A Multifaceted Study on the Use of TLS and Auto-detect in Email Ecosystems

Ka Fun Tang, Che Wei Tu, Sui Ling Angela Mak, Sze Yiu Chau
Department of Information Engineering, The Chinese University of Hong Kong
{doriatang, tonytu1999, angelalamak}@link.cuhk.edu.hk, sychau@ie.cuhk.edu.hk

***Abstract*—Various email protocols, including IMAP, POP3, and SMTP, were originally designed as "plaintext" protocols without inbuilt confidentiality and integrity guarantees. To protect the communication traffic, TLS can either be used *implicitly* before the start of those email protocols, or introduced as an opportunistic upgrade in a post-hoc fashion. In order to improve user experience, many email clients nowadays provide a so-called "auto-detect" feature to automatically determine a functional set of configuration parameters for the users. In this paper, we present a multifaceted study on the security of the use of TLS and auto-detect in email clients. First, to evaluate the design and implementation of client-side TLS and auto-detect, we tested 49 email clients and uncovered various flaws that can lead to covert security downgrade and exposure of user credentials to attackers. Second, to understand whether current deployment practices adequately avoid the security traps introduced by opportunistic TLS and auto-detect, we collected and analyzed 1102 email setup guides from academic institutes across the world, and observed problems that can drive users to adopt insecure email settings. Finally, with the server addresses obtained from the setup guides, we evaluate the sever-side support for implicit and opportunistic TLS, as well as the characteristics of their certificates. Our results suggest that many users suffer from an inadvertent loss of security due to careless handling of TLS and auto-detect, and organizations in general are better off prescribing concrete and detailed manual configuration to their users.**

I. Introduction

Despite the proliferation of instant messaging apps in the last decade, emails continue to be heavily used in work and school settings. According to a recent market research, the number of global email users has almost reached 4 billion in 2023 [7]. Historically, various email protocols, such as the Internet Message Access Protocol (IMAP), the Post Office Protocol (POP), and the Simple Mail Transfer Protocol (SMTP) were designed as "plaintext" protocols that do not provide any inherent confidentiality and integrity guarantees. In order to protect the communication between an email server and an email client, including the exchange of user credentials and potentially sensitive messages, Transport Layer Security (TLS) is often used to introduce a secure communication channel for the email protocols.

There are two typical approaches of which TLS can be used with the email protocols. One approach expects the client and server to directly start with the TLS handshake first, and only after a TLS session is succesfully negotiated, then the email protocols begin and communicate inside the already established TLS channel. This approach is commonly known as "implicit TLS", because from the perspective of email protcols, they did not have to explicitly negotiate whether TLS can be used. A contrasting approach, which is sometimes known as "explicit TLS" or "opportunistic TLS", lets the plaintext email protocols start communicating first, and then attempt to negotiate a security upgrade to use TLS, the success of which depends on the capability of the client and server. Depending on the security policies being enforced, the email protocols could stay unencrypted even when opportunistic TLS is attempted, due to an active man-in-the-middle (MITM) downgrade attack [14].

To improve user experience and help users arrive at a functional configuration with respect to the potential uses of TLS as well as various server port numbers and user authentication methods, many email clients these days provide a so-called "auto-detect" feature, where the email client probes the server to determine an operative set of configuration parameters. Intuitively, design and implementation flaws in the auto-detect mechanism could covertly downgrade the security outcomes for its users, and depending on whether such intricacies are known to the users, might even lead to a false sense of security. Although the possibility of command injection in opportunistic TLS due to confusions in buffer management has been explored before [42], to the best of our knowledge, no prior work has evaluated the security implications of auto-detect mechanisms in email clients. Thus, in this paper, we present a multifaceted security evaluation of email clients and their deployments, with a focus on TLS and auto-detect.

In this study, we consider three perspectives of the email ecosystem in tandem. First, we evaluate the designs and implementations of client applications (apps). This allows us to identify flaws in the clients that can lead to inadvertent security downgrades, which could in turn nullify the protections of TLS and leak sensitive messages and user credentials. Second, with the potential security traps due to auto-detect and other design weaknesses already identified, we evaluate email setup guides that are prescribed by IT admins of academic institutes around the world, to see if they adequately instruct users on how to avoid the traps, and adequately protect the sensitive traffic of the email protocols. This allows us to observe and better understand the current deployment practices embraced by real-world production setups. Finally, using the server addresses identified from the setup guides, we evaluate the server-side support for implicit and opportunistic TLS, as well

Network and Distributed System Security (NDSS) Symposium 2025
24-28 February 2025, San Diego, CA, USA
ISBN 979-8-9894372-8-3
https://dx.doi.org/10.14722/ndss.2025.240532
www.ndss-symposium.org

as characteristics of the certificate chains used by institutional email servers. This enables us to gain insights on whether the email clients can in fact adopt stricter configuration settings, and whether their auto-detect features are actually helping or hindering users from achieving that.

Summary of findings. We found that out of the 49 email clients tested, 30 support some forms of auto-detect for the various email protocols, 4 of which force the use of auto-detect during the first-time configuration. When manual configuration is used, only 8 out of 49 clients allow opportunistic TLS and can be downgraded by an active MITM *if the user chooses opportunistic TLS*. However, when auto-detect is used, 14 out of the 30 that support it exhibit noticeable security downgrade. 6 can be downgraded by an active MITM, while the other 8 can leak credentials to even a weaker passive attacker. This is particularly worrisome as users might trust the auto-detect features without fully understanding their security implications. Additionally, even when TLS is used, we found 19 clients that accept certificate chains that should not be accepted, which can also jeopardize the communication security. On the email setup guide front, we found that many setup guides indeed instruct their users to unnecessarily use auto-detect, thus opening doors to various security downgrades even when the client and server both are capable of the more secure implicit TLS. Moreover, many setup guides do not advise users on what to do when being asked about invalid certificates, and some blatantly ask users to blindly accept any certificates when being prompted. All these suggest that the current use of TLS in email clients might not be providing an adequate level of protection for various email protocols, and the auto-detect feature can in some cases be surprisingly harmful to users.

Contributions. This paper makes the following contributions:

1) We design experiments to evaluate 49 email clients, uncovering previously unknown vulnerabilities related to auto-detect and certificate validation when TLS is enabled.
2) We present an evaluation of the email ecosystem used by academic institutes by collecting and evaluating 1102 setup guides. Problems that can drive users to insecure outcomes are identified and discussed.
3) We collected and analyzed 798 certificate chains from email servers identified from setup guides. This analysis reveals some unsatisfactory practices in terms of certificate issuance and management.
4) With a focus on client-side problems, we discuss different possibilities that can make the email ecosystem more secure for its users.

II. Background

In this section, we discuss the relevant technical background for our work. We first give an overview of the three most common email protocols, with respect to their roles in email submission and retrieval. Then we compare implicit TLS (hereafter I-TLS) and opportunistic TLS in the context of email protocols, and explain potential problems in the auto-detect feature of email clients. Although the three email protocols differ slightly in their negotiation messages, they share effectively the same idea of opportunistic TLS.

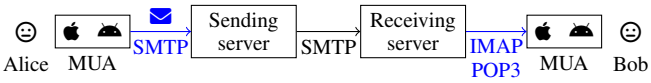

Fig. 1. Example of an email being sent from Alice to Bob. The connections to and from MUAs, highlighted in blue, are the main focus of this study.

A. Adversary Model

We consider two types of on-path MITM attackers: A type ❶ attacker that is *passive*, also known as "honest-but-curious", who merely monitors the network traffic; a type ❷ attacker that is *active*, who in addition to the capabilities of a type ❶ attacker, can also block or tamper with the traffic. A type ❶ attacker is generally more covert than a type ❷ attacker, as the latter might trigger error alerts from the email clients. Also note that any Internet Service Providers (ISPs), including admins of shared Wi-Fi networks, are naturally positioned to be type ❶ and ❷ attackers.

B. Common Email Protocols

Here we discuss the email protocols relevant to our study. An overview of the roles they play in the delivery of an email from Alice to Bob can be found in Figure 1. The connections marked in blue are the primary focus of our study.

SMTP allows electronic mail transmission from Mail User Agent (MUA) to mail provider's Mail Transfer Agent (MTA) [27]. As shown in Figure 1, when Alice sends email to Bob, the email is first handled by the MUA and then sent to the sender's email provider using SMTP. The sending server then passes the email via SMTP to the receiver's email provider, known as the receiving server, which in turn delivers the email to Bob's MUA using IMAP or POP3.

When an SMTP session is initiated, the client first sends `EHLO` command to the SMTP server. The server then responds with a list of supported SMTP extensions. If the server supports opportunistic TLS, then it must include the `STARTTLS` capability in its response to show that it is capable of negotiating and using TLS. According to RFC3207 [23], after the client sends out the `STARTTLS` command to begin TLS negotiation, the server responds in one of the three reply codes listed below:

```
1)  220 Ready to start TLS
2)  501 Syntax error (no parameters allowed)
3)  454 TLS not available due to temporary reason
```

Normally, if server responds with 1), the client then proceeds to the typical TLS handshake, and eventually transits to the TLS secure channel if the handshake is successful. The server responds with 2) when invalid arguments exist in the client's command. However if the server responds with 3), client needs to decide whether to continue with the SMTP session. As suggested by RFC3207 [23], with a server response of 3), the client can try to continue with the session if the server allows no authentication, or wait and try again later, or give up with the connection. If the client attempts to login without TLS, an SMTP servers can reply with a `530 Must issue a STARTTLS command first` to force the client into first establishing a TLS channel and prevent transmitting user credentials over an insecure network.

IMAP allows clients to access messages on their receiving servers and perform various actions to their mailboxes. The client typically initiates an IMAP session by sending the `CAPABILITY` command to get the server extensions. Similar to SMTP, if the server supports opportunistic TLS, then it includes `STARTTLS` in its response, and the client can then use the `STARTTLS` command to attempt an upgrade to TLS. If the IMAP server is willing to upgrade, then it will send a response tagged with `OK`; if not, it will send a response tagged with `NO` or `BAD`. Just like in SMTP, an IMAP server can also try to force the client into using TLS, by indicating the special `LOGINDISABLED` capability when a TLS session is not in place.

According to RFC9051 [34], when it comes to the use of TLS, IMAP clients must implement both I-TLS and STARTTLS to maintain backward compatibility. Additionally, it also suggests that, unless overridden by user configurations, the IMAP client should try both I-TLS on port 993 and STARTTLS on port 143 concurrently, following the so-called Happy Eyeballs algorithm [47], which might reduce the latency perceived by the user. Interestingly, this suggestion is only mentioned for IMAP but not the other email protocols.

