on a particular threat model, in which the attacker's goal is to pose as a legitimate brand or entity to perform BEC or credential phishing attacks. The model assumes three specific attack types:

1. **Lookalike Domains**: The adversary registers a domain name that looks like the legitimate domain name of the targeted entity, such as `"examp1e.com"` instead of `"example.com"`, or they use Internationalized Domain Names (IDN) to perform homoglyph attacks.

2. **Subdomain Deception**: The attacker uses a legitimate, non-brand domain but crafts a misleading subdomain (e.g., `paypal.support.malicious.com`).

3. **Spoofing Capability**: The adversary attempts to send emails directly from the target domain, exploiting the absence of strict DMARC enforcement (p=none or missing records).

The system assumes that the evaluator has no internal knowledge of the targeted entity's infrastructure. Consequently, the tool relies exclusively on public DNS records and algorithmic string analysis. It is assumed that while the presence of DKIM selectors cannot be discovered via DNS queries, the presence of common default selectors can be used as an indicator of the presence of email infrastructure.

Typical operational goals include:

- User deception via visually similar domains (typosquatting, homoglyphs).
- Message delivery improvement through partial sender-authentication configuration (e.g., SPF softfail, DMARC policy p=none).
- Brand-trust transference by embedding brand tokens in subdomains (e.g., paypal.com.attacker.tld), exploiting the fact that many users inspect only a prefix or the leftmost labels.

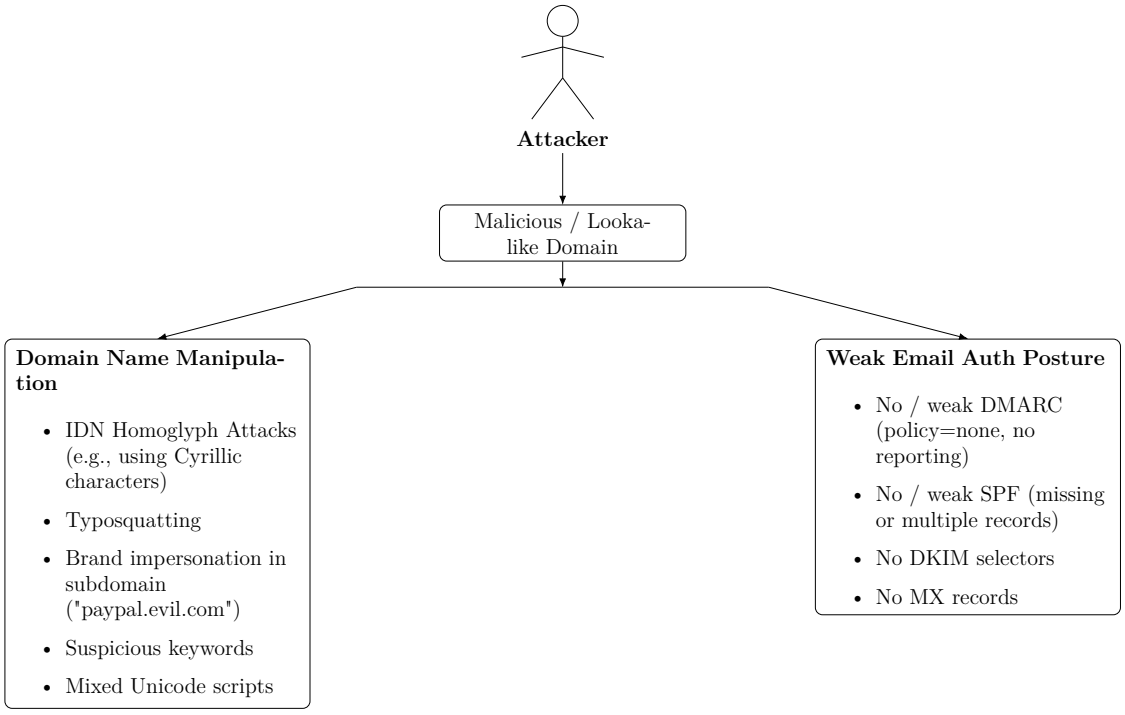

Figure 7.1: Threat model

Regarding the desing assumptions:

- DNS is the authoritative observation plane. All checks are based on DNS responses obtained via a recursive resolver.

- Static analysis, not traffic verification. The tool does not verify that mail is actually signed with DKIM or that SPF authorization succeeds for a specific sending IP. Instead, it uses conservative heuristics (e.g., selector probing) to estimate likely posture.
- eTLD+1 as the stable unit of analysis. The system collapses an input FQDN/URL into a registrable domain (eTLD+1) to standardize posture checks and scoring.
- Explainability is a first-class output. The score is accompanied by reason tags to support analyst interpretation and to avoid "black-box" risk reporting.

7.3 System Architecture

7.3.1 High-Level Architecture

The tool is a modular Python application with a central controller. It is based on a producer-consumer pattern: a list of target domains comes in, gets processed by several parallel threads, and finally pluses up into structured reports.

The entry point is defined in `main.py`, which initializes:

- the argparse interface to handle inputs (which could be a single domain or a file with many of them),
- configuration parameters such as custom DNS resolvers and timeout values.

The system also includes other configuration options such as custom resolvers and timeout options. It uses a thread-safe approach when resolving DNS, so even if several threads are running concurrently, there won't be any race conditions and socket exhaustion.

The tool is organized around a simple layered architecture:

1. **Input layer (CLI)**: Parses arguments (`--input, --brands, --dns, --timeout, --workers, --format, --dkim-probe`) and loads domain lists.
2. **Normalization & decomposition**: Converts inputs into normalized domain strings and extracts the registrable domain (eTLD+1) plus optional subdomain components.
3. **Feature extraction**:
   - **Syntax analyzer**: detects IDN/mixed-script/suspicious tokens/typosquatting/brand-in-subdomain.
   - **DNS posture analyzer**: checks MX, DMARC, SPF and optionally probes DKIM selectors.
4. **Scoring & explainability**: Aggregates features into a bounded score [0,100] and assigns a severity tier.
5. **Reporting**: Emits a table (stdout) or machine-readable CSV/JSON.

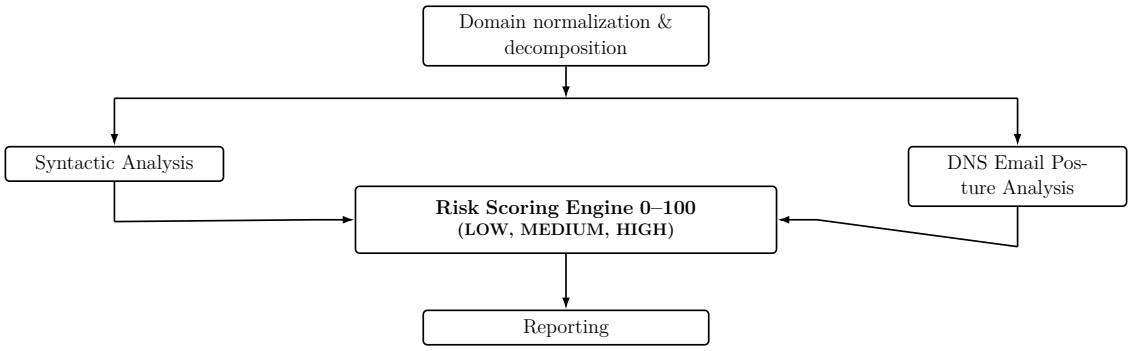

Figure 7.2: High-level control flow architecture

7.3.2 Processing Pipeline

The core processing logic is encapsulated in the `analyze_one` function within `main.py`. For each input domain it executes:

- **DNS Reconnaissance**: The pipeline instantiates a thread-local resolver via `get_thread_resolver` to avoid resolver-state sharing across threads and query MX, TXT (for SPF and DMARC), and specific DKIM records.
- **Syntax Analysis**: The domain is normalized and decomposed into its registrable part (eTLD+1) and subdomains using `analyze_syntax(domain, brands)`. eTLD+1 is extracted through "syn.registrable_domain" and query:
  - **MX**: `query_mx(resolver, etld1)`
  - **DMARC TXT**: `query_txt(resolver, f"_dmarc.etld1")`
  - **SPF TXT**: `query_txt(resolver, etld1)`
  - **DKIM** (optional): `probe_dkim_selectors(resolver, etld1, enabled=dkim_probe)`
- **Posture Analysis**: Raw DNS responses are parsed by dedicated modules to extract tags and semantic policy data.
- **Risk Scoring**: The scoring engine aggregates findings from all previous stages to compute a numeric risk score (0-100) and generate a list of "reason tags."
- **Result Object Construction**: A structured Result dataclass is populated and returned to the main thread for reporting.

This pipeline design decouples data acquisition from analysis, allowing for independent extensibility of syntax checks or protocol parsers without refactoring the orchestration logic.

7.4 Domain Syntax Analysis

The syntax layer aims to detect domain-name deception patterns that are independent of authentication posture and frequently precede successful impersonation attempts.

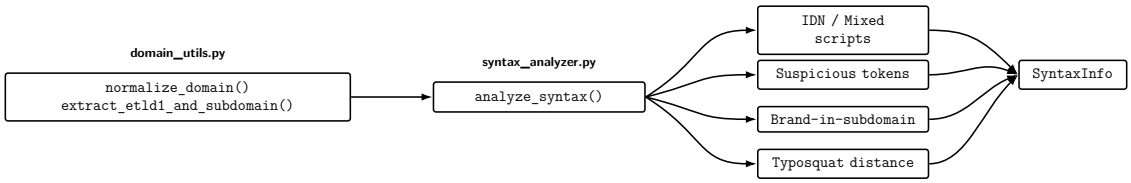

Figure 7.3: Domain Analysis Pipeline

7.4.1 Input Normalization and eTLD+1 Extraction

The tool starts normalizing user input in `normalize_domain()` by stripping schemes (`http://, https://`), paths, ports, and trailing dots, then lowercasing the string. This ensures consistent DNS query names and stable tokenization. To distinguish between the organizational domain and the subdomain, the system employs the `extract_etld1_and_subdomain` function. This utilizes the `tldextract` library to accurately separate the Public Suffix (e.g., `.co.uk, .com`) from the registrable domain. This distinction is critical because DMARC policies are typically inherited from the organizational domain (eTLD+1), whereas syntax-based deception often occurs in the subdomain.

7.4.2 Internationalized Domains and Homoglyph Risk

IDN homograph attacks represent a significant impersonation risk. Internationalized Domain Names (IDNs) enable Unicode labels, which introduces homoglyph/confusable risks when characters from different scripts resemble Latin glyphs. The tool flags IDN risk using `is_idn()`, which detects either Punycode labels (`xn-`) or non-ASCII code points. A `mixed_scripts` boolean flag is then calculated by invoking `detect_mixed_scripts`, which inspects the Unicode character properties of the domain labels. If a label contains characters from multiple incompatible scripts (e.g., Cyrillic and Latin) it is flagged as high-risk, as this is a strong indicator of homoglyph obfuscation intended to deceive users. This operationalizes a practical subset of the Unicode security guidance [40], which explicitly treats mixed-script confusables as a key risk category.

7.4.3 Typosquatting Detection

Typosquatting is modeled as an edit-distance similarity between the analyzed domain label and a set of brand labels provided via `--brands`. The tool computes the distance either using the optional python Levenshtein package (fast path) or a pure-Python Damerau–Levenshtein implementation `damerau_levenshtein_py()`. The tool extracts the second-level label (e.g., paypal from paypal.com) from the analyzed eTLD+1 and for each brand domain, it extracts the corresponding brand label. If $0 < \textit{distance} \leq 2$, the domain is treated as a typo candidate and stored in `typo_matches`, sorted by minimum distance. The scoring layer then assigns a high weight to distance 1 (typical single substitution, insertion or deletion) and a moderate weight to distance 2, reflecting the higher exploitability of close variants in user perception.

7.4.4 Subdomain-Based Deception Signals

Attackers frequently embed sensitive keywords in subdomains to simulate legitimacy, for example `paypal.secure-login.attacker.com` visually foregrounds the target brand in the leftmost labels, while DNS ownership remains with `attacker.com`. The tool analyzes this vector by tokenizing the subdomain string using a regular expression splitter `re.split(r"[â-z0-9]+", ...)`. This procedure detects this only when the input actually contains a subdomain. The resulting tokens are intersected with a predefined `SUSPICIOUS_TOKENS` set, which includes terms like "login," "verify," "billing," and "secure." Finally, if a brand list is provided, the tool checks if any brand name appears explicitly as a token within the subdomain, thus detecting "brand-in-subdomain" attacks, where a trusted brand is used as a prefix on a malicious domain. These kind of matches are scored aggressively because they indicate deliberate impersonation intent rather than incidental similarity.

7.5 DNS and Email Authentication Posture Analysis

The DNS posture analysis focuses on deployability for email-based impersonation and on policy signals that reduce the cost of spoofing campaigns.

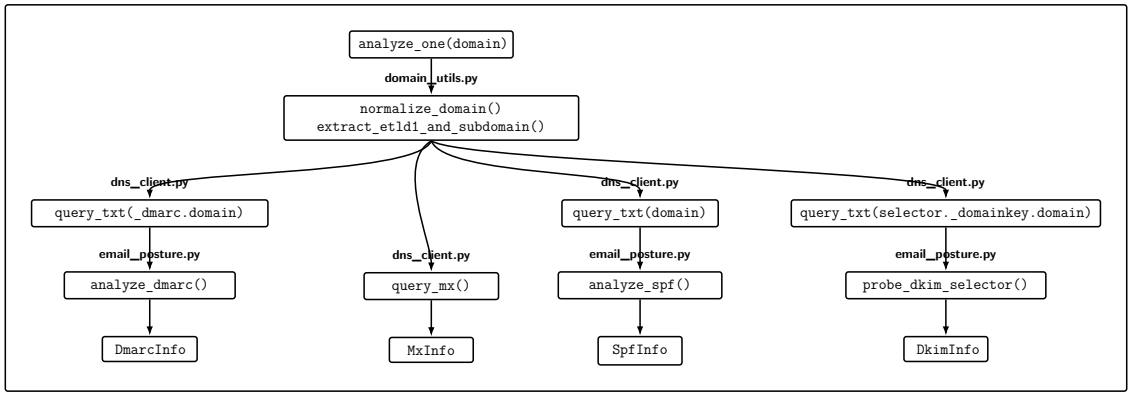

Figure 7.4: Email Posture Analysis Pipeline

7.5.1 MX Record Presence

The presence of Mail Exchange (MX) records is the primary indicator of a domain's capability to receive email. Its presence is queried via `query_mx(resolver, etld1)`. The absence of MX records suggests the domain is not intended for email reception, although it may still be used to send spoofed outbound mail, for this reason a small score increment is assigned when the record is not present to avoid over-interpreting this signal.

7.5.2 DMARC Configuration Analysis

DMARC analysis is handled by `analyze_dmarc` function.

Listing 7.1: DMARC analysis function

```
def analyze_dmarc(txt_records: List[str]) -> DmarcInfo:
    info = DmarcInfo(raw_records=txt_records[:])
    dmarc_recs = [r for r in txt_records if "v=dmarc1" in r.lower()]
    if not dmarc_recs:
        info.present = False
        return info

    info.present = True
    if len(dmarc_recs) > 1:
        info.ambiguous = True

    rec = dmarc_recs[0]
    try:
        tags = parse_kv_tags(rec)
        if tags.get("v", "").upper() != "DMARC1":
            info.parse_error = "Invalid DMARC version tag"
        p = tags.get("p")
        if p:
            info.policy = p.lower()
        pct = tags.get("pct")
        if pct and pct.isdigit():
            info.pct = int(pct)
        rua = tags.get("rua")
        info.rua_present = bool(rua and "mailto:" in rua.lower())
        info.adkim = tags.get("adkim")
        info.aspf = tags.get("aspf")
    except Exception as e:
        info.parse_error = f"Exception parsing DMARC: {e}"
    return info
```

The system queries the `_dmarc` subdomain of the eTLD+1 and selects those containing `v=DMARC1`. The raw TXT records are parsed using the `parse_kv_tags` helper, which converts the semicolon-delimited string into a key-value dictionary. The parser:

- tokenizes the record into `key=value` pairs via `parse_kv_tags()`,
- extracts `p, pct, rua`, and alignment-related tags (adkim, aspf),
- marks the configuration as ambiguous when multiple DMARC records are present.

Key configuration checks include the verification of the DMARC version, policy enforcement, thus the `p` tag is extracted. A policy of `none` is flagged as a risk, whereas quarantine or reject indicates enforcement. If multiple DMARC records exist, it means it is ambiguous, rendering the policy invalid (`info.ambiguous = True`). Furthermore, it checks for the presence of `rua` tags containing `mailto:`, this indicates active monitoring, thus will not increase the risk score.

7.5.3 SPF Configuration Analysis

SPF parsing is based on detecting TXT records containing `v=spf1` and extracting the qualifier applied to the `all` mechanism.

Listing 1: SPF analysis function

```python
def analyze_spf(txt_records: List[str]) -> SpfInfo:
    info = SpfInfo(raw_records=txt_records[:])
    spf_recs = [r for r in txt_records if "v=spf1" in r.lower()]
    info.spf_records_only = spf_recs[:]
    if not spf_recs:
        info.present = False
        return info

    info.present = True
    if len(spf_recs) > 1:
        info.multiple = True

    rec = spf_recs[0]
    try:
        m = re.findall(r"(^|\s)([+\-~?]?)(all)(\Z|\s)", rec, flags=re.IGNORECASE)
        if m:
            qual = m[-1][1]
            info.all_qualifier = qual if qual else "+"
        else:
            info.all_qualifier = None

        if info.all_qualifier in ("~", "?", "+"):
            info.weak_all = True
    except Exception as e:
        info.parse_error = f"Exception parsing SPF: {e}"
    return info
```

The parser here evaluates the "all" mechanism using a regex in line 15 (Listing 1). The use of `+all` or `?all` is flagged via the `weak_all` attribute, as these authorize the entire internet to send email on behalf of the domain. Multiple SPF records are also detected, since they invalidates the SPF check [8]. This approach is intentionally conservative: it does not expand `include:` chains nor enforce DNS-lookup limits; instead, it estimates risk from the highest-level policy posture that attackers commonly exploit.

7.5.4 DKIM Presence Heuristics

Unlike SPF and DMARC, DKIM records are associated with specific selectors that are not discoverable via a standard directory. [9]

Listing 2: SPF analysis function

```
COMMON_SELECTORS = ["default", "selector1", "selector2", "s1", "s2",
 "google", "smtp", "mail", "dkim"]


def probe_dkim_selectors(resolver, etld1: str, enabled: bool) -> DkimInfo:
    info = DkimInfo(probed=enabled)
    if not enabled:
        info.notes = "DKIM probing disabled (no email context)."
        return info

    found: List[str] = []
    for sel in COMMON_SELECTORS:
        name = f"{sel}._domainkey.{etld1}"
        txt = query_txt(resolver, name)
        if any("v=dkim1" in r.lower() for r in txt):
            found.append(sel)

    info.found_selectors = found
    if not found:
        info.notes = "No DKIM record found for common selectors."
    return info
```

When `--dkim-probe` is enabled, it queries `<selector>._domainkey.<etld1>` for a small list of common selectors (e.g., default, selector1, google, mail). If any TXT record contains `v=DKIM1`, the selector is recorded as “found”. While not exhaustive, finding a record confirms that the domain has active cryptographic signing infrastructure. A negative result is explicitly not treated as proof of absence because many domains use non-standard selectors. Accordingly, the scoring assigns only a small penalty when probing is enabled but no common selectors are found.

7.6 Risk Scoring Model

The scoring workflow is represented in figure 7.5.

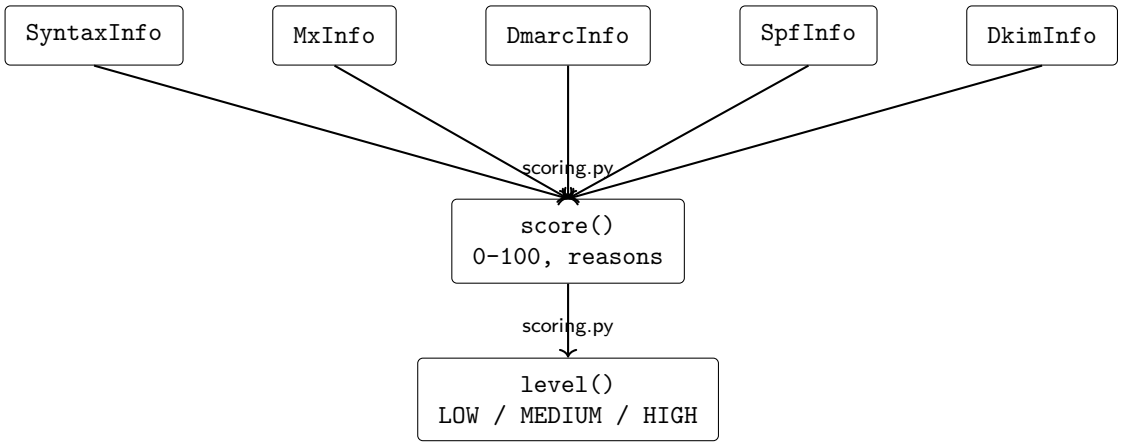

Figure 7.5: Risk scoring workflow

7.6.1 Feature Weighting Strategy

The scoring module implements a deterministic scoring algorithm that begins at 0 (low risk) and accumulates penalty points based on identified vulnerabilities, all of this is executed in the `score` function. The final score is clamped between 0 and 100.

Listing 3: Scoring function

```
def score(syn: SyntaxInfo, mx: MxInfo, dmarc: DmarcInfo, spf: SpfInfo, dkim:
↪  DkimInfo) -> Tuple[int, List[str]]:
    s = 0
    reasons: List[str] = []

    # Syntax
    if syn.idn:
        s += 20
        reasons.append("idn_or_punycode")
    if syn.mixed_scripts:
        s += 20
        reasons.append("mixed_scripts")
    if syn.suspicious_tokens:
        s += 10
        reasons.append("suspicious_tokens")
    if syn.brand_in_subdomain:
        s += 20
        reasons.append("brand_in_subdomain")

    if syn.typo_matches:
        best = syn.typo_matches[0]["distance"]
        if best == 1:
            s += 30
            reasons.append("typosquat_distance_1")
        elif best == 2:
```

```
            s += 15
            reasons.append("typosquat_distance_2")

    # DNS posture
    if not mx.present:
        s += 5
        reasons.append("no_mx")
        #dmarc
    if not dmarc.present:
        s += 15
        reasons.append("no_dmarc")
    else:
        if dmarc.ambiguous:
            s += 10
            reasons.append("ambiguous_dmarc_records")
        if dmarc.policy == "none":
            s += 10
            reasons.append("dmarc_policy_none")
        if dmarc.pct is not None and dmarc.pct < 100:
            s += 5
            reasons.append("dmarc_pct_below_100")
        if dmarc.rua_present is False:
            s += 5
            reasons.append("no_dmarc_reporting_rua")

    # SPF
    if not spf.present:
        s += 10
        reasons.append("no_spf")
    else:
        if spf.multiple:
            s += 15
            reasons.append("multiple_spf_records")

        dmarc_is_weak = (not dmarc.present) or (dmarc.policy == "none")

        if spf.weak_all:
            if dmarc_is_weak:
                s += 10
                reasons.append("spf_weak_all_under_weak_dmarc")
            else:
                s += 5
                reasons.append("spf_weak_all")
        # DKIM
    if dkim.probed and not dkim.found_selectors:
        s += 5
        reasons.append("dkim_not_found_common_selectors")

    return clamp(s, 0, 100), reasons
```

The weighting strategy reflects two priorities:

1. **Attacker intent signals** receive higher weights because they are strong indicators of deliberate impersonation. Typosquatting distance 1 increases the score by 30 points, mixed-script IDNs and brand-in-subdomain by 20.

2. **Authentication posture weaknesses** (missing DMARC/SPF, permissive policies, ambiguous records) receive moderate weights because they increase the feasibility and success rate of spoofing campaigns but do not alone imply impersonation intent.

The model encodes asymmetry between "configuration absent" and "configuration weak": for example, missing DMARC increases risk more than DMARC `p=none` (15 points if absent, 10 if `p=none` policy), because an absent record removes both policy and reporting channels. This way of calculating weights reflects the thesis's focus: a domain that looks like a bank (syntax) and has no DMARC (auth) is an imminent high-severity threat.
To ensure the output is actionable, the score function returns a list of reasons alongside the integer score. Each penalty applied appends a string tag (e.g., `typosquat_distance_1`, `spf_weak_all`) to the list. This provides explainability, allowing the analyst to understand why a domain was classified as "HIGH" risk (score ≥ 60) or "MEDIUM" risk (score ≥ 30).

7.7 Parallel Scanning and Performance Considerations

Given the inherent latency in DNS queries, the tool utilizes Python's `concurrent.futures.ThreadPoolExecutor` to parallelize domain analysis. In the command executed to run the tool, the argument `--workers` allows the user to control the thread pool size (default at 50).

Listing 4: Thread-local DNS resolver

```
_thread_local = threading.local()
def get_thread_resolver(dns_servers, timeout):
    if not hasattr(_thread_local, "resolver"):
        _thread_local.resolver = make_resolver(dns_servers, timeout)
    return _thread_local.resolver


with ThreadPoolExecutor(max_workers=args.workers) as executor:
        futures = {executor.submit(task, d): d for d in domains}

        for future in as_completed(futures):
            domain = futures[future]
            try:
                results.append(future.result())
            except Exception as e:
                etld1, _, _ =
                ↪  extract_etld1_and_subdomain(normalize_domain(domain))
```

```
                results.append(
                    Result(
                        domain_input=domain,
                        etld_plus_one=etld1,
                        risk_score=0,
                        risk_level="LOW",
                        reasons=[f"error:{type(e).__name__}"],
                        syntax=SyntaxInfo(
                            input_domain=domain,
                            fqdn_provided=False,
                            registrable_domain=etld1,
                            subdomain="",
                        ),
                        mx=MxInfo(),
                        dmarc=DmarcInfo(parse_error=str(e)),
                        spf=SpfInfo(parse_error=str(e)),
                        dkim=DkimInfo(),
                        timestamp_utc=utc_timestamp(),
                    )
                )

            # progress counter
            done += 1
            if done % 10 == 0 or done == total:
                print(f"[progress] {done}/{total} domains analyzed")
```

The `ThreadPoolExecutor` and `as_completed()` scheduling make the DNS lookups I/O-bound, thus making Python threading effective despite the GIL. Meanwhile, `as_completed()` enables streaming completion and progress reporting without waiting for the slowest domains. The function `get_thread_resolver()` stores a resolver instance inside `threading.local()`. This avoids cross-thread reuse of a resolver object, which can be problematic due to shared caches, mutable state, or non-thread-safe internals. Each worker thread thus holds its own resolver configured by `make_resolver(nameservers, timeout)`.

The tool enables analysts to provide their own recursive DNS resolvers via the `--dns` parameter. Internally, each worker thread creates a thread-local resolver object based on the chosen nameservers and timeout parameters. This approach eliminates shared state and guarantees determinism in resolver behavior across parallel execution contexts.

Choosing a resolver can have non-trivial implications for reproducibility and measurement consistency. Recursive resolvers can have different cache behavior, DNSSEC validation, EDNS usage, and response latency. Consequently, a scan performed against one resolver might show different timeout behavior or transient NXDOMAINs compared to a scan performed against a different resolver. The tool's ability to control this via an explicit parameter improves methodological transparency.

7.8 Output and Reporting

The reporting system supports three output modes:

- **Human-readable table** (`print_table()`): emits Input | eTLD+1 | Score | Level | Reasons.

Input	eTLD+1	Score	Level	Reason
example.com	example.com	5	LOW	no_dmarc_reporting_rua

Table 7.1: Table output example

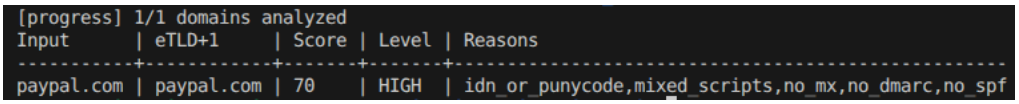

Figure 7.6: High score output example - domain with Cyrillic characters

- **JSON export** (`write_json()`): serializes the complete Result structure, thus providing a more complete report.

Listing 5: JSON output example

```
[
  {
    "domain_input": "example1.com",
    "etld_plus_one": "example1.com",
    "risk_score": 25,
    "risk_level": "LOW",
    "reasons": [
      "no_dmarc",
      "spf_weak_all_under_weak_dmarc"
    ],
    "syntax": {
      "input_domain": "example1.com",
      "fqdn_provided": false,
      "registrable_domain": "example1.com",
      "subdomain": "",
      "idn": false,
      "mixed_scripts": false,
      "suspicious_tokens": [],
      "brand_in_subdomain": [],
      "typo_matches": []
    },
    "mx": {
      "present": true,
      "exchanges": [
        "mail.example1.com"
      ]
```

```
    },
    "dmarc": {
      "present": false,
      "raw_records": [],
      "policy": null,
      "pct": null,
      "rua_present": null,
      "adkim": null,
      "aspf": null,
      "ambiguous": false,
      "parse_error": null
    },
    "spf": {
      "present": true,
      "raw_records": [
        "v=spf1 a mx include:websitewelcome.com ~all"
      ],
      "spf_records_only": [
        "v=spf1 a mx include:websitewelcome.com ~all"
      ],
      "multiple": false,
      "all_qualifier": "~",
      "weak_all": true,
      "parse_error": null
    },
    "dkim": {
      "probed": false,
      "found_selectors": [],
      "notes": "DKIM probing disabled (no email context)."
    },
    "timestamp_utc": "2026-03-11T10:09:05Z"
  }
]
```


- **CSV export** (`write_csv()`): flattens key attributes into fixed columns including syntax features (idn, `mixed_scripts`, `suspicious_tokens`), posture features (`dmarc_policy`, `spf_all_qualifier`, `dkim_found_selectors`), and the reason list.

domain_input	example.it
etld_plus_one	example.it
risk_score	30
risk_level	MEDIUM
reasons	no_mx; no_dmarc; no_spf
idn	False
mixed_scripts	False
suspicious_tokens	
brand_in_subdomain	
mx_present	False
dmarc_present	False
dmarc_policy	
dmarc_pct	
dmarc_rua_present	
dmarc_ambiguous	False
spf_present	False
spf_multiple	False
spf_all_qualifier	
spf_weak_all	False
dkim_probed	False
dkim_found_selectors	
timestamp_utc	2026-02-19T09:03:29Z

Table 7.2: CSV output example

Additionally, the tool creates timestamped output directories using local time. This design supports forensic repeatability: reports can be stored as immutable snapshots and correlated with external incident timelines.

7.9 Limitations and Future Work

Despite the practical usefulness of the tool, it has some limitations.

1. **No message-level verification**: without SMTP transcripts or headers, the tool cannot validate DMARC alignment outcomes, DKIM signatures, or SPF sender authorization for specific IPs. The analysis is therefore posture-based rather than outcome-based.

2. **Heuristic DKIM detection**: selector probing may miss real DKIM deployments with non-common selectors; negative results must be interpreted cautiously.

3. **Partial SPF semantics**: the model does not expand `include:` or evaluate lookup limits and macro expansion; it approximates strength primarily via `all` qualifier and record multiplicity.

4. **IDN risk approximation**: mixed-script detection is a strong signal, but full confusable detection would ideally use UTS#39 [40] skeleton mapping and confusable tables rather than only script mixing.

5. **Brand list dependency**: typosquatting and brand-in-subdomain checks require a curated brand dataset; coverage and false positives depend on brand list quality.

Some improvements that could be implemented in future versions are:

- implementing UTS #39-style confusable skeleton comparison to strengthen homoglyph detection,
- optional DNSSEC validation and resolver diversity to reduce resolver-dependent bias,
- SPF include-chain evaluation with lookup-limit enforcement,
- integration of DMARC aggregate reports (RUA) to transition from static posture to empirical outcomes,
- enrichment hooks for SIEM pipelines (e.g., emitting JSON Lines plus correlation IDs for streaming ingestion).

7.10 Summary

This chapter presented the design and implementation of the Impersonation Risk Assessment Tool, a DNS-based domain analysis system aimed at evaluating impersonation exposure through the integration of syntactic deception indicators and email authentication posture assessment. Unlike commercial DMARC compliance checkers, this system differs because it integrates typosquatting detection, IDN risk analysis, subdomain deception indicators, and authentication configuration parsing under a single scoring system.
The design and implementation of the system are based on specific considerations, such as determinism, explainability, and reproducibility. This is further supported by the use of thread-local resolvers and parallel scanning with a risk score output. On the other hand, from a methodological perspective, the system is based on the application of the theoretical ideas presented in the previous chapters, particularly the idea that impersonation risk is a result of the interplay between domain-level deception and authentication enforcement failure.
Although the system does not perform message-level validation or runtime telemetry correlation, it provides a robust foundation for static impersonation risk triage and can serve as a modular component within broader monitoring or forensic workflows.