POP3 has the `CAPA` command for the client to obtain the extensions supported by the server. A Post Office Protocol (POP3) server announces its STARTTLS capability by including `STLS` in its response. When the client sees this, it can then send a `STLS` command to propose an upgrade to TLS. If the server agrees to the proposal, it should then send a response of `+OK Begin TLS negotiation`. It might send a `-ERR` response when an upgrade to TLS is not possible or a secure channel is already active. Unlike SMTP and IMAP, POP3 does not have a mechanism to force the client to use TLS.

User authentication. The Simple Authentication and Security Layer (SASL) framework is frequently used in email protocols [35] for authenticating the user. As an extensible authentication framework, SASL supports a collection of authentication mechanisms, among which the PLAIN mechanism is the most frequently supported by email servers [24]. RFC4616 [56] states that the PLAIN mechanism should only be used when adequate data security protection is present. In the context of email protocols, such security protection is typically provided by TLS. Due to the plaintext nature of SMTP, IMAP, and POP3 without (START)TLS, when PLAIN is used, even a type ❶ (passive) MITM can trivially observe and obtain user credentials.

Server authentication and TLS. Additionally, SMTP, IMAP, and POP3 with SASL PLAIN also lack guarantees on server authenticity. With TLS (implicit or opportunistic), this problem is typically solved by having the client verify the certificate chain sent by the server during handshake. At a minimum, the client has to perform: (1) certificate chain validation as outlined in RFC5280 [9], which includes verifying digital signatures and expiration dates; and (2) server name verification to make sure that the server certificate has a correct server name. If any of these checks failed, the client should terminate the connection. The TLS handshake continues if the client decides to accept the server's certificate chain. Using a key exchange algorithm negotiated during handshake, the client and server can establish a shared secret, from which secret keys can be derived. The two sides can then use symmetric-key cryptography to protect the confidentiality and integrity of their communication.

C. Use of TLS under different occasions

Now we discuss the various options of using TLS in email protocols and their security guarantees under the threat model defined in Section II-A. Concerning the implementation of STARTTLS, there are actually two possible variants. One is the conventional opportunistic TLS (O-TLS), which *implicitly* also supports falling back to no-TLS. Another variant we refer to as opportunistic *and only* TLS (OO-TLS), which does not fall back to no-TLS when a TLS session fails to establish. Both O-TLS and OO-TLS can defend against the covert eavesdropping of a type ❶ attacker. However, only OO-TLS can defend against a type ❷ attacker, as O-TLS is vulnerable to an active MITM downgrade [14]. Thus, in the face of a type ❶ attacker, the relative security of these TLS usage options can be ordered as follows ($>$ means better): I-TLS = OO-TLS = O-TLS $>$ no-TLS. Alternatively, if one considers a type ❷ attacker, the relative security can be ordered as: I-TLS = OO-TLS $>$ O-TLS = no-TLS. A possible underspecification in the standards is whether an implementation of STARTTLS should be O-TLS or OO-TLS. Such ambiguity presents an interesting engineering decision to email clients, and can lead to noticeably different security outcomes.

D. Problems with configuration auto-detect

Some email clients attempt to auto-detect a functional set of connection parameters. This can be achieved by trying out combinations of conventional port numbers and options of using TLS with the server. Some clients might also attempt to retrieve configuration information through the Autoconfig [6] and Exchange AutoDiscover [37] protocols, as well as DNS SRV [11], [37], [39]. While auto-detect might improve usability, there are a few potential security problems related to it. First, it introduces another security downgrade opportunity that might be exploitable by a type ❷ (active) MITM, which bears some resemblance to the classic TLS version downgrade problem. Although it is possible to signal a version downgrade via ciphersuite [38] and server random [44] in TLS, there is no equivalent solution for determining whether and how TLS should be used in email protocols. Currently it is up to individual vendors to decide how to implement the auto-detect feature of their clients, and they might not always follow the order of relative security discussed above. In fact, as we will show later, some clients perform unnecessary downgrades even when better options are possible. Consequently, users who are not aware of such subtle intricacies might suffer from an inadvertent loss of security.

III. Methodology

To thoroughly and accurately evaluate the email ecosystem with respect to the problems in the use of TLS and auto-detect, we consider three perspectives in this study. First, we test the design and implementation of email clients using purposely designed test cases[1]. This helps us to identify flaws in their auto-detect and certificate verification. Second, we evaluate

[1]Our security downgrade test cases are publicly available at https://github.com/tls-downgrade/email-security.git

email setup guides from educational institutes to see if they provide adequate guidance to drive users to secure outcomes on their email clients. Finally, with the server addresses identified from email setup guides, we collect and evaluate their support for implicit and opportunistic TLS, as well as characteristics of their certificate chains.

A. Email Client Testing

Applications considered. We select email clients from 4 major operating systems according to popularity. For Android apps, we use an open source tool `Kplaysearch` [41] and search for apps that mention the keywords of "email client" and "mail", sorted by installation count on 27th Dec, 2023. We then select the top-10 email clients from this sorted list. For iOS apps, since Apple only allows developers to see the installation count of their own applications [26], we manually select email clients listed in top free "Productivity" and "Business" apps of the US and CN regions[2]. We prioritize selecting iOS email client apps that we also select for Android. Similarly for macOS and Windows, we also select cross-platform email client apps that are available on Android and iOS. We further populate the list of apps to be tested, by gathering email clients used in previous papers [8], [42] as well as in setup guides from different universities (see Section III-B for details). Since some setup guides have not been updated for many years, several email clients mentioned are either superseded by newer versions, or deprecated entirely. If an email client is superseded with a newer version available to us (such as Microsoft Outlook 2013 getting replaced by Microsoft 365), we will consider the latest version of the client app; if an email client is deprecated without a clear successor, then it will not be considered in our study. Eventually, 49 email clients are selected for testing. A detailed list of the selected client apps, their origins and version numbers can be found in Table XII in Appendix E.

Security Downgrade Test Cases. To evaluate the robustness of email client apps with respect to their opportunistic TLS and auto-detect features, we take the role of a type ❷ attacker, and designed 4 test cases based on the relevant RFCs, all of which are applicable to both manual configuration and auto-detect, as well as all three of the email protocols:

- **T_1**: The classic STARTTLS stripping attack where an active MITM removes the `STARTTLS` capability offered by the server, as shown in Fig. 2. This tests whether the client will fallback to no-TLS when opportunistic TLS is not advertised by the server.
- **T_2**: The active MITM replaces `ServerHello` with a cleartext message indicating TLS is not available after the client sent a `ClientHello` for TLS negotiation, as shown in Fig. 3. This tests how a client would respond to an unexpected message at the TLS level.
- **T_3**: Keep the STARTTLS capability offered by the server, but when the client agrees to start TLS negotiation in cleartext stage (after client's `STARTTLS` but before its `ClientHello`), we send a cleartext message indicating TLS is not available, as shown in Fig. 4. Following RFC3207 [23], a client needs to decide whether to proceed with the SMTP session when TLS upgrade is not possible, and this test determines the client's decision on that. Although the idea of this test originates from the SMTP RFC, it can be adopted to test IMAP and POP3 as well, by sending the appropriate error response to `STARTTLS` as discussed in Section II-B (*i.e.*, `454 TLS not available` for SMTP, `BAD` for IMAP, and `-ERR` for POP3).
- **T_4**: When the client and server agree to use STARTTLS to opportunistically upgrade to TLS, disrupt the TLS session by sending arbitrary messages (*e.g.*, `NOOP`) to see how the client responds, as shown in Fig. 5. This tests whether the client can handle disruption after a successful handshake.

For auto-detect, our tests focus on the case where the client heuristically guesses the connection parameters, without relying on Autoconfig, Exchange AutoDiscover and DNS SRV. This is because those 3 mechanisms are not always available, and a type ❷ attacker can force the client to use heuristic guessing by blocking the corresponding DNS queries (type `SRV` for DNS SRV, and type `A/AAAA` for Autoconfig/AutoDiscover subdomains). In our experiments, we observed that only a small number of clients attempt DNS SRV or Autoconfig.

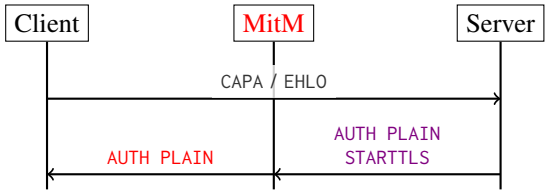

Fig. 2. Classic STARTTLS stripping (test case T_1)

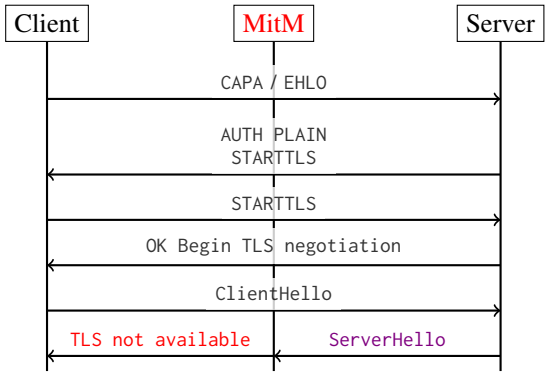

Fig. 3. Replace ServerHello with a plaintext indicating TLS not available (test case T_2)

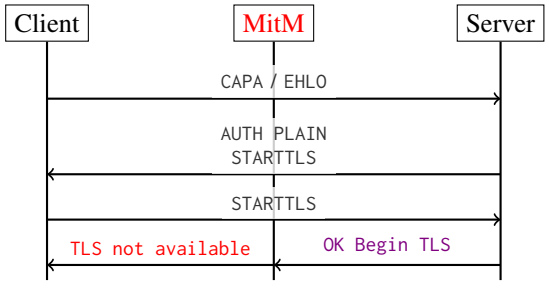

Fig. 4. Return TLS not available in cleartext (test case T_3)

[2]Apps like Protonmail are tied to their own mail services and thus excluded.

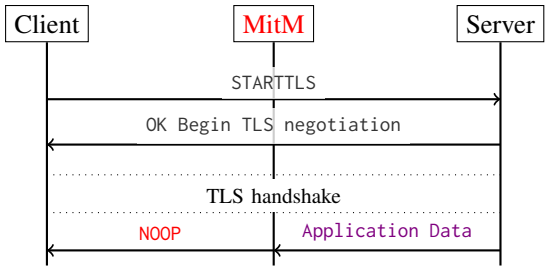

Fig. 5. Disrupt an ongoing TLS handshake (test case T_4)

Certificate Validation Test Cases. To test the certificate validation behavior in both implicit TLS and opportunistic TLS, we prepared 4 (+1) certificate chains: a self-signed server certificate (test case $\mathbf{C_1}$); an expired server certificate that has been expired for at least one month before testing (test case $\mathbf{C_2}$); a server certificate with invalid signature, but the root of its chain is a commercial CA (test case $\mathbf{C_3}$); and a valid server certificate with the prefix of its domain name matching that of the server (test case $\mathbf{C_4}$). C_1 and C_2 can cover the basic validation of certificates, while C_3 covers the scenario where the chain of trust is broken. We designed C_4 because previous studies found that server name verification is sometimes implemented incorrectly with a loose substring matching logic [25], [55], for which an attacker can exploit by creating a matching subdomain on a domain under control (*e.g.*, `imap.victim.com.attacker.com`), and then purchasing a certificate for it from a trusted commercial CA. In order to test whether the client is matching with subdomain or simply does not validate whether the domain matches the ones on server certificate, we prepared $\mathbf{C_4}^{\S}$, which is a valid certificate but from a totally different domain. These test cases cover the most common issues in certificate validation considered by previous work [43].

Attack setup. To apply the test cases, we modified the open-source tool *mitmproxy* [10] to intercept network traffic. To mimic real-world situations, we purchased a domain from GoDaddy and set up mail servers on an AWS EC2 instance. We deploy the open-source *dovecot* [2] as the Internet Message Access Protocol (IMAP) and POP3 server, while for SMTP, we deploy the open-source tool *postfix* [51] as the SMTP server. For certificate test cases, since we have full control over the mail server, we only need to replace the server certificate with our test certificates in the mail server configurations. We opt not to use the toolkit from a previous work [42], in part due to our familiarity with *mitmproxy*, which allows us to easily implement test cases T_1–T_4. Additionally, in order to obtain C_3 and C_4 from commercial CAs, we already need to have functional domain and server setups, so it is easy to reuse them for the actual testing. We believe our test setup is practical and closely resembles production setups, as many of them follow a similar architecture in their deployments [1], [3].

Client configurations. We tested the clients against their implicit TLS, opportunistic TLS, and auto-detect settings. For implicit TLS, we set port 993 for IMAP, port 995 for POP3 and port 465 for SMTP. For testing the behavior of opportunistic TLS, we configure the clients to use port 143 for IMAP, port 110 for POP3, and port 587 and port 25 for SMTP as their respective STARTTLS ports. For testing certificates, if the client offers options related to the verification of certificates, such as "only allow valid certificate", we will enable such options to make the client perform certificate validation.

Client-side support of partial countermeasures. As discussed in Section II-B, in both SMTP and IMAP, the mail servers can hint to the client that a TLS session should be established prior to user login. This could offer protections to sensitive credentials and messages against type ❶ attackers. However, we refer to this as partial countermeasures because they are not effective against a type ❷ attackers. We also test the behavior of mail clients and see whether they aggressively attempt user logins without considering the servers' hints.

B. Setup Guide Evaluation

Data Collection. We focus on the setup guides from universities due to their general availability. From existing literature [25], [54], we compiled a list of universities[3], which includes 7045 universities across 53 regions. We deployed Google Custom Search JSON API [21] on 29th Sept, 2023 to get the top 10 search results from the following search keywords: `e-mail site:<domain>`, `<domain> AND (SMTP OR IMAP OR POP3 OR STARTTLS) site:<domain>`. We have conducted a preliminary screening of the results that are irrelevant by filtering out webpages which do not contain the keywords "SMTP", "IMAP", "POP3". Then we further inspect the search results manually to locate suitable email setup guides. We consider a search result to have an email setup guide if it contains information on teaching users how to configure an email client, including but not limited to names of email clients, screenshots of configuration steps, mail server addresses and ports. One academic institute may have multiple email setup guides, because different academic departments might operate their own email services for their staff and students. Since we collect setup guides across the globe, some non-English speaking regions (*e.g.*, China, Japan, Germany) often prepare setup guides in their respective languages. To ensure comprehensive interpretations and representation of multilingual setup guides, we use Google Translate to help us understand wordings of languages that we do not speak. Among the setup guides we successfully collected, we only evaluate the ones that teach users to connect to a *custom* mail server, and exclude the ones about using Microsoft 365 and Google Workspace, as they are often copies of the setup guides from the service providers [22], [49]. For each setup guide, we manually identify the recommened client, protocols supported, server addresses and ports, along with intructions on the use of auto-detect, TLS, and certificate validation settings, and and whether guidance is given to users on what to do with unexpected prompts and warnings. To ensure consistency, we used a Google Form for data entry, and the data analysis was done on the resulting spreadsheet. The data evaluation process was conducted from November 2023 to March 2024. To improve the stability of our results and resolve potential conflicts or ambiguities, two authors of this paper further inspected and discussed the collected data to eliminate erroneous entries.

Reachable custom mail servers. Eventually, we collected 1899 email setup guides, 1102 of which are about setting up connections with custom mail servers. Given that the server

[3]https://gist.github.com/hugohue/66a45b16bd444f73e757b65eba858113

domains mentioned in some setup guides might have become inactive, we only consider setup guides that provide addresses of servers that are still reachable. For this, we probed mail servers according to the addresses and port numbers mentioned in the setup guides. In the end, we have 810 email setup guides that have reachable servers, 639 of which mention IMAP configurations, 244 mention POP3 configurations, and 729 mention SMTP configurations[3]. These reachable servers also serve as the basis for measuring the server-side characteristics described below.

C. Server-side Characteristics

Certificate collection. We collected certificate chains sent by the reachable mail servers using OpenSSL `s_client`, and then analyze the quality of certificates. The potential security impacts of certificates that do not follow the best practices are two-fold. On the one hand, certificates that cannot be easily verified by programs might incentivize users to skip the critical verification. On the other hand, certificates with weak parameters might also threaten the overall authenticity of the TLS sessions. For our experiments, we used OpenSSL version 1.0.2g, as it supports lower TLS versions and shorter key sizes. In the end, we collected a total of 798 certificate chains from IMAP, POP3 and SMTP mail servers using both I-TLS and STARTTLS. There are 414 unique leaf certificates and 75 unique (1515 total) CA certificates. To test the chains, we use the pyOpenSSL library, and the default CA bundle on Ubuntu 22.04 as the trust anchor. Since our focus is on whether the chains can easily verify, this setup suffices[4].

Server-side partial countermeasures. Finally, we also measure whether mail servers have deployed the mechanisms for hinting to the clients that they should use TLS (*i.e.*, `530 Must issue a STARTTLS command first` for SMTP and the `LOGINDISABLED` capability for IMAP). These mechanisms can be seen as partial countermeasures, because email clients that honor these hints will not transmit sensitive user credentials and messages in plaintext channels.

IV. Findings on Email Clients

A. Security Downgrade Test Cases

Table I shows the security downgrade behaviors of different email clients, considering both manual configuration and auto-detect. Out of the 49 email clients tested, we found that 19 clients may inadvertently downgrade the security of the email protocols to no-TLS without notifying the user, which in some cases can lead to a false sense of security and expose user credentials to an attacker.

O-TLS versus OO-TLS. As discussed in Section II-C, one potential cause of inadvertent security downgrade is the contention between O-TLS and OO-TLS. Through our tests, we have observed that 9 clients indeed adopt O-TLS instead of OO-TLS (T_1–T_4 in manual configuration columns of Table I), and would continue with PLAIN authentication without TLS. Because of this, even if user specifies the use of opportunistic TLS in the client-side configuration, so long as a type ❷ attacker can determine the specific tricks that can induce a downgrade to no-TLS, it is possible to steal the user credentials without the users knowing. In some cases, although the tricks cannot induce a downgrade, the client gets stuck in an infinite loop[5] due to unexpected messages or events (T_2^∞–T_4^∞ in Table I). Client apps are not considered vulnerable if their opportunistic TLS follows the OO-TLS approach (✓ in manual configuration columns of Table I). We note however that some apps prompt their users to decide whether to continue with the downgrade to no-TLS. Following the terminology used in the literature [4], [54], we refer to this as User Insecure (✓̸ in Table I). We will discuss whether this creates security issues in real-world deployments in Section V.

No support for STARTTLS. According to RFC9051 [34], both client and server implementations must implement STARTTLS on cleartext ports. However, we found that some clients do not support STARTTLS when IMAP is used. For instance, myMail on Android only offers the options of implicit TLS and no-TLS for both IMAP and SMTP configurations. Therefore, for a server that uses STARTTLS as the only security mechanism, this client could end up not establishing a secure TLS channel, causing the subsequent communication, including the user credentials, to be sent in cleartext.

Although in RFC8314 [39] the use of I-TLS is more recommended than that of STARTTLS, from the perspective of implementing a client, this does not mean that STARTTLS should not be used, especially when the client is willing to go as low as no-TLS. Email clients should leverage the best security options available to protect the users.

Security downgrade due to auto-detect. In Section II-D, we discussed the potential problems with auto-detect. Through our experiments, we observed two variants of inadvertent security downgrade due to auto-detect. The first variant concerns the order of preference in the different uses of TLS. As discussed in Section II-C, no-TLS is always the worst in terms of security outcome. However, we found that the auto-detect feature of TypeApp Mail and BlueMail prefers no-TLS regardless of the server-side capabilities, instead of first testing whether implicit TLS and STARTTLS can be used (P in the auto-detect columns of Table I). It is not clear to us why these clients automatically opt for the least secure option, which ultimately transmits user credentials over the network in cleartext.

On the other hand, some clients directly jump from I-TLS to no-TLS without showing any warning to the users. For instance, Edison Mail on iOS only offers "Auto", "SSL" and "None" options for IMAP. We tested the "Auto" option, and found that it ignores the STARTTLS capability offered by server and supports neither O-TLS nor OO-TLS. When I-TLS does not work, it proceeds with no-TLS and performs PLAIN authentication without prompting the user.

Apple Mail on iOS takes a similar strategy as Edison Mail on iOS, and ignores the STARTTLS capability offered by server. When configuring IMAP or POP3, if I-TLS does not work, Apple Mail will prompt the user that the connection is not secure, and allows the user to decide whether to proceed (✓̸ in Table I). If the user agrees to proceed, no-TLS will be used.