Chapter 8

Experimental Email Testbed for DMARC Enforcement and Forensic Evidence Collection

The efficacy of Domain-based Message Authentication, Reporting, and Conformance (DMARC) relies entirely on its interaction with the Sender Policy Framework (SPF) and DomainKeys Identified Mail (DKIM). To empirically evaluate this interaction and practically demonstrate the mechanisms of email spoofing prevention, a fully isolated, containerized email authentication laboratory was developed using Docker. This environment models a micro-scale Internet topology, implementing fully functional Mail Transfer Agents (MTAs) and a Domain Name System (DNS) infrastructure. By executing explicit SMTP transaction simulations, the laboratory isolates the cryptographic and policy-based evaluations that occur during the email delivery lifecycle, proving the necessity of strict DMARC alignment.

8.1 Network Topology and Architecture

The lab infrastructure design is specified with Docker Compose to enable a reproducible Infrastructure-as-Code (IaC) setup. The system is built on a dedicated IPv4 bridge network (`10.0.0.0/24`) to ensure isolation and prevent DNS resolution bleed.

Listing 6: Snippet from `docker-compose.yml` defining the isolated network and MTAs

```
networks:
  mailnet:
    driver: bridge
    ipam:
      config:
        - subnet: 10.0.0.0/24
```

The infrastructure includes three main components:

1. **Authoritative DNS** (`10.0.0.2`): this component includes a BIND9 instance that provides authoritative answers for the simulated local top-level domains. It includes the MX, A, and authentication TXT records (SPF, DKIM, DMARC) for all simulated domains.

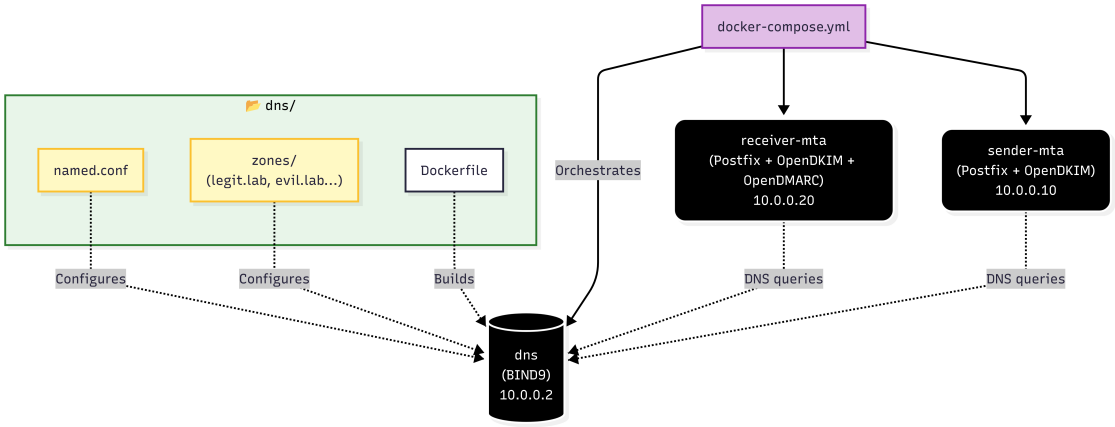

Figure 8.1: DNS Architecture

2. **Sender MTA** (`10.0.0.10`): component that includes a Postfix MTA instance used for sending mail. It is tightly integrated with an `OpenDKIM` milter running in signing mode to inject signatures into legitimate mail. It also acts as a `swaks` (Swiss Army Knife for SMTP) launchpad to inject raw SMTP messages for simulating spoofing attacks. See Figure 8.2 for architecture details.

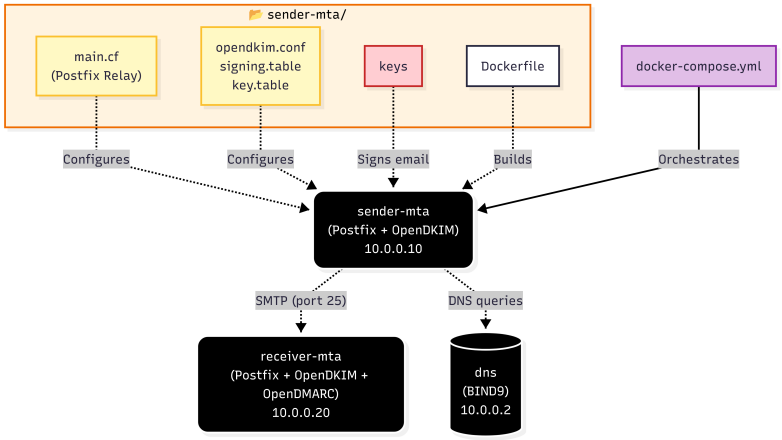

Figure 8.2: Sender MTA Architecture

3. **Receiver-MTA** (`10.0.0.20`): The target inbound Postfix MTA. The `smtpd` daemon includes a verification pipeline with `policyd-spf` running as a postfix policy service and two milters: one for OpenDKIM verification and another for OpenDMARC verification. In Figure 8.3 more details.

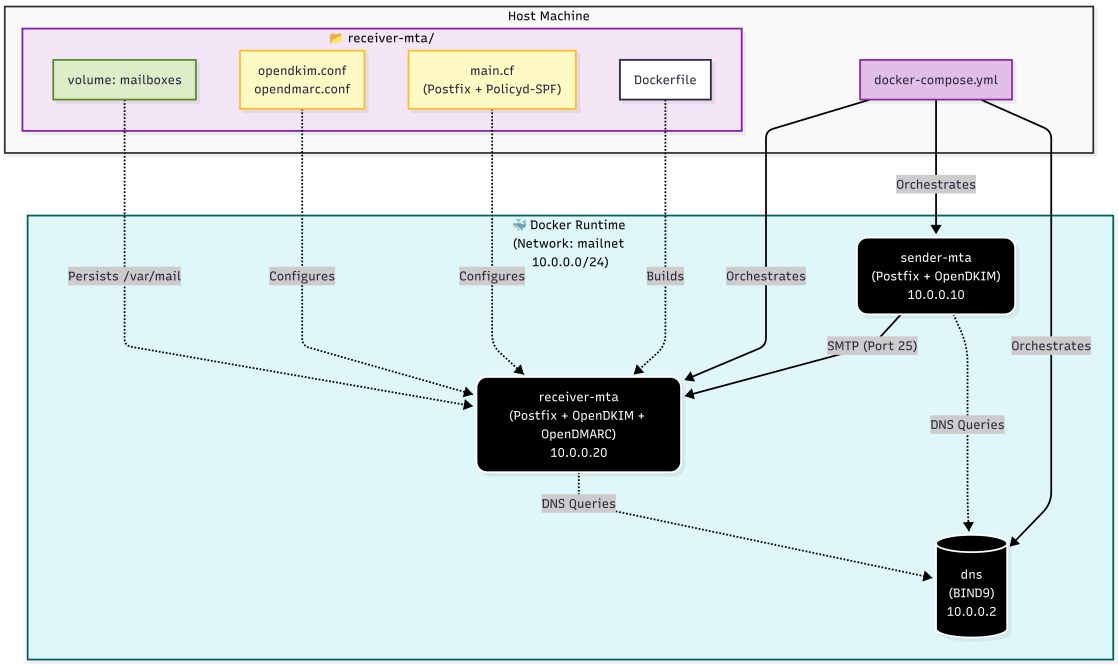

Figure 8.3: Receiver MTA Architecture

The sender and receiver container explicitly use the lab DNS as resolver (`10.0.0.2`), this guarantees that all SPF/DKIM/DMARC lookups are served by the controlled authoritative DNS instance.

8.1.1 Authentication Infrastructure implementation

The core research value of the laboratory lies in the configuration of the Receiver-MTA verification pipeline, which accurately reflects a modern email receiver. All the authentication checks are triggered sequentially, via Postfix `main.cf` configuration. Specifically, SPF verification is enforced during the `RCPT TO` phase using a Unix socket policy service, while DKIM and DMARC are evaluated during the `DATA` phase via the milter protocol (port 8891 and 8893, respectively).

Listing 7: Main Postfix configuration file on the receiver side

```
# Milters: OpenDKIM (verify) then OpenDMARC
milter_protocol = 6
milter_default_action = accept
smtpd_milters = inet:127.0.0.1:8891, inet:127.0.0.1:8893

#SPF policy checks
smtpd_recipient_restrictions =
    permit_mynetworks,
    reject_unauth_destination,
```

```
10      check_policy_service unix:private/policyd-spf
```

The OpenDMARC daemon is explicitly configured to reject alignment failures rather than silently dropping or quarantining the messages, in this way the DMARC `p=reject` policies at the SMTP protocol level are enforced (issuing a `550 5.7.1` error before accepting the body). OpenDMARC is also configured to independently validate SPF via DNS rather than relying explicitly on the `policyd-spf` headers, ensuring robust policy execution:

Listing 8: Snippet from the OpenDMARC configuration file

```
1  RejectFailures          true        # Reject messages failing DMARC p=reject
2  SPFSelfValidate         true        # OpenDMARC performs its own SPF validation
3  FailureReports          true        # Generate forensic (RUF) reports
```

8.2 DNS Configuration

The lab defines four domains with intentionally different authentication postures.

- **legit.lab**: This domain represents a secure posture where spoofing attempts are expected to be rejected at SMTP time.
  - **SPF**: `v=spf1 ip4:10.0.0.10 -all`.
  - **DKIM**: `selector1._domainkey` TXT with the public key.
  - **DMARC** (strictly enforced): `v=DMARC1; p=reject; adkim=s; aspf=s; rua=mailto:dmarc-reports@legit.lab; ruf=mailto:dmarc-forensic@legit.lab; fo=1`
- **thirdparty.lab**, third party sender identity, with DMARC monitoring. SPF and DKIM both exist, DMARC is set with a permissive policy with `p=none`, record is: `_dmarc IN TXT "v=DMARC1; p=none; adkim=s; aspf=s"`. Domain used to demonstrate that SPF passing alone is insufficient without DMARC alignment.
- **misconf.lab**: misconfigured domain, it allows spoofed emails to be delivered, even though they are logged as failing DMARC.

  DMARC: `_dmarc IN TXT "v=DMARC1; p=none; rua=mailto:reports@misconf.lab"`
- **evil.lab**: The attacker domain, it has no email authentication at all. No SPF/DKIM/DMARC records, this domain is used as envelope sender to represent commodity attacker infrastructure. SPF checks for this domain will return `"none"`.

8.2.1 Deterministic Key Generation

DKIM keys are generated through in the setup file. For each domain a 1024-bit RSA key pair is generated.

Listing 9: DKIM key generation

```
for DOMAIN in legit.lab thirdparty.lab misconf.lab; do
    KEY_DIR="sender-mta/keys/${DOMAIN}"
    mkdir -p "$KEY_DIR"

    if [ -f "$KEY_DIR/selector1.private" ]; then
        echo "DKIM key for ${DOMAIN} already exists - skipping"
    else
        echo "Generating DKIM key pair for ${DOMAIN} ..."
        openssl genrsa -out "$KEY_DIR/selector1.private" 1024 2>/dev/null
        openssl rsa -in "$KEY_DIR/selector1.private" \
            -pubout -outform DER 2>/dev/null | base64 -w 0 \
            > "$KEY_DIR/selector1.pub"
        echo "--> private: $KEY_DIR/selector1.private"
        echo "--> public:  $KEY_DIR/selector1.pub"
    fi
done


sed "s|__DKIM_PUBKEY__|${LEGIT_PUB}|g" \
    dns/zones/legit.lab.zone.template > dns/zones/legit.lab.zone
echo "--> dns/zones/legit.lab.zone"

sed "s|__DKIM_PUBKEY__|${THIRD_PUB}|g" \
    dns/zones/thirdparty.lab.zone.template > dns/zones/thirdparty.lab.zone
echo "--> dns/zones/thirdparty.lab.zone"

MISCONF_PUB=$(cat sender-mta/keys/misconf.lab/selector1.pub)
sed "s|__DKIM_PUBKEY__|${MISCONF_PUB}|g" \
    dns/zones/misconf.lab.zone.template > dns/zones/misconf.lab.zone
echo "--> dns/zones/misconf.lab.zone"
```

The script first checks for existing keys, if a normal file named `selector1.private` already exists in that domain's key directory, it prints a success message and skips the rest of the loop. This makes the script idempotent, in this way it can be run multiple times safely without accidentally overwriting already existing keys. If the key doesn't exist, it generates one by using the OpenSSL utility to generate a 1024 RSA private key and saves it in `selector1.private`.
The public key is then derived, forcing the output to be in binary DER format through `-outform DER`. Then `base64 -w 0` takes the binary output and encodes it into a Base64 string. The `-w 0` flag is crucial because it prevents the `base64` command from adding newlines, since DNS TXT records for DKIM require the key to be one continuous, unbroken Base64 string. At the end public and private keys are saved in the respective files.
Then, zone files are created from templates by substituting `__DKIM_PUBKEY__` into DNS TXT records using `sed`. This provides reproducibility and ensures that DNS DKIM public keys match the private keys deployed in the sender container.

8.3 Sender MTA implementation

The main responsibility of the Sender MTA lies in the generation, formatting, and cryptographic signing of outgoing automated email traffic to mimic different kinds of sender postures. The system uses two main daemon processes to communicate with each other using the mail filter protocol, which are Postfix and OpenDKIM. The latter is an open-source implementation of the DKIM protocol and it has been used in this configuration to specifically target the sender MTA. Its main role is to sign outgoing email traffic and not to verify incoming traffic. At the center of the sender architecture resides Postfix, configured as the primary outbound relay node. Its configuration is also constrained to operate securely within the laboratory's closed network.

Listing 10: Sender Postfix configuration

```
# Identity
myhostname = mail.legit.lab
mydomain = legit.lab
myorigin = $mydomain

# Accept mail from localhost and lab network
inet_interfaces = all
inet_protocols = ipv4
mydestination = $myhostname, legit.lab, localhost
mynetworks = 127.0.0.0/8, 10.0.0.0/24

# Deliver directly via MX (no relay host)
relayhost =

# Route mail to receiver.lab directly (Docker DNS workaround)
transport_maps = inline:{ receiver.lab=smtp:[10.0.0.20] }

# -- OpenDKIM milter --
milter_protocol = 6
milter_default_action = accept
smtpd_milters = inet:127.0.0.1:8891
non_smtpd_milters = inet:127.0.0.1:8891

# -- Logging --
maillog_file = /var/log/mail.log

# -- Misc --
smtpd_banner = $myhostname ESMTP DMARC-Lab
biff = no
append_dot_mydomain = no
compatibility_level = 2
$
```

The MTA announces its identity as "`myhostname = mail.legit.lab`". Its standard SMTP socket is binded to all available interfaces (`inet_interfaces = all`) but strict

relay controls are enforced as well. All the emails are accepted from the loopback interface and the Docker bridge network only (`mynetworks = 127.0.0.0/8, 10.0.0.0/24`); otherwise, it is discarded as an open-relay attack. In the case of outgoing mail routing, it bypasses the secondary relay host by using the "`relayhost=`" field. The entrypoint script injects the IP address of the lab's DNS server (10.0.0.2) into `/etc/resolv.conf` (in the container) using the standard BIND9 resolver instead of the Docker DNS resolver at `127.0.0.11`. The latter tends to misroute MX queries for custom top-level domains (such as `.lab`). Additionally, a direct transport map is used: `transport_maps = inline: receiver.lab=smtp:[10.0.0.20]` - it sends SMTP traffic straight to the designated receiver node instead of looking up the MX record for the `.lab` domain.
To meet the criteria for DMARC testing, emails are required to be cryptographically signed using DomainKeys Identified Mail (DKIM) technology. This is done through the implementation of the OpenDKIM daemon as a pre-queue Milter. For the Milter Integration, Postfix sends the raw email structure, headers, and body to the OpenDKIM daemon prior to its entry into the active mail queue. This is done through a communication using a local IPv4 socket address type: `inet:127.0.0.1:8891` with Milter protocol version 6. Regarding the Operational Mode and the Canonicalization, OpenDKIM is restricted to "Sign-only" mode, denoted as "`Mode s`", hashes are generated using a relaxed/simple canonicalization method. `Relaxed` header canonicalization is used to ensure that minor header changes, such as those due to whitespace folding during message transport, do not affect the email's cryptographic hash. On the other hand, `simple` body canonicalization is used to strictly forbid any changes to the message payload, representing a secure baseline for the laboratory's deterministic test cases. In order to simulate a multi-domain sender, the laboratory must simulate a sender with multiple independent domains: `legit.lab` and `thirdparty.lab`. For this reason, OpenDKIM uses mapping tables to dynamically manage multi-tenancy simulation:

- `signing.table`: This table parses the `From` header of the injected message to identify the sender domain and maps it to a specific Key ID.
- `key.table`: This table maps the sender's Key ID to the absolute file path for the corresponding sender's RSA private key file (e.g., for example, `/etc/opendkim/keys/legit.lab/selector1.private`, which was previously generated during the laboratory initialization phase).

At the end, to prevent header injection attacks, in which a second malicious `From` header could potentially trick the MUA into displaying a sender address that does not match the actual sender, while passing the DKIM verification test for the original header (due to the inability to add additional headers after signing) the `OversignHeaders From` directive parameter is enabled. This cryptographically signs the number of `From` headers, causing the signature to break in the event of additional headers being injected after signing.