[3]One setup guide can mention settings for one or more protocols, so the numbers of the three protocols do not always add up to the total.

[4]Those who are interested in fine-grained validation error reasons can use the setup proposed in [5] instead.

[5]Continuously shows a loading screen and attempts to reconnect.

TABLE I. RESULTS OF SECURITY DOWNGRADE TEST CASES ON POPULAR EMAIL CLIENTS

Client	manual configuration SMTP	IMAP	POP3	auto-detect SMTP	IMAP	POP3
Android						
Blue Mail - Email & Calendar	T_1,T_3	✓	✓	✓	✓	–
Boxer - Workspace ONE	✓	✓	✓	–	–	–
Email Aqua Mail - Fast,Secure	✓	✓	✓	T_1	P	–
Email - Fast & Secure Mail	✓	✓	–	P	P	–
Gmail†	✓	✓	✓	✓	✓	✓
K-9 Mail	✓	✓	✓	–	–	–
mail.ru	–	–	–	T_1,T_2,T_3	✓	–
Microsoft Outlook	✓	✓	∞	✓	✓	–
myMail: for Outlook & Yahoo	✓⋆	✓⋆	–	–	–	–
Samsung Email	✓	✓	✓	–	–	–
Spark Mail	✓	✓	–	✓	✓	–
Type App mail - email app	T_1,T_3	✓	∞	✓	✓	–
iOS						
Apple Mail†	–	–	–	T_4	✓⋆	✓⋆
Blue Mail - Email & Calendar	✓	✓	✓	P	P	–
Boxer - Workspace ONE	T_1,T_2^∞,T_3^∞,T_4^∞	T_1,T_4^∞	–	✓≈	✓≈	–
Email Aqua Mail-Fast, Secure	✓	✓	✓	–	–	–
Email - Edison Mail	✓	✓⋆	–	P≈	✓	–
Gmail	✓	✓	–	✓	✓	–
mail.ru	–	–	–	T_1,T_2,T_3	✓	–
Mail Master by Netease	✓	✓	✓	✓	✓	–
Microsoft Outlook	✓	✓	–	–	–	–
Spark Mail	✓	✓	–	✓	✓	–
Type App mail - email app	✓	✓	✓	P	P	–

Client	manual configuration SMTP	IMAP	POP3	auto-detect SMTP	IMAP	POP3
macOS						
Apple Mail†	–	–	–	✓	✓	–
Blue Mail - Email | Calendar	✓	✓	∞	P	P	–
Edison Mail	✓	✓	–	–	–	–
eM Client	T_1	T_1	✓	T_1	T_1	✓
Foxmail	T_3	–	✓⋆	–	–	–
Mail Master by Netease	✓	✓	✓	–	–	–
Microsoft Outlook	✓	✓	✓	✓	✓	–
SeaMonkey (formerly Netscape)	✓	✓	✓	–	–	–
Spark Classic - Email App	✓	✓	–	✓	✓	–
Spark Desktop	✓	✓	–	✓	✓	–
Sylpheed	✓	✓	✓	–	–	–
Thunderbird	✓	✓	✓	✓	✓≈	–
TypeApp	✓	✓	∞	P	P	–
Windows 11						
Becky!	✓	✓	✓	–	–	–
Blue Mail - Email & Calendar	✓	✓	∞	P	P	–
Claws Mail	✓	✓	✓	–	–	–
eM Client	T_1	T_1	✓	T_1	T_1	✓
Foxmail	T_1	✓⋆	✓⋆	–	–	–
Microsoft Outlook	✓	✓	✓	–	–	–
nPOP	T_1,T_4^∞	–	T_4^∞	–	–	–
SeaMonkey (formerly Netscape)	✓	✓	✓	–	–	–
Spark Email for Windows	✓	✓	–	✓	✓	–
Sylpheed	✓	✓	✓	–	–	–
Thunderbird	✓	✓	✓	✓	✓≈	–
TypeApp for Windows	✓	✓	✓	P	P	–
Windows Mail & Calendar	✓	✓	✓	–	–	–

Note: exploitable findings are highlighted in red.
†: Mandatory auto-detect, can manually modify settings afterwards
–: Unsupported ✓: Not vulnerable (no downgrade to *no-TLS*) ✓: User Insecure (prompts user when downgrade to *no-TLS*)
T_1: Strip STARTTLS T_2: Reject `ClientHello` T_3: Reject STARTTLS T_4: Disrupt `ApplicationData` ∞: Infinite loop
≈: Concurrent traffic P: Use plaintext authentication only ⋆: Only implicit TLS and plaintext authentication are supported for these protocols

Concurrent behavior. As discussed before in Section II-B, RFC9051 [34] suggests IMAP clients to try both ports 993 and 143 *concurrently* by default, unless overridden by user configuration or DNS SRV records. RFC9051 cited the Happy Eyeballs algorithm [47] as a good example to implement this concurrency, which was originally designed to reduce perceptible network delays on dual-stack (IPv4 and IPv6) hosts, by making the client send out DNS queries for both IPv4 and IPv6 (A and AAAA records) concurrently. In the context of IMAP, however, this leads to two streams of traffic that will flow to port 993 and port 143 at the same time. Interestingly, in our tests of auto-detect features, we have indeed observed such concurrency features being implemented in some email clients (✓≈ in Table I). In addition to IMAP, some apps (Edison Mail and Boxer on iOS) even implemented this concurrency for SMTP, starting three streams of traffic to the port 587, port 465 and port 25 concurrently. This can be seen as a special case of the auto-detect feature.

The deciding factor of whether such concurrent behavior might open up a vulnerability is the same as that of the conventional auto-detect that tries different uses of TLS in a *sequential* manner. That is, the possibility of a practical attack depends on whether the client is willing to automatically use no-TLS. Based on our testing, the concurrent feature of Thunderbird on both macOS and Windows is only User Insecure (✓≈ in Table I), as it prompts the user before proceeding with no-TLS. The concurrent auto-detect of Boxer on iOS is not deemed vulnerable, because it refuses to automatically use no-TLS (and would revert to manual configuration if both implicit TLS and STARTTLS failed during auto-detect). However, in our experiments, we observed that the SMTP concurrent auto-detect of Edison Mail on iOS initiates 3 separate connections with ports 587, 465 and 25, but in the end chooses to use no-TLS on port 25 (P≈ in Table I) and sends the username and password in cleartext traffic. This is yet another example of inadvertent security downgrade due to auto-detect.

In general, if concurrent auto-detect is to be used, care must be given by the client in deciding which use of TLS is allowed and preferred. In the original Happy Eyeballs algorithm [47], although there are no obvious security concerns between IPv4 and IPv6, preference is given to AAAA records. The order of preferences with respect to the different uses of TLS in a concurrent auto-detect is not specified in RFC9051. If a client is willing to use no-TLS, then a type ❷ attacker can simply block the other concurrent connection attempts and drive the client into using no-TLS. Even under a more limited type ❶ attacker, if the client favors whichever port that responds the fastest, this might create a risk of inadvertent security downgrade due to nondeterminism in network delay.

Confusing terminology. Another cause of concern observed in our experiments stems from the fact that the same keyword might have a different semantic meaning on different products. For example, myMail on Android, Boxer on iOS, and Windows Mail all provide the option to use "SSL" connection or not; while macOS Foxmail allows user to choose "Secure Connection". However, these options have ambiguous meaning in different clients. Table II shows different interpretations on the SSL switch in various clients. On Windows Mail, if the "SSL" option is checked, it will use OO-TLS (but not I-TLS); otherwise, it will use O-TLS. Meanwhile, if the "SSL" option

is checked, Boxer on iOS, myMail on Android, and Foxmail on macOS will all treat the port as having I-TLS (even for port numbers that typically used for STARTTLS, such as 587), and directly send `ClientHello` to the port. Unchecking the "SSL" option will cause myMail on Android and Foxmail on macOS to use no-TLS, while Boxer on iOS will then use STARTTLS. This kind of subtle but critical semantic differences of the same keyword are difficult for users to comprehend, and might potentially lead to misconfiguration due to confusions.

TABLE II. DIFFERENT OUTCOMES OF THE "SSL" SWITCHES

Clients (OS)	Wordings	Switched on	Switched off
Windows Mail (W11)	"Require SSL"	OO-TLS	no-TLS
Boxer Mail (iOS)	"Use SSL"	I-TLS	O-TLS
myMail (Android)	"SSL"	I-TLS	no-TLS
Foxmail (macOS)	"Secure Connection"	I-TLS	no-TLS

In our experiments, we observed some Windows clients offering options related to old versions of TLS. For the sake of brevity, we give a detailed account of this in Appendix A.

Proxy-based email clients. The mail.ru clients use an HTTPS proxy for communicating with the mail servers. Thus, we moved the experiment setup to the server-side to test the connection between their proxy server and our mail server. We found that the proxy only attempts I-TLS for IMAP. However, for SMTP, the proxy server of mail.ru is vulnerable to T_1–T_3, as shown in Table I. Under T_1–T_3, when a TLS session cannot be established, the mail.ru proxy will retry the connection using no-TLS, and then sends the user credentials in cleartext. A type ❷ attacker that is on-path between the proxy and the actual mail server (*e.g.*, an ISP in those jurisdictions) will thus be able to exploit this for credential theft.

Lessons learned: Email clients vary noticeably in their implementation of TLS options and thus deliver mixed security outcomes. On some clients, users are better off without relying on auto-detect.

B. Certificate Validation Test Cases

Table III shows the findings of our certificate test cases. Additionally, we also tested the clients on revocation checks. Only 12 clients properly validated the revocation status. We provide a detailed analysis in Appendix B.

Overview of the results. We found that out of the 49 email clients tested, 19 of them accept at least one of our supplied certificate. If a client accepts any of C_1–$C_4^{\S}$, its certificate validation is missing some critical checks under specific use of TLS in some of the email protocols.

Loose checking of the certificate domain names. As shown in Table III, we found that 18 clients do not perform proper checks to match the sever domain name with the domain names on the certificate (C_4 and $C_4^{\S}$ in Table III). All but one clients that accept C_4 also accept $C_4^{\S}$. For the email clients that do not exhibit general leniency towards certificates (*e.g.*, Blue Mail on iOS, with ✓ for $C_{1–3}$), their acceptance of $C_4^{\S}$ suggests that they might not be performing any name checking at all (despite checking the validity of the certificate chain).