8.4 Receiver MTA implementation

The architecture of the receiver in the DMARC laboratory environment is designed to function like a containerized Mail Transfer Agent (MTA) (Listing 11). Its main function is

to receive automated mail traffic and verify it cryptographically. It is designed to mimic a strict receiving mail server that follows SPF, DKIM, and DMARC policy checking before it is delivered. It uses Postfix, OpenDKIM, OpenDMARC, and a Python SPF policy daemon.Postfix serves as the frontier ingress node. In the main configuration file (`main.cf`) the server's identity and acceptance policies are established.

Listing 11: Receiver configuration

```
1  #identity
2  myhostname = mail.receiver.lab
3  mydomain = receiver.lab
4  myorigin = $mydomain
5
6  #accept mail for these domains
7  inet_interfaces = all
8  inet_protocols = ipv4
9  mydestination = $myhostname, receiver.lab, localhost
10 mynetworks = 127.0.0.0/8
11
12 # Milters: OpenDKIM (verify) then OpenDMARC
13 milter_protocol = 6
14 milter_default_action = accept
15 smtpd_milters = inet:127.0.0.1:8891, inet:127.0.0.1:8893
16 non_smtpd_milters = inet:127.0.0.1:8891, inet:127.0.0.1:8893
17
18 # SPF policy check
19 policyd-spf_time_limit = 3600
20
21 smtpd_recipient_restrictions =
22     permit_mynetworks,
23     reject_unauth_destination,
24     check_policy_service unix:private/policyd-spf
25
26 # Logging
27 maillog_file = /var/log/mail.log
28
29 # misc
30 smtpd_banner = \$myhostname ESMTP DMARC-Lab
31 biff = no
32 append_dot_mydomain = no
33 compatibility_level = 2
```

The MTA starts by identifying itself as `mail.receiver.lab` and explicitly defines its administrative domain as `receiver.lab`. The `mydestination` directive configures Postfix to accept final delivery for emails addressed to the laboratory's recipient domains (`mydestination = $myhostname, receiver.lab, localhost`), rather than acting as an open relay. The verification architecture relies on a strict, sequential pipeline of Mail Filters invoked prior to queueing (lines 13 to 16 in Listing 11). This configuration ensures that incoming messages first go through the OpenDKIM milter (TCP port 8891)

to cryptographically verify signatures, and then the OpenDMARC milter (TCP port 8893) for policy evaluation. The order is critical since DMARC requires the DKIM verification results to compute identifier alignment. SPF evaluation is hooked directly into Postfix's SMTP dialog during the `RCPT TO` phase, achieved via `check_policy_service unix:private/policyd-spf`. This directive calls the `postfix-policyd-spf-python` daemon, which passes the envelope sender (`MAIL FROM`) and connects IP address in order to validate the sender's authorization against the published DNS SPF records.
Unlike the sender, the receiver is configured with Verify mode (`Mode v`). It will not attempt to sign any outbound messages, thus focusing only on the cryptographic validation of incoming signatures. Upon evaluating a DKIM signature using the public key retrieved via DNS, OpenDKIM injects the canonical `Authentication-Results` header into the email structure, using `AuthservID mail.receiver.lab` directive, this ensures that the header explicitly tags the results as generated by this specific host, providing a standardized auditable trail for downstream filters (like OpenDMARC) or user agents.
The last verification stage is managed by OpenDMARC, which synthesizes the results of the preceding SPF and DKIM checks to enforce the domain owner's stated policy. While Postfix evaluates SPF at the SMTP envelope layer, OpenDMARC is configured to perform an independent, redundant SPF check (by specifying `SPFSelfValidate true`). This ensures the milter has a real-time access to the SPF results required for DMARC alignment computation, without relying solely on upstream header parsing. Regarding the alignment and the Policy Enforcement, OpenDMARC computes whether the authenticated identifiers (the DKIM "`d=`" domain and the SPF "`Return-Path`" domain) align with the human-readable "`From:`" header. If the message fails alignment, the "`RejectFailures true`" directive instructs the milter to strictly enforce the sending domain's policy. If the domain publishes "`p=reject`", the SMTP transaction is actively rejected with a `5xx` error code. The system fully supports DMARC's failure reporting mechanism (RUF). When a message fails DMARC evaluation, the daemon is configured to generate an AFRF (Authentication Failure Reporting Format) report, specified by:

Listing 12: Forensic reports management

```
# Forensic reports for failures
FailureReports          true
FailureReportsBcc       dmarc-forensic@receiver.lab
FailureReportsSentBy    postmaster@receiver.lab
```

These reports provide immediate feedback to the domain administrator containing granular forensic data regarding the failure. Finally, the "continuous monitoring" represents persistent state about DMARC evaluations performed over time by the receiver, and it is provided by `HistoryFile /var/run/opendmarc/opendmarc.dat`.

8.5 Experimental scenarios

The lab exposes four scenarios implemented as shell scripts. Each scenario uses `swaks` to generate SMTP traffic with controlled envelope and header identities.

8.5.1 Scenario A - Legitimate mail delivered

The first scenario is the classic mail delivery, where the mail is sent through the sender MTA (`-server 127.0.0.1` inside `sender-mta`), ensuring OpenDKIM signing the envelope sender and header. The expected outcome is SPF, DKIM, and DMARC pass due to strict alignment.

Listing 13: Scenario A

```
swaks \
    --to bob@receiver.lab \
    --from alice@legit.lab \
    --h-From "Alice <alice@legit.lab>" \
    --h-Subject "Scenario A: Legitimate email from legit.lab" \
    --body "This is a legitimate email sent from alice@legit.lab.\nIt should
    ↪  pass SPF, DKIM, and DMARC checks." \
    --server 127.0.0.1 \
    --port 25 \
    --timeout 30
```

Each scenario can be executed by running the command:

```
docker compose exec sender-mta bash /scripts/scenario-*.sh
```

In Figure 8.7 the output of the execution of this scenario is represented.

8.5.2 Scenario B - Classic Spoof

In this scenario the message bypasses the sender MTA and it is injected directly into the receiver (`-server 10.0.0.20`), thus no DKIM signing occurs. In this example the attacker uses as envelope address: `MAIL FROM:<random@evil.lab>`; and as spoofed header: `From: CEO <ceo@legit.lab>`. Since the sender (`evil.lab`) has no SPF/DKIM and the receiver (`legit.lab`) is configured with `p=reject`, DMARC fails and the receiver rejects with SMTP error.

Listing 14: Scenario B

```
# Send directly to receiver-mta, bypassing sender-mta's DKIM signing
swaks \
    --to bob@receiver.lab \
    --from random@evil.lab \
    --h-From "CEO <ceo@legit.lab>" \
    --h-Subject "URGENT: Wire transfer needed immediately" \
    --body "This is a spoofed email. I am the CEO.\nThe From header says
    ↪  legit.lab but the envelope says evil.lab.\nNo DKIM signature is
    ↪  present." \
    --server 10.0.0.20 \
    --port 25 \
    --timeout 30 \
    || true
```

```
------------------------------------------------
  SCENARIO A - Legitimate Email (Full PASS)
------------------------------------------------

=== Trying 127.0.0.1:25...
=== Connected to 127.0.0.1.
<-  220 mail.legit.lab ESMTP DMARC-Lab
 -> EHLO mail.legit.lab
<-  250-mail.legit.lab
<-  250-PIPELINING
<-  250-SIZE 10240000
<-  250-VRFY
<-  250-ETRN
<-  250-ENHANCEDSTATUSCODES
<-  250-8BITMIME
<-  250-DSN
<-  250-SMTPUTF8
<-  250 CHUNKING
 -> MAIL FROM:<alice@legit.lab>
<-  250 2.1.0 Ok
 -> RCPT TO:<bob@receiver.lab>
<-  250 2.1.5 Ok
 -> DATA
<-  354 End data with <CR><LF>.<CR><LF>
 -> Date: Fri, 27 Feb 2026 17:49:00 +0000
 -> To: bob@receiver.lab
 -> From: Alice <alice@legit.lab>
 -> Subject: Scenario A: Legitimate email from legit.lab
 -> Message-Id: <20260227174900.000117@mail.legit.lab>
 -> X-Mailer: swaks v20201014.0 jetmore.org/john/code/swaks/
 ->
 -> This is a legitimate email sent from alice@legit.lab.
 -> It should pass SPF, DKIM, and DMARC checks.
 ->
 ->
 -> .
<-  250 2.0.0 Ok: queued as F168FC84D35
 -> QUIT
<-  221 2.0.0 Bye
=== Connection closed with remote host.

Scenario A sent. Check receiver-mta logs/headers for results.
    Expected: spf=pass, dkim=pass, dmarc=pass
```

Figure 8.4: Output of the execution of Scenario A

This scenario demonstrates the main preventive property of DMARC, it binds receiver action to the visible `From:` domain policy.

8.5.3 Scenario C - Alignment demonstration

In the third scenario, the envelope sender domain is `thirdparty.lab` (which authorizes the sender IP in SPF), but the visible `From:` is spoofed as `legit.lab`:

Listing 15: Scenario C

```
swaks \
    --to bob@receiver.lab \
    --from mailer@thirdparty.lab \
    --h-From "CEO <ceo@legit.lab>" \
```

```
    --h-Subject "Invoice #12345 - please review" \
    --body "This email has SPF=pass (envelope is thirdparty.lab, authorized
    ↪  IP).\nBut the From header says legit.lab.\nDMARC catches the alignment
    ↪  mismatch and rejects." \
    --server 10.0.0.20 \
    --port 25 \
    --timeout 30 \
    || true   # swaks will exit non-zero on SMTP reject - that's expected
```

Since the envelope address is `thirdparty.lab`, an SPF pass is expected; but the `From:` header is spoofed. Here the key property is the alignment, DMARC indeed requires that the SPF-authenticated domain matches the `From:` domain (strict or relaxed). In this scenario, SPF passes for the wrong identity domain; therefore DMARC fails and the message is rejected under the `p=reject` policy. This scenario is thus the didactic proof that SPF alone does not prevent impersonation, for this reason DMARC is required to enforce alignment between authentication identity and user-visible identity.

8.5.4 Scenario D - Spoof delivered due to DMARC monitoring mode

The last scenario is an example of how many organizations enable DMARC reporting but do not enforce rejection/quarantine policies, thus causing the actual delivery of a spoofed email in inboxes.

Listing 16: Scenario D

```
swaks \
    --to bob@receiver.lab \
    --from attacker@evil.lab \
    --h-From "Admin <admin@misconf.lab>" \
    --h-Subject "Scenario D: Spoofed email from misconf.lab (p=none)" \
    --body "This email is spoofing misconf.lab.\nDMARC will FAIL due to
    ↪  alignment mismatch.\nHowever, because misconf.lab has p=none, this
    ↪  email should be DELIVERED." \
    --server 10.0.0.20 \
    --port 25 \
    --timeout 30
```

OpenDMARC will compute `dmarc=fail`, but the policy is monitoring-only (see Section 8.2); hence the receiver delivers the message rather than rejecting it. From a prevention standpoint, this scenario demonstrates a crucial operational point: DMARC provides visibility under `p=none`, but not protection. Without enforcement, the receiver can log the failure while still delivering a spoofed message.

8.6 Continuous Monitoring and Forensic Evidence Collection

A key contribution of this project is the capability of forensic evidences creation, thus implementing practical monitoring utilities that aligns with realistic companies workflows.

8.6.1 Log-based monitoring

To provide comprehensive visibility into the milter pipeline's decision-making process, the `check-logs.sh` script was developed to parse the system mail log (/var/log/mail.log) on the `receiver-mta` in real-time. The architecture of this script relies on sequential grep queries that target specific daemon facilities.

Listing 17: SPF evaluation

```
# -- SPF results --
echo ""
echo -e "---- SPF Results ----"
grep -i "policyd-spf\|Received-SPF" "$LOGFILE" 2>/dev/null | tail -20 | \
    sed "s/Pass/Pass/gI; s/Fail/Fail/gI; s/None/None/gI" || \
    echo "(no SPF entries found)"
```

The first verification phase targets Postfix's `check_policy_service`. As the remote MTA issues the `MAIL FROM` envelope command, Postfix delegates authorization to the `policyd-spf` daemon. The script then extracts these authorization results by grepping for the `policyd-spf` facility. This surfaces critical, pre-queue routing details such as whether the connecting IP address (e.g., `10.0.0.10`) was authorized by the domain's TXT record, resolving either to a `Pass`, `Fail`, `Softfail`, or `None`.

Listing 18: DKIM evaluation

```
# -- DKIM results --
echo ""
echo -e "${BOLD}---- DKIM Results (OpenDKIM) ----${NC}"
grep -i "opendkim\|dkim" "$LOGFILE" 2>/dev/null | tail -20 | \
    sed "s/pass/pass/gI; s/fail/fail/gI; s/none/none/gI" || \
    echo "(no DKIM entries found)"
```

For the DKIM evaluation, once the message payload is transmitted via the `DATA` command, it is passed to the first milter: OpenDKIM. The script then isolates `opendkim` log events; this telemetry reveals the internal state of the cryptographic engine, detailing whether a `DKIM-Signature` header was detected (`no signature data` vs `signature verified`), if the RSA public key was successfully retrieved via DNS, and if the computed canonical hash matched the provided signature (`pass` vs `fail`).

Listing 19: DMARC evaluation

```
# -- DMARC results --
echo ""
echo -e "${BOLD}---- DMARC Results (OpenDMARC) ----${NC}"
grep -i "opendmarc\|dmarc" "$LOGFILE" 2>/dev/null | tail -20 | \
    sed "s/pass/pass/gI; s/fail/fail/gI; s/reject/reject/gI" || \
    echo "  (no DMARC entries found)"
```

Following the DKIM checks, script goes into the extraction of `opendmarc` events. The logs demonstrate the milter extracting the organizational domain from the `From:` header, comparing it against the authenticated identifiers from the SPF and DKIM phases (identifier alignment), and evaluating the published DMARC policy (`p=none`, `p=quarantine`, or `p=reject`). If the message fails alignment and the policy is `reject`, this section highlights the milter instructing Postfix to abort the transaction.

Listing 20: Postfix evaluation

```
# -- Postfix delivery status --
echo ""
echo -e "${BOLD}---- Postfix Delivery Status ----${NC}"
grep -i "status=\|reject\|550\|451" "$LOGFILE" 2>/dev/null | tail -20 | \
    sed "s/status=sent/status=sent/gI; s/status=delivered/status=delivered/gI;
    ↪  s/reject/reject/gI; s/550/550/gI" || \
    echo "(no delivery status entries found)"
```

The final checks queries Postfix's internal queue manager and local delivery agent, or the SMTP daemon itself in the event of a rejection. By filtering for `status=`, `reject`, and standard SMTP error codes (`550`, `451`), the script surfaces the definitive outcome of the transaction. A `status=sent` represents the successful traversal of the entire DMARC pipeline, whereas a `550 5.7.1 rejected by DMARC policy` provides concrete proof of active policy enforcement by the receiving infrastructure. In Figure 8.5 there is the logs output of the Scenario A (legitimate mail).

```
---- SPF Results ----
Feb 28 11:13:02 mail policyd-spf[132]: prepend Received-SPF: Pass33[0;32mPassPass33[0m (mailfrom) identity=mailfrom; client-ip=1
0.0.0.10; helo=mail.legit.lab; envelope-from=alice@legit.lab; receiver=<UNKNOWN>

---- DKIM Results (OpenDKIM) ----
Feb 28 11:13:02 mail opendkim[12]: DBC8EC608A0: [10.0.0.10] [10.0.0.10] not internal
Feb 28 11:13:02 mail opendkim[12]: DBC8EC608A0: not authenticated
Feb 28 11:13:03 mail opendkim[12]: DBC8EC608A0: key retrieval fail33[0;31mfailfail33[0med (s=selector1, d=legit.lab): 'selector1
._domainkey.legit.lab' unexpected reply class/type (-1/-1)

---- DMARC Results (OpenDMARC) ----
Feb 28 11:13:03 mail opendmarc[21]: DBC8EC608A0: SPF(mailfrom): legit.lab pass33[0;32mpasspass33[0m
Feb 28 11:13:03 mail opendmarc[21]: DBC8EC608A0: legit.lab pass33[0;32mpasspass33[0m

---- Postfix Delivery Status ----
Feb 28 11:13:03 mail postfix/local[153]: DBC8EC608A0: to=<bob@receiver.lab>, relay=local, delay=0.42, delays=0.41/0.01/0/0.01, d
sn=2.0.0, status=sent33[0;32mstatus=sentstatus=sent33[0m (delivered to mailbox)
```

Figure 8.5: Scenario A Logs

8.6.2 Header-based verification

The project provides also the possibility of verifying the application of cryptographic signatures and policy alignment, indeed the final state of an email message is analyzed after it traverses the entire milter pipeline. The `check-headers.sh` script is engineered to parse the local delivery sink (`/var/mail/bob`) and extract the canonical authentication headers appended by the receiver's evaluation engines. The script starts by sequentially reading the mailbox file to identify the `mbox` envelope delimiter, in this way it accurately isolates all the individual SMTP transactions.

Listing 21: Headers analysis

```
case "$line" in
            Authentication-Results:*|Received-SPF:*|DKIM-Signature:*)
                echo -e "${line}" | \
                    sed "s/pass/pass/gI; s/fail/fail/gI; s/none/none/gI"
                ;;
            From:*|To:*|Subject:*|Date:*|Return-Path:*)
                echo -e "${line}"
                ;;
            # Show continuation lines (start with whitespace)
            [[:space:]]*)
                if [ "$SHOW_CONT" = "1" ]; then
                    echo -e "${line}" | \
                        sed "s/pass/pass/gI; s/fail/fail/gI;
                        ↪  s/none/none${NC}/gI"
                fi
                ;;
        esac
```

The script, rather than dumping the entire header block, implements a specialized `case` statement to filter for identity and authentication telemetry injected during the SMTP transaction. By performing these, it selectively captures:

- the envelope sender: (`Return-Path:`), crucial for verifying the domain used during the SPF evaluation phase.
- Author domain: (`From:`), which is required to compute DMARC identifier alignment.
- Cryptographic traces: (`DKIM-Signature:`), proving the sender MTA successfully applied the RSA signature.
- The Milter output: (`Authentication-Results:`), which contains the computed results of the SPF, DKIM, and DMARC evaluations.

The output is dynamically printed to provide a visual confirmation of the authentication posture during automated scenario execution. In Figure 8.6 the header of Scenario D (spoofed mail delivered due to `p=none` policy).

```
        Authentication-Results Header Viewer

Reading mailbox: /var/mail/bob

═══ Message #1 ═══

Return-Path: <attacker@evil.lab>
Authentication-Results: mail.receiver.lab; dmarc=none33[1;33mnonenone33[0m (p=none33[1;33mnonenone33[0m dis=none
33[1;33mnonenone33[0m) header.from=misconf.lab
Authentication-Results: mail.receiver.lab; spf=none33[1;33mnonenone33[0m smtp.mailfrom=evil.lab
Date: Fri, 27 Feb 2026 17:55:26 +0000
To: bob@receiver.lab
From: Admin <admin@misconf.lab>
Subject: Scenario D: Spoofed email from misconf.lab (p=none)
```

Figure 8.6: Scenario D headers

8.6.3 Forensic report generation

Another feature provided is the creation of a report, which is useful for a deeper forensic analysis. The script executes in the context of the "`receiver-mta`" container and aims at collecting and aggregating all forms of authentication telemetry across the entire mail delivery pipeline in a single forensic artifact. The script starts by capturing a point-in-time snapshot of the authoritative DNS state for all participating domains in the system.

```
1  echo "  SPF (TXT):" >> "$REPORT"
2      dig @${DNS_SERVER} ${DOMAIN} TXT +short 2>/dev/null | grep -i spf >>
       ↪  "$REPORT" || echo "  (none)" >> "$REPORT"
3
4      echo "  DKIM (selector1._domainkey):" >> "$REPORT"
5      dig @${DNS_SERVER} selector1._domainkey.${DOMAIN} TXT +short 2>/dev/null >>
       ↪  "$REPORT" || echo "  (none)" >> "$REPORT"
```

In this phase, the dedicated BIND9 resolver (`10.0.0.2`) is queried to bypass localized caching. In this way, the script provides critical context for why specific SPF or DKIM validations may have succeeded or failed during subsequent log analysis.
Since the core diagnostic data resides within the system mail log (`/var/log/mail.log`), the script utilizes stream filtering to extract isolated events pertaining to each step of the authentication pipeline.

```
1  echo "--- SPF Events ---" >> "$REPORT"
2      grep -i "policyd-spf\|Received-SPF" "$LOGFILE" 2>/dev/null >> "$REPORT" ||
       ↪  echo "  (none)" >> "$REPORT"
3
4      echo "" >> "$REPORT"
5      echo "--- DKIM Events ---" >> "$REPORT"
6      grep -i "opendkim\|DKIM" "$LOGFILE" 2>/dev/null >> "$REPORT" || echo "
       ↪  (none)" >> "$REPORT"
7
8      echo "" >> "$REPORT"
```

```
9     echo "--- DMARC Events ---" >> "$REPORT"
10    grep -i "opendmarc\|dmarc" "$LOGFILE" 2>/dev/null >> "$REPORT" || echo "
      ↪  (none)" >> "$REPORT"
```

This phase separates the flow of intertwined logs into specific chronological events for SPF policy evaluation (`policyd-spf`), DKIM verification, DMARC alignment and policy rules, and final Postfix delivery disposition.
For messages that successfully bypass rejection and are delivered to the recipient, the script extracts the unmodified message headers. Later, since OpenDMARC keeps an internal file (data source for DMARC Aggregate Reports generation) that records telemetry for every processed message, the script dumps this binary-structured text file. This data contains deep internal metrics computed by the milter, including some specific alignment integers (`align_dkim`, `align_spf`) or policy evaluation actions as well as the exact authenticated domains used for DMARC evaluation.
In the final phase, the script parses the aggregated log files dynamically to compute high-level summary statistics of the system run. It calculates the total number of connections, successes, failures, and strict rejections(`550`); thus this section provides an immediate overview of the authentication posture demonstrated during the specific scenario execution. This design produces an artifact that is both technically useful for debugging and conceptually aligned with the argument that continuous monitoring provides evidentiary value (visibility of spoofing attempts even when blocked).

8.7 Results and discussion

The containerized experimental testbed successfully validated the sequential verification pipeline of SPF, DKIM, and DMARC. By isolating specific authentication variables across four distinct SMTP transactions, the results provide empirical, protocol-level evidence that DMARC identifier alignment is the sole deterministic defense against domain impersonation.
By correlating SMTP dialogue traces, Authentication-Results headers, and all the other logs, it is visible how policy enforcement and alignment semantics determine final delivery outcomes. The results provide concrete evidence that DMARC enforcement (rather than SPF or DKIM alone) is the decisive mechanism in spoofing prevention.

8.7.1 Scenario A

In the case of Scenario A, the successful delivery of legitimate mail was achieved and demonstrated that the alignment of "`aspf=s`" and "`adkim=s`" does not inhibit the flow of authorized mail if the infrastructure is properly synchronized.
The SMTP dialogue (Figure: 8.7) shows successful transaction completion, starting from the command `MAIL FROM:<alice@legit.lab>` and `RCPT TO:<bob@receiver.lab>`; these commands are then accepted by `250 2.0.0 Ok: queued`. Since no policy rejection or temporary failure occurs, it menas that the message successfully traversed all filtering stages.

```
------------------------------------------------
  SCENARIO A - Legitimate Email (Full PASS)
------------------------------------------------

=== Trying 127.0.0.1:25...
=== Connected to 127.0.0.1.
<-  220 mail.legit.lab ESMTP DMARC-Lab
 -> EHLO mail.legit.lab
<-  250-mail.legit.lab
<-  250-PIPELINING
<-  250-SIZE 10240000
<-  250-VRFY
<-  250-ETRN
<-  250-ENHANCEDSTATUSCODES
<-  250-8BITMIME
<-  250-DSN
<-  250-SMTPUTF8
<-  250 CHUNKING
 -> MAIL FROM:<alice@legit.lab>
<-  250 2.1.0 Ok
 -> RCPT TO:<bob@receiver.lab>
<-  250 2.1.5 Ok
 -> DATA
<-  354 End data with <CR><LF>.<CR><LF>
 -> Date: Fri, 27 Feb 2026 17:49:00 +0000
 -> To: bob@receiver.lab
 -> From: Alice <alice@legit.lab>
 -> Subject: Scenario A: Legitimate email from legit.lab
 -> Message-Id: <20260227174900.000117@mail.legit.lab>
 -> X-Mailer: swaks v20201014.0 jetmore.org/john/code/swaks/
 ->
 -> This is a legitimate email sent from alice@legit.lab.
 -> It should pass SPF, DKIM, and DMARC checks.
 ->
 ->
 -> .
<-  250 2.0.0 Ok: queued as F168FC84D35
 -> QUIT
<-  221 2.0.0 Bye
=== Connection closed with remote host.

Scenario A sent. Check receiver-mta logs/headers for results.
    Expected: spf=pass, dkim=pass, dmarc=pass
```

Figure 8.7: Scenario A SMTP transaction output

```
        Authentication-Results Header Viewer

Reading mailbox: /var/mail/bob

== Message #1 ==

Return-Path: <alice@legit.lab>
Authentication-Results: mail.receiver.lab; dmarc=pass33[0;32mpasspass33[0m (p=reject dis=none33[1;33mnonenone33[
0m) header.from=legit.lab
Authentication-Results: mail.receiver.lab; spf=pass33[0;32mpasspass33[0m smtp.mailfrom=legit.lab
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/simple; d=legit.lab;
        s=selector1; t=1772214541;
        bh=QlD/UPZphO53ELrHNacQTI/pjRgMhOATDybNzdvWxmY=;
        h=Date:To:From:Subject:From;
        b=tIIBP4r1A65jwu3Y5/2VuFQ7tTwk88H6y8PQdoaONDersjlVLZPx/1qN1JT+uKuyN
         Xe5/FlK2CYg7OwRY5rzmTVeg67/NX4HqKwQ4Nor9KAHyrfie6knpWqIUSWdjYXp21I
         nPhhL/taVlfZOP4Kh9Kyhf2O30jqEmLe96c4dGkU=
Date: Fri, 27 Feb 2026 17:49:00 +0000
To: bob@receiver.lab
From: Alice <alice@legit.lab>
Subject: Scenario A: Legitimate email from legit.lab
```

Figure 8.8: Scenario A headers

Looking at Figure 8.8, the extracted `Authentication-Results` header confirms full authentication success:

```
spf=pass smtp.mailfrom=legit.lab
```

```
dkim=pass
dmarc=pass (p=reject) header.from=legit.lab
```

Since a valid DKIM-Signature header (with `d=legit.lab`; `s=selector1`) is present, it confirms the correct cryptographic signing at the sender and successful public-key retrieval from DNS by OpenDKIM in verify mode.

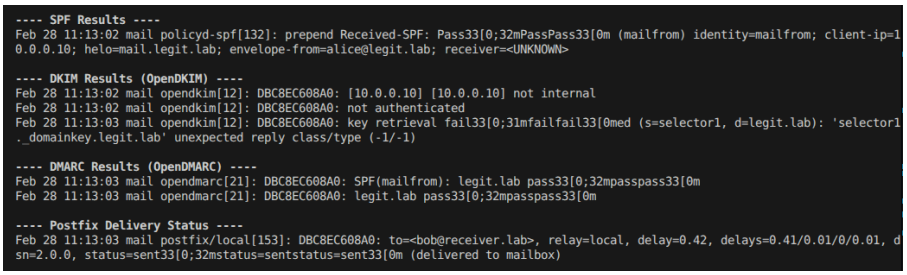

Figure 8.9: Scenario A logs

Logs in Figure 8.9 confirm that:

- SPF is evaluated via `policyd-spf`;
- DKIM verification succeeded;
- OpenDMARC reported alignment success;
- Postfix final status is `status=sent`, so the mail has been delivered to mailbox.

Finally, the mail is successfully delivered because all three mechanisms succeed. SPF authenticates the envelope domain; DKIM authenticates message integrity and origin domain; DMARC confirms alignment between authenticated domain and `From:` header domain. This scenario validates the correctness of the lab configuration and establishes a reference baseline.

8.7.2 Scenario B

In this scenario a direct spoofing attack is shown with: envelope sender `random@evil.lab`, Header `From: ceo@legit.lab` and no DKIM signature.
As represented in Figure 8.10, the SMTP session terminates with `550 5.7.1 rejeceted by DMARC policy for legit.lab`. This response demonstrates that the message will never be accepted in the receiver's queue, thanks to the DMARC policies correctly configured.
Receiver logs in Figure 8.11 shows: `SPF(mailfrom): evil.lab none`, `legit.lab fail`, `milter-reject ... rejected by DMARC policy for legit.lab`. These lines indicate that SPF does not authenticate legit.lab, DKIM is absent and DMARC alignment fails; for these reasons the policy `p=reject` triggers enforcement.
This scenario demonstrates the critical enforcement property of DMARC. Even though SPF is evaluated and DKIM is checked, neither alone determines delivery. Instead DMARC extracts the domain from the visible `From:` header and verifies alignment with

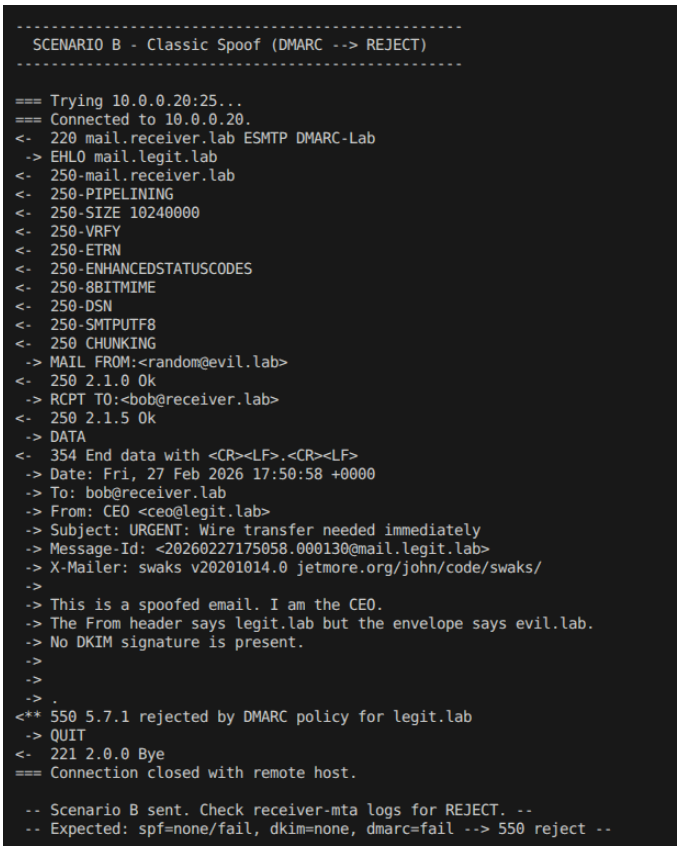

Figure 8.10: Scenario B SMTP output

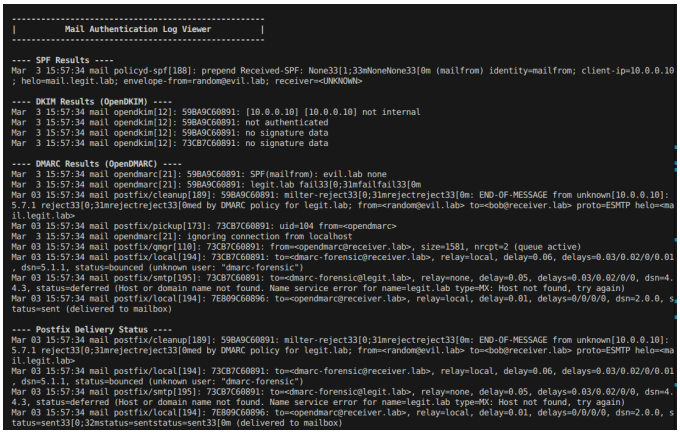

Figure 8.11: Scenario B logs

SPF/DKIM authenticated domains. Then, it applies the domain owner's published policy, thus the rejection occurs at SMTP time, which is the strongest possible mitigation strategy.