Proxy-based email clients. Under this setting, the proxy is in charge of validating the server certificate. We found that the mail.ru proxy only supports implicit TLS with IMAP (◖) and STARTTLS with SMTP (◗), and both accept all of our test certificates. Similar vulnerabilities have been observed in other proxy-based appliances by previous work [12], [13], [52].

Discrepancy between protocols and use of TLS. From Table III, we can see that several apps exhibit discrepancies across the three email protocols. For example, Outlook on macOS accepts an expired certificate in POP3 (● in C_2) but not in other protocols. Similarly, apps like Spark, Aqua Mail on iOS and Edison Mail on macOS have certificate validation issues in IMAP but not in SMTP.

Furthermore, from Table III, we can see that for IMAP, there are generally more certificate validation issues with STARTTLS than I-TLS (◑). This is in contrast to the results of a previous work [42], which reports that the certificate validation issues in I-TLS and STARTTLS are similar. This can be attributed to the fact that many of the clients contributing to this discrepancy here were not tested before. For instance, Spark alone contributed many instances of certificate validation issues in STARTTLS but not in I-TLS on different operating systems. Likewise, Aqua Mail on iOS and Edison Mail on macOS also exhibit similar discrepancies.

Interestingly, Email - Fast & Secure on Android seems to be performing certificate chain validation but not the name verification for IMAP with I-TLS (◑ in $C_{1–3}$ but ● in C_4 and $C_4^{\S}$), whereas for IMAP with STARTTLS it accepts all certificate test cases. Boxer on iOS accepts $C_{1–4}$ but not $C_4^{\S}$ for IMAP, which suggests that no certificate chain validation is in place, but it performs a prefix match of the server name.

Fallback to no-TLS over bad certificates. We found that Apple Mail on iOS prompts the user to fall back to no-TLS when the server certificate chain cannot be accepted. This is a worrisome design, because users might choose to continue without understanding the implications. A better alternative would be to prompt the user to accept the certificate and continue with the TLS session, which can at least protect against a type ❶ attacker, especially if the genuine mail server indeed uses a bad certificate chain.

Lessons learned: Some email clients continue to miss the mark on certificate validation. Additionally, some have inconsistent implementations of certificate validation within the same app, and thus users do not always get a coherent level of security across products, protocols, and setups.

C. Client-side support for partial countermeasures

When manual configuration is used, 29 clients allow the choice of no-TLS. We found that only 9 of them honor the `LOGINDISABLED` capability sent by IMAP servers. Three of them will automatically switch to use STARTTLS, and the remaining 6 do not allow users to proceed and terminate the connection. The other 20 clients do not honor `LOGINDISABLED` and continue to login with credentials sent in cleartext. The full result can be found in Table IV. For SMTP, 11 out of the 29 clients will *aggressively* include user credentials in their login attempts. Because of this, before the server gets to send `530 Must issue a STARTTLS command first`, passwords will already be transmitted in cleartext.

TABLE III. RESULTS OF CERTIFICATE TEST CASES ON POPULAR EMAIL CLIENTS

Client	C_1 SMTP	C_1 IMAP	C_1 POP3	C_2 SMTP	C_2 IMAP	C_2 POP3	C_3 SMTP	C_3 IMAP	C_3 POP3	C_4 SMTP	C_4 IMAP	C_4 POP3	$C_4^{\S}$ SMTP	$C_4^{\S}$ IMAP	$C_4^{\S}$ POP3
Android															
Blue Mail - Email & Calendar	✓	✓	∞	✓	✓	∞	✓	✓	∞	✓	✓	∞	✓	✓	∞
Boxer - Workspace ONE	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Email Aqua Mail - Fast,Secure	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Email - Fast & Secure Mail	✗	◑	–	✗	◑	–	✗	◑	–	●	●	–	●	●	–
Gmail	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
K-9 Mail	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
mail.ru	◗	◖	–	◗	◖	–	◗	◖	–	◗	◖	–	◗	◖	–
Microsoft Outlook	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
myMail: for Outlook & Yahoo	✓TLS	✓TLS	–	✓TLS	✓TLS	–	✓TLS	✓TLS	–	✓TLS	✓TLS	–	✓TLS	✓TLS	
Samsung Email	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Spark Mail	✗	◑	–	✗	◑	–	✗	◑	–	●	●	–	●	●	–
Type App mail - email app	✓	✓	∞	✓	✓	∞	✓	✓	∞	✓	✓	∞	✓	✓	∞
iOS															
Apple Mail⋆	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Blue Mail - Email & Calendar	✓	✓	✓	✓	✓	✓	✓	✓	✓	●	●	✓	●	●	✓
Boxer - Workspace ONE	◐∞	●	–	◐∞	●	–	◐∞	●	–	◐∞	●	–	✗∞	◑	✗
Email Aqua Mail-Fast, Secure	✓	◑	✓	✓	◑	✓	✓	◑	✓	✓	◑	✓	✓	◑	✓
Email - Edison Mail	✗	✗	–	✗	✗	–	✗	✗	–	✗	◖	–	✗	◖	–
Gmail	✓	✓	–	✓	✓	–	✓	✓	–	✓	✓	–	✓	✓	–
mail.ru	◗	◖	–	◗	◖	–	◗	◖	–	◗	◖	–	◗	◖	–
Mail Master by Netease	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
Microsoft Outlook	✗	✗	–	✗	✗	–	✗	✗	–	✗	✗	–	✗	✗	–
Spark Mail	✗	◑	–	✗	◑	–	✗	◑	–	✗	◑	–	✗	◑	–
Type App mail - email app	✓	✓	✓	✓	✓	✓	✓	✓	✓	●	●	◐	●	●	◐
macOS															
Apple Mail	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Blue Mail - Email | Calendar	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Edison Mail	✗	◑	–	✗	◑	–	✗	◑	–	✗	◑	–	✗	◑	–
eM Client	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Foxmail	●	–	◖	●	–	◖	●	–	◖	●	–	◖	●	–	◖
Mail Master by Netease	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
Microsoft Outlook	✗	✗	✗	✗	✗	●	✗	✗	✗	✗	✗	✗	✗	✗	✗
SeaMonkey (formerly Netscape)	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Spark Classic - Email App	✗	◑	–	✗	◑	–	✗	◑	–	✗	◑	–	✗	◑	–
Spark Desktop	✗	◑	–	✗	◑	–	✗	◑	–	✗	◑	–	✗	◑	–
Sylpheed	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Thunderbird	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
TypeApp	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Windows 11															
Becky!	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Blue Mail - Email & Calendar	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Claws Mail	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
eM Client	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Foxmail	●	◖	◖	●	◖	◖	●	◖	◖	●	◖	◖	●	◖	◖
Microsoft Outlook	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
nPOP	✓	–	✓	✓	–	✓	✓	–	✓	✓	–	✓	✓	–	✓
SeaMonkey (formerly Netscape)	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Spark Email for Windows	✗	◑	–	✗	◑	–	✗	◑	–	●	●	–	●	●	–
Sylpheed	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Thunderbird	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
TypeApp for Windows	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Windows Mail & Calendar	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗

⋆: Downgrade to no TLS if user chooses to continue when prompted TLS: Only implicit TLS is available
✓: Not vulnerable ✗: User Insecure –: Unsupported protocol ∞: Infinite loop C_1: Self-signed certificate
C_2: Expired certificate C_3: Invalid signature certificate C_4: Mismatch prefix certificate $C_4^{\S}$: Mismatch subject certificate
◑: Vulnerable in STARTTLS ◐: Vulnerable in I-TLS ●: Vulnerable in both I-TLS and STARTTLS
◗: Vulnerable in STARTTLS (no support for I-TLS) ◖: Vulnerable in I-TLS (no support for STARTTLS)

There are 8 clients that have P in the auto-detect columns of Table I. We found that only Aqua Mail and Edison Mail on Android do not honor the `LOGINDISABLED` capability of an IMAP server, and the remaining 6 all terminate the connection. Concerning SMTP, only Edison Mail in Android will aggressively include user credentials in its login attempt.

Lessons learned: Nearly half of the tested clients aggressively sends user credentials, which is not the best strategy for protecting its users. The partial countermeasures have their merits and should be better supported by more clients.

V. FINDINGS ON REAL-WORLD DEPLOYMENTS

A. Setup guides

Overall statistics. Based on the server and protocol information extracted from the 810 setup guides with reachable servers, we classified them into 4 categories. The results are listed in Table V. A majority of setup guides have *clearly defined* information about the security mechanisms to use, and around one-fifth of setup guides instruct users to use *auto-detect*. Some setup guides only specify the server port number and abstractly mention the use of TLS (*ambiguous*), and some only mention the server port number without specifying

TABLE IV. CLIENTS THAT SUPPORTS PARTIAL COUNTERMEASURES

Clients	IMAP `LOGINDISABLED`	SMTP `must issue STARTTLS`
Android		
Blue Mail - Email & Calendar	✗	✗△
Email Aqua mail - Fast, Secure	✗	✗△
Email - Fast & Secure Mail	✗	✓
Gmail	✗	✓
K-9 Mail	✓	✓
Microsoft Outlook	✗	✓
Samsung Mail	✗	✓
Type App mail - email app	✗	✗△
iOS		
BlueMail - Email & Calendar	✗	✗
Email Aqua mail - Fast, Secure	✗	✗
Email - Edison Mail	✗	✓
Mail Master by Netease	✗	✓
Microsoft Outlook	✗	✓
Type App mail - email app	✗	✗
macOS		
Blue Mail - Email & Calendar	✗	✗△
Edison Mail	✗	✓
Mail Master by Netease	✗	✗△
Microsoft Outlook	✗	✓
SeaMonkey (formerly Netscape)	✓†	✓
Sylpheed	✓	✓
Thunderbird	✓	✓†
TypeApp	✗	✗△
Windows 11		
BlueMail - Email & Calendar	✗	✗
Claws Mail	✓†	✓†
Microsoft Outlook	✓	✓
SeaMonkey (formerly Netscape)	✓	✓
Sylpheed	✓†	✓
Thunderbird	✓	✓†
TypeApp for Windows	✗	✗

✓: Honor `LOGINDISABLED` or `Must issue STARTTLS first`
✗: Do not honor `LOGINDISABLED` or `Must issue STARTTLS first`
†: Switch to STARTTLS connection automatically
△: Send user credentials aggressively if `AUTH PLAIN` is selected

whether TLS should be used or not (*unknown*). We note that setup guides with *ambiguous* TLS information are not necessarily problematic, so long as their recommended client has a well-defined SSL/TLS option in its user interface (UI).