8.7.3 Scenario C

This scenario represents the most critical technical proof of the study: an SPF "`Pass`" is entirely insufficient for identity protection without DMARC alignment validation.

```
------------------------------------------------
  SCENARIO C - Partial Bypass (SPF pass,
               DMARC fail due to alignment)
------------------------------------------------

=== Trying 10.0.0.20:25...
=== Connected to 10.0.0.20.
<-  220 mail.receiver.lab ESMTP DMARC-Lab
 -> EHLO mail.legit.lab
<-  250-mail.receiver.lab
<-  250-PIPELINING
<-  250-SIZE 10240000
<-  250-VRFY
<-  250-ETRN
<-  250-ENHANCEDSTATUSCODES
<-  250-8BITMIME
<-  250-DSN
<-  250-SMTPUTF8
<-  250 CHUNKING
 -> MAIL FROM:<mailer@thirdparty.lab>
<-  250 2.1.0 Ok
 -> RCPT TO:<bob@receiver.lab>
<-  250 2.1.5 Ok
 -> DATA
<-  354 End data with <CR><LF>.<CR><LF>
 -> Date: Fri, 27 Feb 2026 17:53:40 +0000
 -> To: bob@receiver.lab
 -> From: CEO <ceo@legit.lab>
 -> Subject: Invoice #12345 - please review
 -> Message-Id: <20260227175340.000156@mail.legit.lab>
 -> X-Mailer: swaks v20201014.0 jetmore.org/john/code/swaks/
 ->
 -> This email has SPF=pass (envelope is thirdparty.lab, authorized IP).
 -> But the From header says legit.lab.
 -> DMARC catches the alignment mismatch and rejects.
 ->
 ->
 -> .
<** 550 5.7.1 rejected by DMARC policy for legit.lab
 -> QUIT
<-  221 2.0.0 Bye
=== Connection closed with remote host.

--- Scenario C sent. Check receiver-mta logs for alignment mismatch. ---
    Expected: spf=pass (thirdparty.lab), dkim=none, dmarc=fail
    Key point: SPF passed but for the WRONG domain!
```

Figure 8.12: Scenario C SMTP output

As shown in Figure 8.12, the DMARC daemon successfully intercepts the structural mismatch and forces a `550 5.7.1 rejection`, preventing the exploitation of the legitimate third-party relay.
The receiver logs in Figure 8.13 show that the attacker successfully bypassed initial SPF filters. The `policyd-spf` daemon, in fact, registers a valid authorization (`Pass...envelope-from=mailer@thirdparty.lab`). So, SPF authentication succeeded, but it authenticated the wrong domain (`thirdparty.lab`). Since the user visible header is spoofed as `ceo@legit.lab`, the organizational domains do not align, thus it is going to be detected by DMARC. Consequently, the milter computes a `legit.lab fail`. Finally, it was demonstrated that even if an attacker was able to obtain a "`SPF Pass`" by using authorized third-party infrastructure, the OpenDMARC milter was able to successfully invoke a rejection at the SMTP protocol level if a "`p=reject`" policy was in place.

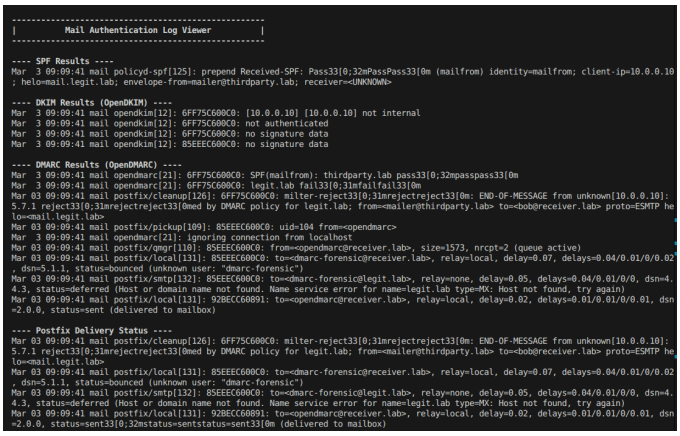

Figure 8.13: Scenario C logs

8.7.4 Scenario D

From the assessment of Scenario D, it was recognized that an organization was at critical risk if they chose to implement a "`p=none`" policy, in which case the authentication failure was logged and mail was still delivered, thereby leaving the organization open to "Man in the Mail" and CEO fraud types of attack, even with monitoring in place.

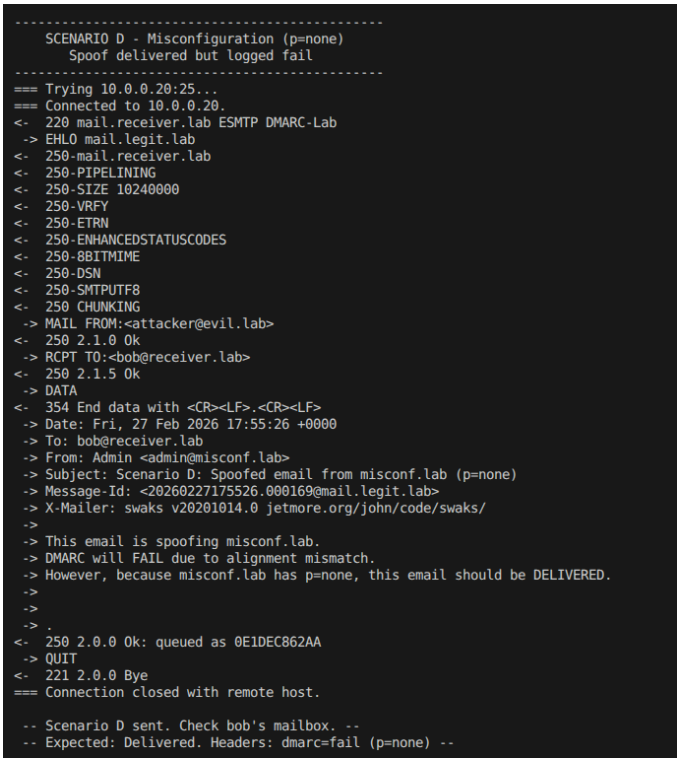

Figure 8.14: Scenario D SMTP output

Since the `misconf.lab` has a `p=none` policy, the message will be delivered in the recipient's inbox, as shown in Figure 8.14: `250 2.0.0 Ok: queued`, here no rejection occurs.

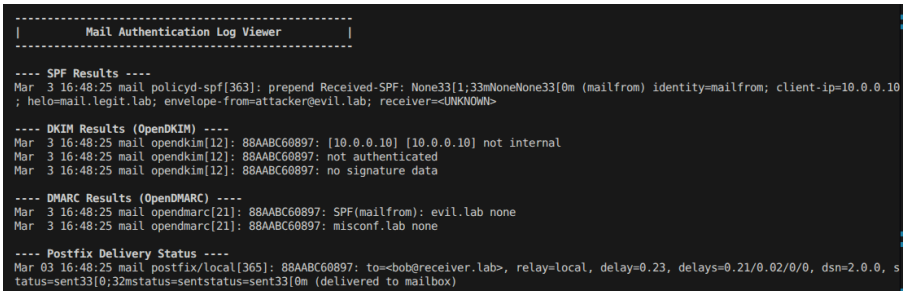

Figure 8.15: Scenario D logs

The system logs (Figure 8.15) confirm the complete absence of cryptographic integrity (`opendkim: no signature data`) and authorization (`opendmarc: SPF(mailfrom): evil.lab none`), thus the Postifx delivery status is `sent`.

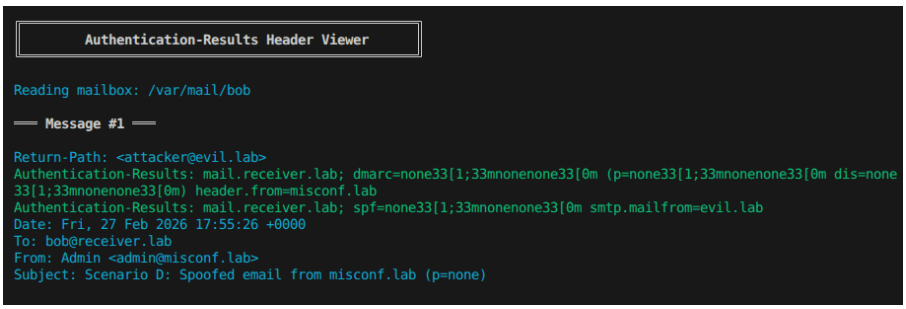

Figure 8.16: Scenario D headers

Finally, headers exposes the resulting forensic artifact. The `Authentication-Results` explicitly log `spf=none` and `dmarc=none`. Because the domain is configured with the disposition as `none`, the receiver MTA is forced to deliver the actively spoofed email (`admin@misconf.lab`) directly to the user's inbox, exposing the organization to immediate BEC or phishing risks.

8.8 Limitations of the experimental scenarios and real-world considerations

Despite the experimental lab environment effectively illustrates the fundamental operational mechanisms of SPF, DKIM, and DMARC enforcement, it remains a simplified model of actual operational infrastructures. The four scenarios outlined in the testbed essentially highlight key authentication results, including successful authenticated delivery, rejection of spoofing attempts, rejection on the basis of alignment, and monitoring-only results. However, it should be noted that a critical evaluation of these limitations remains crucial in order to effectively understand the results of the experiments.

8.8.1 Simplified mail infrastructure

In a lab, email setup is straightforward: a sending MTA, a receiving MTA, and a dedicated DNS service. But in a real-world email system, things are a lot more complex and distributed. There are many sending relays, third-party cloud email services, inbound filters, and so on, all the way down to reputation-based filters.
In a large organization, there are often third-party email service providers, cloud transactional email services, marketing automation systems, and helpdesk systems, all sending email from the same domain. In this case, SPF must specify a large number of approved sending hosts via `include` mechanism, and there might be a number of different key holders for a given DKIM setup. This is a level of complexity that is difficult to replicate in a testbed environment.
Furthermore, in real receiving infrastructures there are multiple levels of filters such as spam, malware, reputation, and behavioral filters using machine learning algorithms. In a lab scenario, acceptance of a message depends entirely on authentication results. In a real-world case, acceptance of a message is a result of a combination of authentication results, reputation of the sender, and content and heuristic filters.

8.8.2 Absence of Email Forwarding and Mailing List Transformations

Another important limitation concerns the absence of forwarding mechanisms and mailing list processing. In real email ecosystems, messages are frequently forwarded across multiple intermediate mail servers, which may modify headers, alter message bodies, or change envelope information.
Such transformations can affect authentication results in several ways.Since the forwarding server becomes the new sending host without being authorized in the original domain's SPF record, it often happens that SPF fails during the forwarding phase. DKIM signatures may break if the message body is modified by mailing lists that append footers or reformat message content. DMARC alignment may therefore fail even for legitimate messages, creating operational challenges for domain owners.
These scenarios intentionally avoid such transformations in order to isolate the core authentication logic. However, real-world deployments must often adopt mitigation mechanisms such as Authenticated Received Chain (ARC) to preserve authentication results across forwarding chains. The absence of ARC or similar mechanisms in the laboratory means that the scenarios do not fully reflect authentication challenges encountered in operational email ecosystems.

8.8.3 Lack of Sender Reputation and Behavioral Filtering

Modern email security systems heavily rely on sender reputation and behavioral analysis. Receiving mail servers often maintain reputation scores based on historical sending behavior, complaint rates, spam trap hits, and sending volume patterns. In the experimental testbed, all SMTP sessions originate from controlled IP addresses within a closed Docker network. As a result, the scenarios do not incorporate reputation-based filtering or rate-limiting mechanisms that are widely used by large email providers. In practice, spoofing

detection frequently depends on the interaction between authentication signals and reputation metrics rather than on authentication alone. Additionally, many modern email security gateways apply machine-learning models that analyze message structure, linguistic patterns, and historical communication relationships between senders and recipients. These systems can detect sophisticated phishing campaigns even when authentication checks pass successfully. Such behavioral detection mechanisms fall outside the scope of the experimental laboratory.

8.8.4 Absence of Large-Scale DMARC Monitoring Infrastructure

In the developed scenarios, evaluation of the DMARC policy is considered at the receiving server, as well as the logging of the authentication results. However, in the real world, the effectiveness of DMARC is dependent on the constant monitoring of aggregate reports (RUA) and forensic reports (RUF) that the mail server sends out. RUA and RUF allow domain owners to monitor where authentication failures are happening, as well as where unauthorized senders are trying to send emails that impersonate their domain. The test setup that is being utilized in this laboratory is more dependent on the local authentication results, as opposed to the entire system that needs to be managed in the real world to effectively implement DMARC.

8.8.5 Limited adversarial model

The laboratory setup for the attack scenarios is quite simple, it involves spoofing attacks such as domain misalignment or the absence of authentication signatures. They can be used for illustrating how the protocol works, but real-world attackers may use even more sophisticated attacks.
For example, attackers may use lookalike domains, where they can evade the DMARC protocol by taking advantage of slight differences between characters, such as homoglyphs, which can be used for sending spam mail. They may also gain unauthorized access to legitimate accounts, which can be used for sending authenticated mail that passes SPF, DKIM, and DMARC. All of this highlights the fact that the DMARC protocol can only be used for protecting against domain impersonation attacks, while other attacks may not be completely covered by the protocol. To protect against such attacks, other measures may be necessary, such as domain similarity analysis, behavioral analysis, and awareness training.

8.8.6 Implications for experimental interpretation

These constraints do not disprove the findings; rather, they indicate the environment to which the findings are applicable. In the lab environment, the basic mechanics behind domain-based email authentication are simplified to the essential components necessary to understand how SPF, DKIM, and DMARC all come together to determine whether or not a message passes through or if it is blocked.
However, it should be understood that the simplified environments should not be taken as actual simulations of the real world of email systems. In the real world, the actual results of email authentication combine the outcomes of many other factors, both technological,

organizational, and behavioral, that contribute to the actual effectiveness of anti-spoofing measures. Recognizing such limitations is essential to fully understanding the laboratory results and the role that the mechanisms described in the experiments play in the actual world of modern email systems' security infrastructures.

8.9 Security Implication and Conclusions

Finally, the experimental results validate several theoretical claims discussed in earlier chapters. One of the main points to point at is the distinction between authentication and identity assurance. This is perhaps best illustrated in the case of Scenario C. While SPF authenticated the envelope domain (`thirdparty.lab`), the visible identity presented to the user (`ceo@legit.lab`) remained unauthenticated and misaligned. Without DMARC, this message might have been delivered anyway, even though SPF was successful, thus creating a false sense of authenticity. This highlights the importance of DMARC and its role in preventing messages with incorrect authentication results. This underlines the fact that DMARC's alignment requirement isn't optional but rather a fundamental requirement to prevent impersonation.

Scenario D, where `p=none` is configured, reveals an important real-world phenomenon: domains may appear to have implemented DMARC, yet remain fully susceptible to spoofing. The Authentication-Results header records `dmarc=none`, and logs confirm evaluation failure; however, the message is delivered because the published policy explicitly disables enforcement. This illustrates a separation between telemetry generation and preventive capability. Monitoring-only deployments provide visibility but do not modify message handling behavior. Consequently, organizations that remain indefinitely in `p=none` mode benefit from reporting but do not meaningfully reduce spoofing exposure.

From a governance perspective, the findings underscore the importance of continuous monitoring. Even in rejection scenarios, the receiver logs preserve structured evidence of SPF evaluation, DKIM verification status, DMARC alignment decisions, and final delivery outcomes. These logs then represent a concrete record of the attempt to spoof. In a production environment, these kind of statistics can feed SIEM pipelines, generate alerts, and support incident response workflows. Without active log analysis, however, these signals remain dormant and offer no protective value. Thus, enforcement and monitoring must be understood as complementary components of an effective defense strategy.

Finally, these findings have broader strategic relevance. Email security is often perceived as a binary state (configured or not configured). The experimental evidence contradicts this view. Security emerges from the interaction between protocol semantics, DNS publication accuracy, receiver behavior, and policy enforcement configuration; misalignment between these layers results in exploitable gaps.

These implications substantiate the central thesis of this work: effective spoofing prevention requires not only technical deployment but deliberate enforcement and ongoing monitoring.

Chapter 9

Conclusion

Email is one of the key ways we interact with one another, both on the job, within government, and in personal lives. However, Simple Mail Transfer Protocol (SMTP) was developed at a time when robust authentication and defense against attacks were not major concerns, so there is room for impersonation attacks, which utilize the system's lack of verification of the sender's identity.

In this thesis, the goal was to examine the problem of preventing email spoofing from different angles, both technically, operationally, and legally, with the use of domain-based authentication, with special emphasis on the use of DMARC (Domain-based Message Authentication, Reporting, and Conformance). The study shows how SPF and DKIM can be used, but how, on their own, they do not effectively stop domain spoofing, while DMARC provides the essential feature of identifier alignment. In terms of protocol, the thesis examines how SPF, DKIM and DMARC interact, including how decisions about authentication come from DNS TXT records, cryptographic signatures, and alignment. Meanwhile, the findings demonstrate how decisions about enforcement (rejection, quarantining, or monitoring) by domains using DMARC are crucial, including how domains with permissive policies ("`p=none`") remain susceptible to impersonation even if there is the existence of authentication records.

To complement the theoretical analysis, this research introduced two experimental contributions.

The first contribution is a risk assessment tool for domain impersonation that assesses the risks of spoofing by analyzing DNS configuration, authentication posture, and syntactic characteristics of domain names. The tool automatically checks for the presence of DMARC, SPF and DKIM records, MX records and typosquatting signals combining these indicators into a weighted scoring model that outputs an interpretable risk score. The tool has been designed to be easy to understand, with explicit "reason tags" that allow the user to see exactly what factors contribute to the risk level being higher or lower.