We then further separate the 810 setup guides into *generic* setup guides and *specific* setup guides. Generic setup guides are not tied to a specific client app. They typically only provide general information regarding the mail server address, port, and the use of TLS. Users will have to adapt the given information to suit the client app they choose to use. On the other hand, a specific setup guide is tied to a particular client app on an operating system, and usually provides detailed steps and/or screenshots that teach users how to configure the client app. Out of the 810 setup guides, we found that 310 (38.27%) are generic, and 500 (61.73%) are specific. As can be seen in Table V, many *specific* setup guides have *clearly defined* information about what security mechanisms to use.

To see how the tested email clients are used in actual deployments, we classified them into three categories: (1) `Secure`, (2) `User Insecure` and (3) `Insecure`, for three different aspects, which are, *Downgrade*, *Certificate validation*, and *Configuration UI*. The first 2 aspects correspond to the findings discussed in Sections IV-A and IV-B. The *Configuration UI* aspect concerns whether there are UI options for users to turn off certain security-critical mechanisms.

A breakdown of email apps prescribed in the 500 specific setup guides can be found in Table VI. A client is considered `Secure` for the *Downgrade* and *Certificate validation* columns, if all the tests in Table I and III are secure. If any of those tests are `User Insecure` (or respectively, `Insecure`), then the client will be marked as such in Table VI. For the *Configuration UI* column, a client is considered `User Insecure` if it contains options for users to disable the use of TLS or certificate checks (*e.g.*, a checkbox to "accept all certificates").

Leave security up to interpretation. The first interesting observation is that many email setup guides leave users to interpret and figure out what settings to use. In our data set, this observation manifests in two scenarios. The first one is that there is a noticeable portion (38.27%) of setup guides that are deemed generic. Users who are given these generic setup guides are not prescribed a recommended client app nor detailed steps on how to configure their clients. Given that users are free to choose their preferred clients, and we have shown that many email clients are `User Insecure`, generic setup guides might not be able to help users to arrive at the best configuration in terms of security. For instance, among the 310 generic setup guides, 13 (7 for POP3 and 6 for SMTP) only abstractly mention "SSL" or "TLS", without explaining whether the client should be using implicit TLS or STARTTLS. Additionally, we found 24 generic setup guides (9 for IMAP, 2 for POP3, and 13 for SMTP) only mention the server port that users should connect to, without specifying how and whether TLS should be used.

Another scenario that depends on users' interpretation concerns the 42 *specific* setup guides that recommend the use of a special client called "Android System Client". We note that the exact client app for this varies on devices and typically depends on the device vendors. For example, Google devices usually default to Gmail, while Samsung devices default to Samsung Email. Other vendors can also ship their own email app as the system client. Together with the fact that clients might differ in their behaviors and UI options, users given a specific setup guide based on one system client might end up not achieving the best possible configuration outcome when using a different system client.

Some certificate validation disabled from UI. Among the recommended email clients shown in Table VI, 5 of them offer the UI option of "accept all certificates", which effectively disables certificate validation even when TLS is used. We pulled their corresponding setup guides to see if users are told to choose this option. While a majority of these *specific* setup guides leave that option unchecked *by default*, 6 setup guides (2 for Becky! and 4 from non-Gmail Android System Clients, e.g., Samsung Mail) instruct users to enable the "accept all certificates" option. Only 1 setup guide *explicitly* advises its user to make sure that option is unchecked.

Poor exception handling in certificate validation. As discussed before, when it comes to certificate validation, many email clients are `User Insecure`. However, we found that a vast majority of setup guides do not instruct users what to do if they see a warning prompt during certificate validation. In fact, we found that 14 setup guides (9 for IMAP, 4 for POP3, 13 for SMTP) ask the users to directly continue whenever they are prompted. We crawled 26 certificate chains from those mail servers, and found that 22 of them can successfully pass through the chain validation and hostname verification using

TABLE V. SECURITY MECHANISMS MENTIONED BY SETUP GUIDES

Security Mechanisms	IMAP (total = 639) Count	Perc.	Specific	Generic	POP3 (total = 244) Count	Perc.	Specific	Generic	SMTP (total = 729) Count	Perc.	Specific	Generic
Clearly defined	484	75.74%	223	261	173	70.90%	84	89	518	71.06%	302	216
Auto-detect	132	20.66%	104	28	55	22.54%	28	27	130	17.83%	95	35
Port+TLS unknown	19	2.97%	10	9	9	3.69%	6	3	29	3.98%	16	13
Port+TLS ambiguous	4	0.63%	1	3	7	2.87%	0	7	52	7.13%	30	22

TABLE VI. OCCURRENCE OF EMAIL CLIENTS IN SETUP GUIDES

Clients	Count	Percent	Down-grade	Cert. Validaton	Config. UI
Android	Total = 61				
Android System Client†	42	68.85%	–	–	–
Gmail	–	–	✓	✓̸	✓̸
Samsung Email	–	–	✓	✓̸	✓̸
Gmail	10	16.39%	✓	✓̸	✓̸
K-9 Mail	4	6.56%	✓	✓̸	✓̸
Microsoft Outlook	2	3.28%	✓	✓̸	✓
Boxer - Workspace ONE	1	1.64%	✓	✓	✓̸
iOS	Total = 73				
Apple Mail	68	93.15%	✗	✓̸	✓
Microsoft Outlook	3	4.11%	✓	✓	✓
Thunderbird	2	2.74%	✓̸	✓̸	✓
macOS	Total = 100				
Apple Mail	74	74.00%	✓	✓̸	✓
Thunderbird	11	11.00%	✓̸	✓̸	✓
Microsoft Outlook	11	11.00%	✓	✗	✓
Foxmail	2	2.00%	✗	✗	✓
Windows	Total = 266				
Microsoft Outlook	130	48.87%	✓	✓̸	✓
Thunderbird	95	35.71%	✓	✓̸	✓
Windows Mail & Calendar	20	7.52%	✓	✓̸	✓
Becky!	6	2.26%	✓	✓	✓̸
Foxmail	6	2.26%	✗	✗	✓
SeaMonkey	2	0.75%	✓	✓̸	✓

Note: 11 discontinued clients are excluded from this list.
†: Android System Client depends on the device's make and model.
✓: Secure ✓̸: User-Insecure ✗: Insecure

the setup discussed in Section III-C. As such, there is actually no legitimate reasons for the corresponding setup guides to ask their users to accept any arbitrary certificates.

Overall, we found only 2 setup guides that gave some useful information on handling warning prompts. Unfortunately, they only mention the name of the certificate issuer, and do not explicitly tell the user what actions to take. We also note that so long as the chain validity is not checked, the issuer name is an unauthenticated input that can be arbitrarily chosen by a type ❷ attacker, and thus even if a user tries to match the issuer name, server impersonation can still succeed.

Auto-detect is common but can be tricky. For generic setup guides, we found 90 (28 for IMAP, 27 for POP3, and 35 for SMTP) that abstractly state auto-detect can be used by the user for configuration. For specific setup guides, we found 227 (104 for IMAP, 28 for POP3, 95 for SMTP) that instruct users to use auto-detect on the recommended clients. These numbers suggest that auto-detect is commonly used by users in real-world deployments. However, we also observed none of these setup guides discuss what the user should do if there is a warning prompt during auto-detect. As discussed in Section IV-A, a number of clients are `User Insecure`, that is, before they downgrade to using no-TLS, they show a prompt and let the user decide whether to continue or not. Consequently, not instructing users on what to do when they see the prompt leaves open the possibility for an unexpected downgrade to cleartext communication.

Additionally, we also observed that the 3 clients marked as `Insecure` for *Downgrade* in Table VI are indeed being used in practice. We also note that the design of Apple Mail on iOS could be worsening the overall security. Out of the 68 setup guides that recommend it, 43 instruct users to complete the configuration using its auto-detect feature, and the remaining 25 provide detailed information on security settings for manual configuration. However, since Apple Mail makes it mandatory to use auto-detect when creating a new configuration, by the time that the user gets to manually adjust the security settings, the client would have already performed at least one login attempt, possibly in cleartext.

Prescribing the use of no-TLS and STARTTLS. We found that 12 setup guides (3 for IMAP, 3 for POP3, 6 for SMTP) instruct users to use no-TLS. This is far from ideal, as users who follow these setup guides are directly exposed to the threat of credential theft. Additionally, we found that 52 setup guides (12 for IMAP, 4 for POP3, 36 for SMTP) only instruct users to use STARTTLS.

In order to evaluate whether these setup guides could have prescribed a more secure option, we used Python and OpenSSL to probe all the reachable mail servers to determine the use of TLS that they support. We try to connect to the mail servers with implicit TLS, STARTTLS and no TLS, using the addresses and port numbers extracted from the setup guides, as well as the conventional port numbers for implicit TLS and STARTTLS. The overall statistics of what the 810 servers support can be found in Table VII. Encouragingly, a significant portion of SMTP servers and an overwhelming majority of IMAP and POP3 servers support implicit TLS, although a lot more SMTP servers support STARTTLS than implicit TLS. In fact, quite some IMAP and POP3 servers *only* support implicit TLS. Nevertheless, a noticeable portion of mail servers continue to support no-TLS.

With these results, we can then revisit the no-TLS and STARTTLS setup guides. Interestingly, for the 12 setup guides that instruct users to use no-TLS, all of their mail servers actually support STARTTLS. This means that the setup guides should have prescribed STARTTLS instead, which can at least defend against type ❶ attacks.

Furthermore, for the 52 setup guides that instruct users to use STARTTLS, we found that their servers all support implicit TLS. With the majority of email clients capable of using implicit TLS, the setup guides could have prescribed

TABLE VII. SECURITY MECHANISMS DEPLOYED BY SERVERS

Supported use of TLS	IMAP servers (total = 280)		POP3 servers (total = 129)		SMTP servers (total = 321)	
	Count	Perc.	Count	Perc.	Count	Perc.
Implicit TLS	274	97.86%	126	97.67%	197	61.37%
Implicit TLS only	111	39.64%	54	41.86%	18	5.61%
STARTTLS	160	57.14%	64	49.61%	290	90.34%
STARTTLS only	2	0.71%	0	0.00%	67	20.87%
no TLS	70	25.00%	49	37.98%	134	41.74%
no TLS only	0	0.00%	4	3.10%	7	2.18%

that instead, and mitigate possibilities of inadvertent security downgrades under type ❷ attacks.