The second contribution is a containerized experimental lab designed to replicate real-world email authentication scenarios. The laboratory setup has been designed to replicate a basic email ecosystem, including sender and receiver MTAs, a DNS service, as well as authentication components such as OpenDKIM and OpenDMARC. The setup has four

different scenarios that test different aspects of email authentication, including: legitimate authenticated emails, spoofing blocked by DMARC enforcement, alignment-based spoofing attempts detected despite SPF success, and spoofing successfully delivered because of a permissive DMARC policy.

Results highlighted a number of important security lessons learned. The key takeaways were that SPF alone was not enough to properly secure identity since it checks against the envelope domain rather than what the user sees. DKIM has strong integrity with its cryptography but fails to prevent impersonation without proper alignment with what the user sees. Finally, it was clear that DMARC was the key to making all of these authentication mechanisms actionable at the receiving server. In other words, it was clear that with strict policies like `p=reject` in place in the lab scenarios, it was almost impossible to spoof messages at SMTP time, but monitoring-only policies failed to offer similar security benefits.

In addition to the technical part, this thesis also examined the changes in the legislation and laws regarding email security. The GDPR, the ePrivacy Directive, and the NIS2 Directive are all moving into one direction: that companies properly implement technical and organizational security measures. Although no specific authentication protocols are mandated in any of these directives, the widespread use of SPF, DKIM, and DMARC points toward a standardization that is likely becoming the norm for reasonable security in email systems. Authentication records, DMARC policies, and email headers are all potentially important pieces of evidence in legal cases regarding spoofing-related fraud and in determining if a company has taken reasonable care in protecting its communications infrastructure.

Regarding longer-term ideas, this work could expand in many directions. First, large-scale measurement studies could be conducted to understand the usage of DMARC in the real world across different domains. Second, the domain risk analysis tool could be extended to incorporate other indicators such as certificate transparency, passive DNS, and reputation-based indicators. Finally, the laboratory setup could be improved by incorporating the ingestion of DMARC reports as well as SIEM systems to resemble the real world cases.

In conclusion, this thesis highlights the central role of DMARC as a structural mechanism for restoring trust in email sender identity. While no solution can ever completely solve the problem of social engineering, the likelihood of large-scale spoofing can be greatly diminished with the proper use of domain-based authentication. Since more reliance is put on the use of email for critical communications, the strengthening of authentication systems will continue to play an integral role in the overall concept of cyber trust.

Appendix A

User Manual

This manual provides simple instructions for installing, starting, and using the two developed tools. Let's start with a step-by-step guide to deploying and running the Domain Analyzer CLI tool.

A.1 Domain Analyzer Tool

1. **Prerequisites** - before using the tool, ensure that your system has Python 3.8 or higher and `pip` installed;

2. **Installation and deployment** - once you downloaded the source code:

   (a) Create a virtual environment in your working folder to avoid dependency conflicts, run:

   ```
   python3 -m venv .venv
   source .venv/bin/activate
   ```

   (b) Install dependencies manually using `pip`:

   ```
   pip install dnspython tldextract idna python-Levenshtein requests
   ```

3. **Running the tool** - the tool is executed via the `main.py`

   - **Analyzing a Single Domain** - to analyze a specific domain, pass it as the default argument:

     ```
     python main.py example.com
     ```

   - **Analyzing Multiple Domains** - if you have a large list of domains, place them in a text file (one per line) and use the `-i` or `--input` flag:

     ```
     python main.py -i domains_list.txt
     ```

   - **Adding Brands for Typosquatting Detection** - to enable brand-spoofing and typosquatting detection heuristics, provide a text file containing the known legitimate brands using the `--brands` flag:

     ```
     python main.py --brands brands.txt -i domains_list.txt
     ```

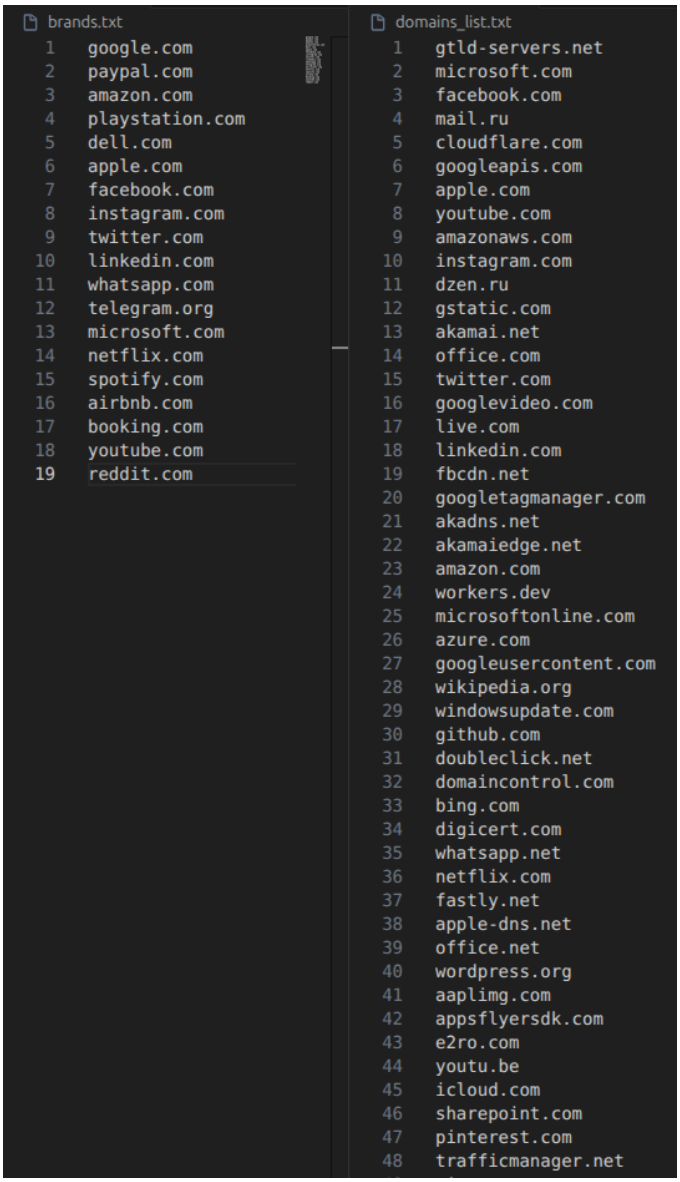

Figure A.1: Brands and domain list files example

- **Customizing DNS and Threading**
  - Custom DNS Resolvers: by default, the system's DNS is used. To use custom resolvers (e.g., Cloudflare, Google), specify them with `--dns`:

    ```
    python main.py example.com --dns 1.1.1.1 8.8.8.8
    ```

  - Timeout: it is possible to adjust the DNS query timeout (default is 2.5 seconds) using `--timeout`:

    ```
    python main.py example.com --timeout 5.0
    ```

  - Concurrency / Thread Pool: the tool uses parallel threads for rapid batch analysis, it also allows to maximize network throughput by increasing

workers (default is 50):

```
python main.py -i domains_list.txt --workers 100
```

- **Specific Probes** - DKIM Selector Probing allows probing for common DKIM selectors (such as `google`, `default`, `s1`, `mail`) using the `--dkim-probe` flag. This will issue additional DNS queries.

```
python main.py example.com --dkim-probe
```

4. **Output formats and Reports** - tool supports three output formats, controlled by the `--format` flag:

- ASCII Table (Default) - prints a fixed-width table directly to standard output.

```
python main.py example.com --format table
```

Input	eTLD+1	Score	Level	Reason
example.com	example.com	15	LOW	no_dmarc_reporting_rua,no_spf

Table A.1: Table output example

- JSON - serializes the result structures to JSON format.

```
python main.py example.com --format json
```

Listing 22: JSON output example

```
        [
          {
                    "domain_input": "example1.com",
                    "etld_plus_one": "example1.com",
                    "risk_score": 30,
                    "risk_level": "MEDIUM",
                    "reasons": [
              "no_mx",
              "no_dmarc",
              "no_spf"
                    ],
                    "syntax": {
                     "input_domain": "example1.com",
      "fqdn_provided": false,
      "registrable_domain": "example1.com",
      "subdomain": "",
      "idn": false,
      "mixed_scripts": false,
      "suspicious_tokens": [],
      "brand_in_subdomain": [],
      "typo_matches": []
    },
    "mx": {
      "present": false,
      "exchanges": []
```

```
    },
    "dmarc": {
      "present": false,
      "raw_records": [],
      "policy": null,
      "pct": null,
      "rua_present": null,
      "adkim": null,
      "aspf": null,
      "ambiguous": false,
      "parse_error": null
    },
    "spf": {
      "present": false,
      "raw_records": [],
      "spf_records_only": [],
      "multiple": false,
      "all_qualifier": null,
      "weak_all": false,
      "parse_error": null
    },
    "dkim": {
      "probed": false,
      "found_selectors": [],
      "notes": "DKIM probing disabled (no email context)."
    },
    "timestamp_utc": "2026-02-16T18:39:25Z"
  }
]
```


- CSV- flattens the parsed results into a single-row-per-domain CSV file.

```
python main.py example.com --format csv
```

A.2 DMARC container laboratory

Now let's see how the virtual laboratory developed for the practical demonstration of SPF, DKIM, and DMARC protocols can be installed and run.

1. **Software prerequisites** - ensure the following tools are installed: Docker, Docker Compose, OpenSSL

2. **Open the terminal and navigate to the folder where the project is saved**:

   ```
   cd /path/to/project
   ```

3. **Run the initial setup** - this operation needs to be done once before starting the lab, it creates DKIM cryptographic keys and files describing DNS zones. Run command:

   ```
   chmod +x setup.sh
   ./setup.sh
   ```

4. **Initializing the laboratory**

   - Start the containers with:

     ```
     docker compose up --build d
     ```

   - Wait for initialization with:

     ```
     sleep 15
     ```

     recommended to wait for make virtual machines fully accessible

5. **Run the scenarios** - run the commands one at a time or only the desired ones:

   ```
   docker compose exec sender-mta bash /scripts/scenario-a.sh
   docker compose exec sender-mta bash /scripts/scenario-b.sh
   docker compose exec sender-mta bash /scripts/scenario-c.sh
   docker compose exec sender-mta bash /scripts/scenario-d.sh
   ```

6. **Inspecting and Verifying the Results** - to understand the outcomes of the operations inspect the logs with:

   ```
   docker compose exec receiver-mta bash /scripts/check-logs.sh
   ```

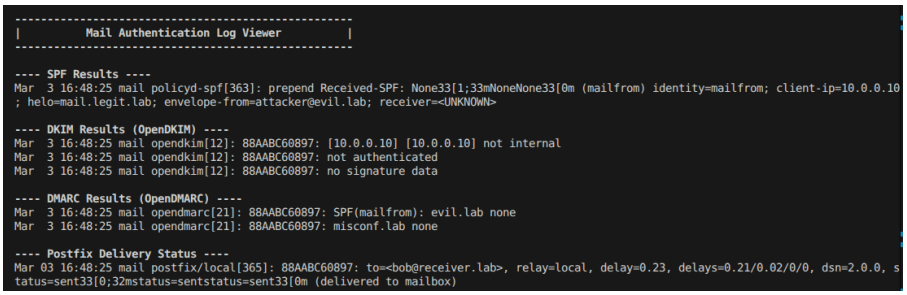

Figure A.2: Scenario D logs example

Analyze headers saved in delivered mail running:

```
docker compose exec receiver-mta bash /scripts/check-headers.sh
```

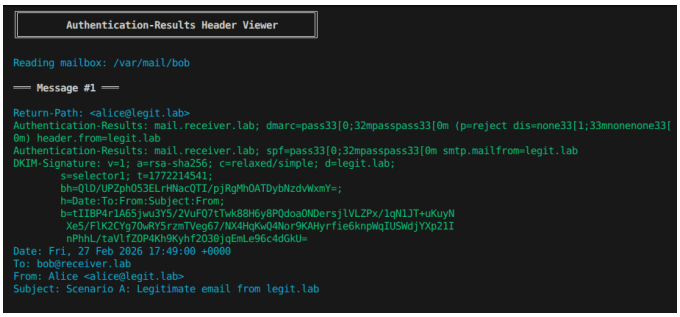

Figure A.3: Scenario A headers example

7. **Generate and Export the Global Forensic Report** - report generation showing the DNS tree, passed emails, rejected ones, and raw logs. Run:

   ```
   docker compose exec receiver-mta bash /scripts/forensic-report.sh
   ```

8. **Suspension and Teardown** - at the end of the tests, shut down the containers and free network resources so as not to overload the computer by running:

   ```
   docker compose down -v
   ```

9. **Clear logs to perform another run** - to execute another scenario in a clean way, it is possible to clear the logs in order to have a more readable logs and headers output:

   ```
   ./path_to_the_project/scripts/clear-logs.sh
   ```

Appendix B

Programmer Manual

This appendix details the architectural design and code structure of the two tools. It provides software engineers and researchers with the information necessary to understand the codebase and to implement future developments, heuristics, or data-collection models. Let's start with the Domain Analyzer CLI.

B.1 Domain Analyzer Tool

- **Architectural Overview** - the project relies on a modular architecture written in Python. It aggregates data from two loosely coupled pipelines (Syntax Analysis and DNS/Email Posture) and feeds the aggregated evidence into a static scoring engine. The entire codebase is parallelized vertically per domain. A multi-threaded orchestrator pushes each input domain through the entire pipeline individually. This ensures high throughput without complex inter-process communication algorithms. General architecture is represented in Figure B.1.

- **Core modules description**

  - `models.py` (Data Model Layer): Provides rigid structure utilizing Python's `@dataclass`. These classes (`SyntaxInfo`, `MxInfo`, `DmarcInfo`, `SpfInfo`, `DkimInfo`, and `Result`) define the contract between different modules, ensuring data immutability and precise JSON serialization.

  - `domain_utils.py` (Input normalizers): Pre-processes the user's input, stripping HTTP/HTTPS protocols, resolving paths, lowercasing tokens, grouping by `tldextract` (eTLD+1), and detecting/decoding Punycode (`xn-`) using `idna`.

  - `dns_client.py` (DNS Wrapper): Responsible for executing DNS network operations. It wraps `dnspython`, creating custom threaded `Resolver` instances, executing MX and TXT queries, concatenating extensive multi-string Resource Records, and silencing runtime failures (like timeouts or NXDOMAIN) to guarantee the workflow doesn't crash on network unreliability.

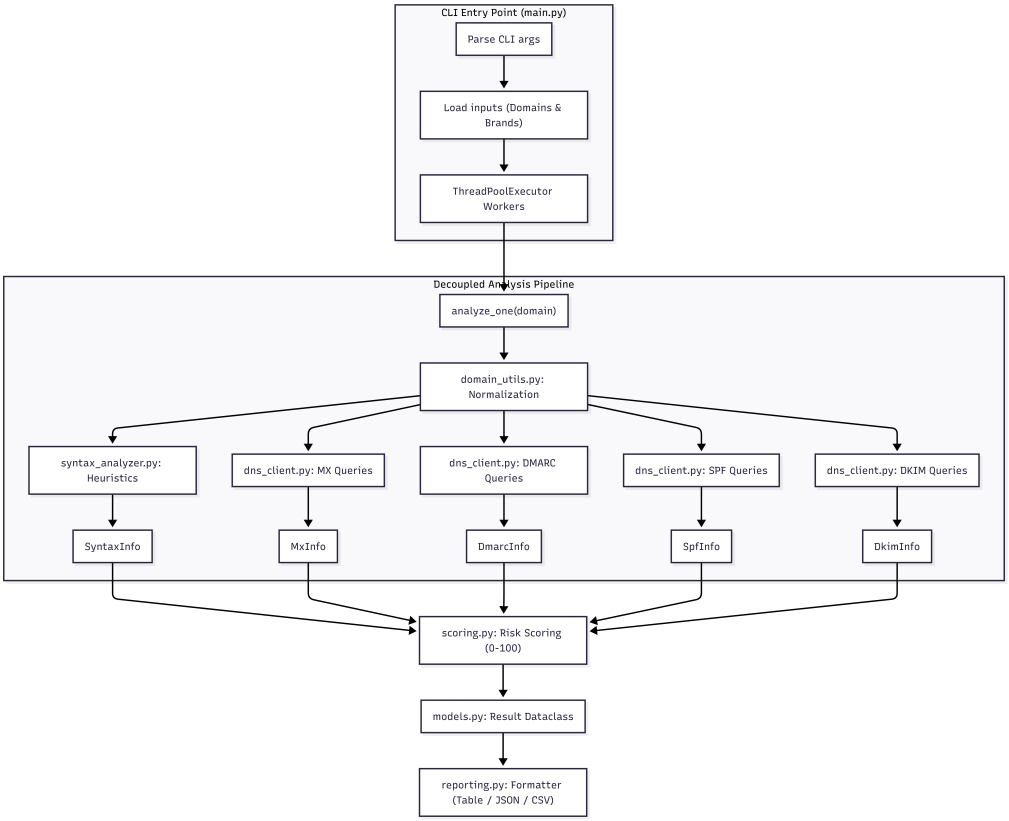

Figure B.1: General Architecture

  - `syntax_analyzer.py` (syntactic engine): Encapsulates static evaluations of the strings, including Mixed-Script spoofing (via Unicode ranges), keyword-based tokenization (e.g. tracking "login" or "billing" substrings), and brand impersonation computing the Damerau-Levenshtein distance against protected brands.
  - `email_posture.py` (record parsers): Digests the exact syntax of the retrieved DNS text records, strictly defining if DMARC syntax is ambiguous or valid, extracting SPF mechanisms with regex, and detecting explicit RFC violations.
  - `scoring.py` (risk algorithm): Applies a predefined set of static integer weights against flags reported by the Syntactic and Email Posture engines. Outputs a deterministic `[0, 100]` score alongside a risk tier string (`LOW`, `MEDIUM`, `HIGH`).
  - `reporting.py`: Exports the array of Result structures to screen or disk.