Use of unconventional ports. We noticed that some mail servers use unconventional ports for different mail protocols. 3 servers use port 993 for POP3 with implicit TLS, instead of the typical port 995; 3 servers use port 587 for SMTP with implicit TLS, instead of the typical port 465; 6 servers use port 465 as STARTTLS port, and 4 servers use *no-TLS* in port 465. We found that all of their corresponding setup guides offer detailed information for users to configure *manually*. Nevertheless, these non-standard ports might cripple auto-detect on some email clients, and if the server happens to provide services at a worse security level on conventional ports (*e.g.*, SMTP with no TLS on port 25), then the auto-detect mechanism on client-side might fail to detect the existence of better security options, and end up downgrading to the worse ones.

Lessons learned: Many setup guides are not explicit on how the users can better protect themselves, especially regarding the use of security mechanisms and their exception handling. Instead of focusing on making clients work, setup guides should elaborate on how to make clients safe.

B. Characteristics of server certificates

Chain validation. We attempt to verify the 798 certificate chains collected from mail servers using the setup mentioned in Section III-C. We found that only 21 of them failed in chain verification. While 3 of them are self-signed certificates, 4 are expired, and 6 are missing some issuer certificates. 8 of 21 chains failed validation due to not including the `subjectKeyIdentifier` in the root certificate, which can be seen as non-compliant to RFC5280 [9], as it states that the `subjectKeyIdentifier` extension must appear in all conforming CA certificates. We then investigate the remaining 2 failed certificate chains, and found that both of them have a root of trust that is not trusted by our root CA bundle.

Hostname verification. We also performed hostname validation on leaf certificates. Only 13 of them failed the hostname verification. 11 (2.66%) of them failed to match with the hostname in both Common Name and subjectAltName. 2 of them do not include subjectAltName extension in their leaf certificates. In this case, according to RFC6125 [45], if a certificate does not include a DNS-ID, SRV-ID, URI-ID or any other application-specific identifiers types, then the Common Name field may be used as the last resort to perform DNS domain name checking. While one of the 2 has a Common Name "Unknown" and thus ultimately failed the hostname verification, we found that the other has a Common Name field that can match with the server domain name, which is acceptable according to RFC6125. However, we note that at the time of writing, RFC6125 is obsoleted by RFC9525, which states that the Common Name field should no longer be used for hostname verification, and so this certificate might get rejected by email clients that implement the new standard.

TABLE VIII. CERTIFICATE ANALYSIS

Parameter	Leaf cert.=414		CA cert.=89	
	Count	Perc.	Count	Perc.
Public Key Type				
RSA Public Key	399	96.37%	83	93.26%
EC Public Key	15	3.62%	6	6.74%
Key Size in bits (algorithm)				
$<$ 2048 (RSA)	1	0.24%	0	0.00%
$\geq$ 2048 (RSA)	398	96.14%	83	93.26%
384 (EC)	5	1.21%	3	3.37%
256 (EC)	10	2.42%	3	3.37%
Signature Algorithm				
SHA1-RSA	1	0.24%	10	11.23%
SHA256-RSA	236	57.00%	52	58.43%
SHA384-RSA	164	39.61%	21	23.60%
SHA512-RSA	0	0.00%	1	1.12%
SHA256-ECDSA	12	2.90%	0	0.00%
SHA384-ECDSA	1	0.24%	5	5.62%

Public keys and signature algorithms. To get a sense of the general security posture of the certificates used by mail servers, we tried to extract the public key information, key size and signature algorithm used in the certificates. The results are generally promising, and can be seen Table VIII. Only 1 leaf certificate has an RSA key shorter than 2048 bits (the modulus is 1024-bit long), and only 1 leaf certificate has the relatively weak SHA1-RSA signature, which may be susceptible to collision attacks [31]. An overwhelming majority of leaf certificates have decent key sizes and signatures. A similar trend can be seen in the CA certificates that form the chain of trust, although there are slightly more CA certificates signed with SHA1-RSA.

Lifespan. We also analysed the lifespan of the certificates collected. For leaf certificates, 10.14% of them have lifespan less than 90 days, 88.89% of them are within the normative period (of not more than 5 years). Only 2 leaf certificates have a lifespan of 10 years and 1 has a lifespan of 16 years. For CA certificates, 6.74% have a lifespan of less than 5 years, 75.28% have typical lifespans (between 5 to 20 years), and 17.98% have a lifespan more than 20 years. The longest lifespan observed on the CA certificates is 30 years.

Lessons learned: Obtaining and deploying trustworthy certificates on the server-side is not a major pain point. As such, the overall security can be improved by making setup guides more stringent on instructing users how to handle the configuration UIs and exceptions regarding certificates.

C. Server-side partial countermeasures

Deployments on mail servers. Finally, we also measure how many mail servers support the partial countermeasures that can explicitly hint to the client that TLS needs to be established before login. For IMAP, this refers to the `LOGINDISABLED` special

capability, and for SMTP, this refers to the `530 Must issue a STARTTLS command first` message.

Our measurements show that about two-third of the tested servers support the aforementioned partial countermeasures, see Table IX for details. Interestingly, for IMAP servers, we also found that 16 of them take a different approach at hinting to clients. That is, instead of stating the `LOGINDISABLED` capability, they will *hide* all the authentication methods from the list of capability, and leave STARTTLS as the only option for the client. Only around 10% of IMAP servers adopts this approach.

For SMTP, the semantics are slightly different. Since message `530` is a reactive signal in response to a client's `AUTH LOGIN` message, in order to prevent aggressive clients (*cf.* Section IV-C) from directly attempting login (and transmitting username and password in one go), typically the servers should also *hide* the other extensions except for `STARTTLS` when responding to client's `EHLO`. Out of the 202 SMTP servers that will send message `530`, we found that only 179 also hide their non-STARTTLS capabilities. Because of this, if an aggressive client is used with the remaining 23 SMTP servers, a type ❶ attacker might still be able to obtain the user credentials.

Finally, we also observed that some SMTP servers enforce different policies on different ports. Perhaps unsurprisingly, there are 17 servers that force client to use STARTTLS before allowing login on port 587, but clients are allowed to login with no TLS on port 25.

TABLE IX. DEPLOYMENTS OF PARTIAL COUNTERMEASURES

	IMAP (total = 160)		SMTP (total = 290)	
Partial countermeasures	Count	Perc.	Count	Perc.
Explicit hints to clients	100	62.50%	202	69.66%
Hiding capabilities	16	10.00%	179†	61.72%
Does not deploy countermeasures	44	27.5%	105	36.21%

†: Those that hide capabilities also send message `530`.

Lessons learned: Many mail servers already deployed partial countermeasures that can prevent cleartext transmission of user credentials. This suggests that the onus is indeed on the email clients to honour these countermeasures and avoid aggressively sending user credentials.

VI. DISCUSSION

Responsible disclosure. The findings in Tables I and III are already shared with the corresponding vendors. At the time of writing, eM Client has replied to us, stating that when auto-detect is used, it only queries the STARTTLS capability during the first connection attempt, and that it assumes the initial configuration is done on a safe network. We replied back to remind them that the safe network assumption is quite strong, and that the STARTTLS stripping attack should be addressed irrespective of auto-detect. Apple has confirmed our findings on iOS Apple Mail, and a fix has been planned for late 2024. We are waiting for the other vendors to respond. We have also responsibly notified the institutions who prescribed problematic setup guides. We received positive feedback from two universities. They thanked us for our reports and promised to reevaluate their setup guides.

Ethical considerations. The testing of email clients are conducted in a controlled environment, with our own server setups. The tests do not involve any sensitive information concerning real-world personal identity. For the server-side measurements, we only probed the servers to observe their capabilities, without sending or obtaining any sensitive information. We never sent any malformed messages to the servers, and the amount of traffic is negligible. Thus, our measurements should not affect the normal functionalities of the mail servers.

Limitations. Concerning email clients, the so-called "Android System Client" can vary across vendors (see Section V), and our options are limited by the variety of Android devices available to us. We thus limit our analysis to Gmail and Samsung Email. For setup guides (and server-side deployment practices), due to the nature of our collected data, our analysis and findings are heavily influenced by deployments at educational institutes. Even with a focus on universities, some email setup guides are hidden in their intranet and have evaded our analysis. It is also possible that some of the collected setup guides are not the latest versions available. These also explain why some of the tested email clients are not observed in the collected setup guides. Part of our analysis involved the use of Google Translate, and we assume its translations preserve the semantic meanings of setup instructions. Despite these limitations, we believe our study still helps to shed light on the common but unsatisfactory practices embraced by vendors and universities, which likely also apply to other organizations.

A. Possible mitigation for IT admins

Avoid using auto-detect. As observed in Section V, the auto-detect feature is indeed popular. Although auto-detect appears to be convenient, when considering the fact that there are many subtle security downgrades (both `Insecure` and `User Insecure`), it makes us question whether auto-detect should be avoided in favor of the more traditional manual configuration. One successful and secure email configuration can last for a relatively long period time, and given this long-term benefit, the amortized effort of manually specifying the port number and use of TLS during configuration seems rather insignificant. We argue that the risks of auto-detect outweigh its benefits.

Avoid "happy path" setup guides. Analogous to happy path testing, "happy path" setup guides only focus on helping users to connect successfully, with no regards on possible attacks. Unfortunately, security is often achieved only with proper exception handling. Because of this, IT admins are strongly encouraged to explicitly educate their users on how to protect themselves when they encounter unexpected warning prompts from email clients. This is especially important given the myriad of clients that exhibit `User Insecure` behaviors in their auto-detect and certificate validation.

B. Possible mitigation for vendors and standardization bodies

Prohibit no-TLS and upgrade O-TLS to OO-TLS. Although RFC8314 [39] has deprecated the use of cleartext access to mail servers, our results show that many clients still retain support of no-TLS and are willing to use it for transmitting sensitive credentials and messages, either in an `Insecure` or `User Insecure` manner. There are no good reasons to keep

supporting no-TLS, especially when an overwhelming majority of mail servers also support STARTTLS and I-TLS (Table VII). Several clients, such as Gmail in iOS, already ceased to support no-TLS. We recommend other vendors to follow suit. An added bonus of prohibiting no-TLS is that the use of STARTTLS will no longer be vulnerable to the stripping-style attacks (effectively upgrading O-TLS to OO-TLS).