- **Future Developments**: The modular logic guarantees that adding new features rarely requires refactoring the core pipeline. Below it is descripted how a programmer can approach to extend the analyzer.

  - Adding a new DNS Analysis step (such as BIMI or DNSSEC):

1. Create the model: in `models.py`, it is possible to define a new dataclass such as `BimiInfo`. Be sure to nest this new class as an attribute inside the primary `Result` dataclass.
2. Implement DNS resolution: utilize the wrappers in `dns_client.py` to draft targeted lookup logic.
3. Parse and Analyze Records: inside `email_posture.py` (or a newly created file), write the regex or parsing logic needed to analyze the specific components of the pulled record. Return an instance of `BimiInfo`.
4. Integrate into the entry point: edit the `analyze_one` method inside `main.py` so that it spawns the new DNS lookup, parses it using your logic, and injects the `BimiInfo` into `Result`.

– Adding a new output formatter: if integration with a SIEM framework or dashboard requires a data export format (e.g., XML, YAML) it could be possible to implement a dedicated function in `reporting.py`(e.g., "`write_xml(results, path)`). Under `main.py`, it is possible to parse the new extra command-line flag inside `argparse` expanding the choices for the `-format` argument and finally wire the dispatcher at the end of the main to call your new formatter when the given format is selected.

– Future research ideas - future academic or commercial improvements to this project could explore:

  * Asynchronous I/O implementation: transitioning from the OS-thread dependent `ThreadPoolExecutor` context to strict `asyncio`, utilizing `aiohttp` for Common Crawl requests and `aiodns` for DNS lookup. This move would decrease memory footprint for massive multi-million-domain batch processing.
  * Machine Learning integration: replacing the integer-based weighted summation mechanism located within `scoring.py` with an offline-trained predictive classification model, trained against datasets composed by aggregating confirmed phishing domains versus benign domains from the top lists.

B.2 DMARC container laboratory

Now let's analyze the technical architecture in which the DMARC laboratory is embedded and operates. The tool's architecture is built on containerization principles, relying entirely on Docker and Docker Compose, which allow the testing environment to be abstracted from the user's host machine, protecting it from potentially polluting network or DNS interference.
The system runs into an isolated internal network, named `mailnet` mapped to the `10.0.0.0/24` subnet, containing three communicating entities:

1. **DNS Container** (IP: `10.0.0.2`): it is an authoritative DNS daemon implemented using `BIND9`. It constitutes the backbone of trust for the mail infrastructure and

maintains both forward and inverse IP DNS zones. It hosts within its TXT records the SPF definitions, DMARC policies, and publishes the public RSA components used to test DKIM keys (e.g., `selector1`) in plain text.

2. **Sender-MTA Container** (IP: `10.0.0.10`): this element plays the role of the "sending actor", employing a Postfix mailer configured solely to generate outbound connections, acting as a textual client towards the target domain. In Signing mode, the OpenDKIM daemon systematically and in real time applies signatures to this instance. Regarding the injection of email payloads (and specific Envelope vs. Header configurations), the command-line tool `swaks` is used, allowing modular construction of the SMTP envelope without passing through standard mail channels.

3. **Receiver-MTA Container** (IP: `10.0.0.20`): it symbolizes the Email Service Provider of the potential victim, impacted by both legitimate transmissions and spoofed attack vectors. It is also based on Postfix, it integrates a strict SMTP check chain (Milter Chain pipeline) with:

    - A policy daemon written in python (`policyd-spf`) executes recipient restrictions, cross-referencing the socket IP address with the `MAIL FROM`. It applies the first upstream decisions.
    - An OpenDKIM manager operating in Verify mode decodes any base64 to calculate email integrity and it will inject formal headers (`dkim=pass/fail`).
    - The OpenDMARC daemon, which is the last bastion responsible for extrapolating the frame evaluated by the first filters, it also queries domain alignment lookups to assess conceptual correctness based on RFCs, discards or accepts, and records forensic metadata.

Analizying the system at low level, it is possible to notice that it is composed of:

- **Security Management and lifecycle bootstrap** to avoid deprecated practices like hardcoding cryptographic keys, Private Keys are injected using a Run/Setup Time approach. The script interface `setup.sh` performs:

    1. invokes the `openssl` suite (or related tools);
    2. dynamically produces and moves raw cryptographic keys and TXT versions;
    3. updates the `.zone.template` files, compiling them and placing them inside `dns/zones/`, ready for BIND9 engagement before container image creation.

- **Filter Call Order** (Postfix Milter): the `main.cf` setup file at receiver side plays a decisive role in evaluating the experiment. Pipes to milters occur in a cascade: `smtpd_milters = inet:localhost:8891, inet:localhost:8893`. OpenDKIM (port 8891) is read, evaluated and resolved with priority over OpenDMARC (port 8893), which structurally depends on the parameters injected by the DKIM daemon. SPF is not a native milter on this stack and is handled in `smtpd_recipient_restrictions` by calling an external service `check_policy_service`.

For future improvements to the project, some architectural ideas:

1. **ARC Protocol Integration** (Authenticated Received Chain) - in real world systems, domains often face DMARC failures generated by "Forwarding" caused by mailing lists or aliases. This practice compromises the authorized IP origin and sometimes the base payload, rendering original DKIM signatures useless.

    - Development: it could be an idea creating an intermediary virtual machine/-container configured for mailing-list forwarding, inserting the OpenARC daemon into the stack.
    - Effects: it will allow the next study to demonstrate how forwarded cryptographic "seals" can preserve and extend trust despite manipulating headers during inter-MTA hops.

2. **Telemetry collection and automatic processing of RUA/RUF analyses**: at the moment, forensic reports on the Receiver are managed by textually interfacing with logs generated by bash checks. An evolution would involve introducing SIEM infrastructures by leveraging frameworks such as `parsedmarc` that transform XML RUF and RUA data into archivable documents. This infrastructure could be combined with siem tools such as Elasticsearch, Logstash, and Kibana, allowing a programmer to read and study advanced metric dashboards instead of standard raw logs, offering immediate graphical indication of attack attempts across various scenarios.

3. **Evolution of tools with web-based interface**: since the current scenarios pass through native docker interaction, it could be an idea to replace this step by introducing a small app written in Express or Flask, receiving web input and injecting `POST` calls, as it would lower the test time-to-market while concurrently bringing the usability benefits of a web application (or remote API) to end-users less familiar with the command line.

4. **TLS and perimeter security protocols** - a potential step forward involves marginal transport encryption (SMTP-in-transit): demand Strict Transport Security (MTA-STS) by defining additional TXT zone files; hook the `Nginx/Apache` daemon for `HTTPS` policy file distribution, enabling instruction of Receiver machines to reject the connection itself if the sending server cannot explicitly encrypt the TLS channel without validated downgrade, providing a ruse for man-in-the-middle interceptions.

Bibliography

[1] F. B. of Investigation, «2024 Internet Crime Report», Internet Crime Complaint Center, Tech. Rep., 2024. [Online]. Available: https://www.ic3.gov/AnnualReport/Reports/2024_IC3Report.pdf.

[2] E. Z. M. Kucherawy, «Domain-based Message Authentication, Reporting, and Conformance (DMARC)», Yahoo!, Tech. Rep., 2013. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7489#autoid-2.

[3] J. Wilson, «The State of Email Trust: Global DMARC Adoption Trends in Q2 2025», *Fortra Email Security*, 2025. [Online]. Available: https://emailsecurity.fortra.com/blog/dmarc-adoption-trends-q2-2025.

[4] M. Tanksale, «Cops save firm's money lost in email spoofing fraud», 2025. [Online]. Available: https://timesofindia.indiatimes.com/city/pune/cops-save-firms-money-lost-in-email-spoofing-fraud/articleshow/123312588.cms.

[5] Reuters, «Austria's FACC, hit by cyber fraud, fires CEO», 2016. [Online]. Available: https://www.reuters.com/article/us-facc-ceo-idUSKCN0YG0ZF/.

[6] D. J. C. Klensin, *Simple Mail Transfer Protocol*, RFC 5321, Oct. 2008. DOI: 10.17487/RFC5321. [Online]. Available: https://www.rfc-editor.org/info/rfc5321.

[7] Krebs on Security, «Tech Firm Ubiquiti Suffers 46M Cyberheist», 2015. [Online]. Available: https://krebsonsecurity.com/2015/08/tech-firm-ubiquiti-suffers-46m-cyberheist/ (visited on 01/29/2026).

[8] S. Kitterman, *Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1*, RFC 7208, Apr. 2014. DOI: 10.17487/RFC7208. [Online]. Available: https://www.rfc-editor.org/info/rfc7208.

[9] M. Kucherawy, D. Crocker, and T. Hansen, *DomainKeys Identified Mail (DKIM) Signatures*, RFC 6376, Sep. 2011. DOI: 10.17487/RFC6376. [Online]. Available: https://www.rfc-editor.org/info/rfc6376.

[10] M. Kucherawy, *Message Header Field for Indicating Message Authentication Status*, RFC 8601, May 2019. DOI: 10.17487/RFC8601. [Online]. Available: https://www.rfc-editor.org/info/rfc8601.

[11] K. Shen, C. Wang, M. Guo, *et al.*, «Weak Links in Authentication Chains: A Large-scale Analysis of Email Sender Spoofing Attacks», *USENIX Association 2021*, 2020. [Online]. Available: https://personal.utdallas.edu/~shao/papers/shen_usenix21.pdf.

[12] P. Dal Checco, «DMARC Compliance Spoofing», 2025. [Online]. Available: https://www.dalchecco.it/dmarcforensics-dmarc-compliance-spoofing/.

[13] C. Beaman and H. Isah, «Anomaly Detection in Emails using Machine Learning and Header Information», *arXiv.org*, 2022. [Online]. Available: https://arxiv.org/pdf/2203.10408.

[14] C. e. a. Rossow, «Prudent Practices for Designing Malware Experiments», *IEEE Security & Privacy*, 2012. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6234405.

[15] G. Ho, A. Sharma, R. Anderson, *et al.*, «Detecting Credential Spearphishing in Enterprise Settings», in *USENIX Security Symposium*, 2016.

[16] S. R. Bauskar, C. R. Madhavaram, E. P. Galla, J. R. Sunkara, and H. K. Gollangi, «AI-Driven Phishing Email Detection: Leveraging Big Data Analytics for Enhanced Cybersecurity», *Library Progress International*, 2024. [Online]. Available: https://bpasjournals.com/library-science/index.php/journal/article/view/1772.

[17] R. Dedenok, «How to repair DMARC», *kaspersky.com*, 2020. [Online]. Available: https://www.kaspersky.com/blog/how-to-cure-dmarc/36787/.

[18] N. Sharma and A. Kumar, «Deep Learning for Stylometry and Authorship Attribution: a Review of Literature», *International Journal for Research in Applied Science and Engineering Technology*, 2024. [Online]. Available: https://www.ijraset.com/research-paper/deep-learning-for-stylometry-and-authorship-attribution.

[19] P. Juola, «Jgaap: A system for comparative evaluation of authorship attribution», in *Journal of the Chicago Colloquium on Digital Humanities and Computer Science*, 2009.

[20] «How SIEM Improves Phishing Detection and Prevention», *SearchInform*, [Online]. Available: https://searchinform.com/articles/cybersecurity/measures/siem/use-cases/phishing-detection/.

[21] Splunk Inc., *Splunk enterprise security documentation*, Technical documentation, 2023. [Online]. Available: https://docs.splunk.com/Documentation/ESCU.

[22] Elastic NV, *Elastic security documentation*, Technical documentation, 2023. [Online]. Available: https://www.elastic.co/docs/solutions/security.

[23] B. Slavin, «8 Reasons to choose a DMARC report analyzer tool with real-time dashboards and alerts», *DuoCircle*, 2025. [Online]. Available: https://www.duocircle.com/dmarc/8-reasons-choose-dmarc-report-analyzer-real-time-dashboards-alerts.

[24] M. F. Hinarejos, J.-L. Ferrer-Gomila, and L. Huguet-Rotger, «A solution for secure certified electronic mail using blockchain as a secure message board», *IEEE Access*, vol. 7, pp. 31 330–31 341, 2019. DOI: 10.1109/ACCESS.2019.2902174.

[25] G. J.C., G.-D. V., and N.-V. E. et al., *Replacing email protocols with blockchain-based smart contracts*, 2020. [Online]. Available: https://doi.org/10.1007/s10586-020-03128-9.

[26] N. Gerda, «Charges Dropped in 67 Criminal Cases Due to Sheriff Evidence Mishandling», *VOICE of OC*, 2021. [Online]. Available: https://voiceofoc.org/2021/01/charges-dropped-in-67-criminal-cases-due-to-sheriff-evidence-mishandling-da-reveals/.

[27] D. Propper, «Colorado crime lab analyst faces over 100 charges for mishandling and altering reports», *New York Post*, 2025. [Online]. Available: https://nypost.com/2025/01/23/us-news/yvonne-missy-woods-charged-after-alleged-crime-lab-misconduct/.

[28] B. Wolford, «How does the GDPR affect email?», *GDPR.eu*, 2018. [Online]. Available: https://gdpr.eu/email-encryption/.

[29] «General Data Protection Regulation, Article 32», [Online]. Available: https://gdpr.eu/article-32-security-of-processing/.

[30] E. Union, «General Data Protection Regulation, Article 82», *GDPR.eu*, [Online]. Available: https://gdpr.eu/article-82-data-subjects-right-to-compensation-and-liability/.

[31] «ePrivacy Regulation - article 16», [Online]. Available: https://eprivacy-regulation.org/articles/chapter-iii/article-16-eprivacy-regulation-unsolicited-and-direct-marketing-communications.

[32] A. per la Cybersicurezza Nazionale, «Framework di Autenticazione per la Posta Elettronica», *ACN*, [Online]. Available: https://www.acn.gov.it/portale/documents/20119/1008330/Framework+di+Autenticazione+per+la+Posta+Elettronica.pdf/ac6e9bbe-247e-3fe2-85fa-997e6bd0809c?t=1755863999953.

[33] E. Union, *Directive (EU) 2022/2555 of the European Parliament and of the Council of 14 December 2022 on measures for a high common level of cybersecurity across the Union (NIS2 Directive)*, https://eur-lex.europa.eu/eli/dir/2022/2555/oj, Official Journal of the European Union, L 333, 27.12.2022, p. 80–152, 2022.

[34] E. Union, *Article 21 - NIS2 Directive*, Official Journal of the European Union, L 333, 27.12.2022, p. 80–152, 2022. [Online]. Available: https://www.nis2-info.eu/article-21-cybersecurity-risk-management-measures/.

[35] «Manipulated invoice via hacked e-mail: Those who do not encrypt are left holding the bag», *2b-Advice*, 2025. [Online]. Available: https://2b-advice.com/en/2025/03/07/manipulated-invoice-via-hacked-e-mail-those-who-do-not-encrypt-remain-liable-for-damages/#.

[36] C. J. Wilson, «Email Fraud and Third-Party Hacks: Lessons From Recent Case Law», *McKercher*, 2025. [Online]. Available: https://www.mckercher.ca/en/news/posts/email-fraud-and-third-party-hacks-lessons-from-recent-case-law.

[37] «General Data Protection Regulation, Article 5», [Online]. Available: https://gdpr.eu/article-5-how-to-process-personal-data/.

[38] M. Baghdasaryan, *PowerDMARC vs Suped: Choosing the Right Email Authentication Platform*, 2026. [Online]. Available: https://powerdmarc.com/suped-vs-powerdmarc/.

[39] «PowerDMARC vs MXtoolbox», *Suped*, [Online]. Available: https://www.suped.com/compare/powerdmarc-vs-mxtoolbox.

[40] M. Davis and M. Suignard, *Unicode Security Mechanisms*, 2025. [Online]. Available: https://www.unicode.org/reports/tr39/.