Avoid ambiguity in concurrent connections. Based on our results and investigation, a possible source of ambiguity appears to be the handling of concurrent connections. It might help to clarify the order of preferences concerning no-TLS, O-TLS, OO-TLS, and I-TLS. Although RFC9051 [34] suggests clients should connect to both implicit port and STARTTLS port concurrently, it was not very up front on which one is more preferred. Likewise, RFC8314 [39] states that implicit TLS is an alternative security mechanism to STARTTLS, without explaining that one might be better than the other. Our recommendation is that RFC9051 could clarify on which use of TLS is more preferred, and together with the aforementioned prohibition of no-TLS, this can potentially avoid the security downgrade problems caused by concurrent auto-detect.

Do not send credentials aggressively (and honor partial countermeasures). Furthermore, we recommend vendors to make sure their clients are not aggressively sending the user credentials together with the `LOGIN` command. The time saving of 1–2 round trips is somewhat insignificant to the user, and it might be better to instead give the server-side partial countermeasures a chance to function, which can at least defend against a type ❶ attacker. Clients should also honor the hints they received from servers.

VII. Related Work

Numerous research efforts have attempted to measure different aspects of the email ecosystems. An early work [16] presents a measurement of SMTP extension usage in the wild, as well as the adoption of STARTTLS on SMTP servers. Meanwhile, [24] presents an Internet-wide analysis of TLS-based email protocol deployments as well as their authentication mechanisms and certificate chain validations. It revealed that most mail servers do not validate the identity of their peers, leaving them vulnerable to MITM attacks. Similar findings have been observed in popular email providers as well [17]. Another Internet-wide scanning of mail servers, with a focus on TLS session parameters including TLS versions, ciphersuites, key sizes and certificate chains, is presented in [32]. Researchers have also studied the problem of managing DNS-based Authentication of Named Entities (DANE) for MTAs, discovering pervasive misconfigurations [29] and management pain points [28] through longitudinal measurements. Another recent study [5] also includes a large-scale Internet measurement and testing email providers. It finds that STARTTLS is now widely supported by mail servers, but many of their certificates are still invalid.

Our study can be seen as orthogonal to these studies, because it mainly considers connections with MUAs, whereas the Internet-wide measurements also include connections between other types of email agents. It thus makes sense that nearly one-third of MTAs in general are found to have bad certificates [5], but the numbers are much lower when we focus on a smaller population of MUA-facing mail servers (Section V-B). The security downgrade problem considered in this paper bears some resemblance to the problem of TLS version downgrade studied by recent work [30], [50], albeit more sophisticated due to the variety of email protocols and auto-detect features of MUAs. Our analysis of real-world setup guides was partly inspired by previous work on enterprise Wi-Fi [4], [25], [48], [53] and academic VPNs [55].

VIII. Conclusion

We present a comprehensive study on the email ecosystem, which considers client-side implementations, setup guides offered by IT admins, and server-side deployments. Our study has identified a number of critical security flaws that can lead to credential theft. On the client-side, a number of MUAs show improper handling of security downgrade and certificate validation. Additionally, the auto-detect features and UI designs of some client apps can trick users into an inadvertent loss of security. Concerning setup guides, many are not explicit in instructing users on how to tighten the security of their configurations, and some blatantly prescribe insecure practices such as accepting all certificates. Finally, the server-side deployments are generally encouraging, which suggests that the client apps and setup guides might actually be the weakest links of the ecosystem. Based on these results, we give concrete suggestions that can improve the security of email users.

IX. Acknowledgments

We thank the anonymous reviewers and shepherd for helping us improve our paper. This work was supported in part by a grant from the Research Grants Council (RGC) of Hong Kong (Project No.: CUHK 24205021), SIEF 3135517, FITE 3200262, and Direct Grant 4055233 from CUHK, as well as grants from the CUHK IE department (project code: GRF/22/SYC and GRF/23/SYC).

References

[1] "Open email survey," 2020. [Online]. Available: https://openemailsurvey.org/

[2] "Dovecot the secure imap server," Sep 2023. [Online]. Available: https://www.dovecot.org/

[3] "Mail (mx) server survey," 2024. [Online]. Available: http://www.securityspace.com/s_survey/data/man.202403/mxsurvey.html

[4] A. Bartoli, E. Medvet, and F. Onesti, "Evil twins and wpa2 enterprise: A coming security disaster?" *Computers & Security*, vol. 74, pp. 1–11, 2018.

[5] B. Blechschmidt and B. Stock, "Extended hell(o): A comprehensive Large-Scale study on email confidentiality and integrity mechanisms in the wild," in *32nd USENIX Security Symposium (USENIX Security 23)*. Anaheim, CA: USENIX Association, Aug. 2023, pp. 4895–4912. [Online]. Available: https://www.usenix.org/conference/usenixsecurity23/presentation/blechschmidt

[6] B. Bucksch, "Mail Autoconfig," Internet Engineering Task Force, Internet-Draft draft-ietf-mailmaint-autoconfig-00, Sep. 2024, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-mailmaint-autoconfig/00/

[7] L. Ceci, "Number of e-mail users worldwide 2026," Jan 2024. [Online]. Available: https://www.statista.com/statistics/255080/number-of-e-mail-users-worldwide/

[8] J. Chen, V. Paxson, and J. Jiang, "Composition kills: A case study of email sender authentication," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/chen-jianjun

[9] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 5280 (Proposed Standard), RFC Editor, Fremont, CA, USA, May 2008, updated by RFCs 6818, 8398, 8399. [Online]. Available: https://www.rfc-editor.org/rfc/rfc5280.txt

[10] A. Cortesi, M. Hils, T. Kriechbaumer, and contributors, "mitmproxy: A free and open source interactive HTTPS proxy," 2010–, [Version 9.0]. [Online]. Available: https://mitmproxy.org/

[11] C. Daboo, "Use of SRV Records for Locating Email Submission/Access Services," RFC 6186 (Proposed Standard), RFC Editor, Fremont, CA, USA, Mar. 2011, updated by RFCs 8314, 8553. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6186.txt

[12] X. d. C. de Carnavalet and M. Mannan, "Killed by proxy: Analyzing client-end TLS interception software," in *Network and Distributed System Security Symposium*, 2016.

[13] J. Debnath, S. Y. Chau, and O. Chowdhury, "When tls meets proxy on mobile," in *Applied Cryptography and Network Security: 18th International Conference, ACNS 2020*. Springer, 2020, pp. 387–407.

[14] V. Dukhovni, "Opportunistic Security: Some Protection Most of the Time," RFC 7435 (Informational), RFC Editor, Fremont, CA, USA, Dec. 2014. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7435.txt

[15] V. Dukhovni and W. Hardaker, "The DNS-Based Authentication of Named Entities (DANE) Protocol: Updates and Operational Guidance," RFC 7671 (Proposed Standard), RFC Editor, Fremont, CA, USA, Oct. 2015. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7671.txt

[16] Z. Durumeric, D. Adrian, A. Mirian, J. Kasten, E. Bursztein, N. Lidzborski, K. Thomas, V. Eranti, M. Bailey, and J. A. Halderman, "Neither snow nor rain nor mitm...: An empirical analysis of email delivery security," in *Proceedings of the 2015 Internet Measurement Conference*, ser. IMC '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 27–39. [Online]. Available: https://doi.org/10.1145/2815675.2815695

[17] I. D. Foster, J. Larson, M. Masich, A. C. Snoeren, S. Savage, and K. Levchenko, "Security by any other name: On the effectiveness of provider based email security," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: Association for Computing Machinery, 2015, pp. 450–464. [Online]. Available: https://doi.org/10.1145/2810103.2813607

[18] GoDaddy, "Repository a collection of important certificate documentation." [Online]. Available: https://certs.godaddy.com/repository

[19] Google, "Add gmail to another email client." [Online]. Available: https://support.google.com/mail/answer/7126229?hl=en&ref_topic=7280141&sjid=17918740399832912769-AP

[20] ——, "Set up gmail with a third-party email client." [Online]. Available: https://support.google.com/a/answer/9003945?hl=en

[21] ——, "Custom search json api," 2023. [Online]. Available: https://developers.google.com/custom-search/v1/overview

[22] ——, "Oauth 2.0 mechanism," 2023. [Online]. Available: https://developers.google.com/gmail/imap/xoauth2-protocol

[23] P. Hoffman, "SMTP Service Extension for Secure SMTP over Transport Layer Security," RFC 3207 (Proposed Standard), RFC Editor, Fremont, CA, USA, Feb. 2002, updated by RFC 7817. [Online]. Available: https://www.rfc-editor.org/rfc/rfc3207.txt

[24] R. Holz, J. Amann, O. Mehani, M. Wachs, and M. Kâafar, "Tls in the wild: an internet-wide analysis of tls-based protocols for electronic communication," in *NDSS 2016*. San Diego, California, USA: Internet Society, 2016, pp. 1–15, network and Distributed System Security Symposium 2016, NDSS'16 ; Conference date: 21-02-2016 Through 24-02-2016. [Online]. Available: http://www.ndss-symposium.org/ndss2016/

[25] M. H. Hue, J. Debnath, K. M. Leung, L. Li, M. Minaei, M. H. Mazhar, K. Xian, E. Hoque, O. Chowdhury, and S. Y. Chau, "All your credentials are belong to us: On insecure wpa2-enterprise configurations," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: ACM, 2021, pp. 1100–1117.

[26] A. Inc., "App analytics - app store connect," 2024. [Online]. Available: https://developer.apple.com/app-store-connect/analytics/

[27] J. Klensin, "Simple Mail Transfer Protocol," RFC 5321 (Draft Standard), RFC Editor, Fremont, CA, USA, Oct. 2008, updated by RFC 7504. [Online]. Available: https://www.rfc-editor.org/rfc/rfc5321.txt

[28] H. Lee, M. I. Ashiq, M. Müller, R. van Rijswijk-Deij, T. T. Kwon, and T. Chung, "Under the hood of DANE mismanagement in SMTP," in *31st USENIX Security Symposium (USENIX Security 22)*, Boston, MA, 2022, pp. 1–16. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/lee

[29] H. Lee, A. Gireesh, R. van Rijswijk-Deij, T. T. Kwon, and T. Chung, "A longitudinal and comprehensive study of the DANE ecosystem in email," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020, pp. 613–630. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/lee-hyeonmin

[30] S. Lee, Y. Shin, and J. Hur, "Return of version downgrade attack in the era of tls 1.3," in *Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 157–168. [Online]. Available: https://doi.org/10.1145/3386367.3431310

[31] G. Leurent and T. Peyrin, "From collisions to chosen-prefix collisions - application to full sha-1," pp. pages 527–555, 2019.

[32] W. Mayer, A. Zauner, M. Schmiedecker, and M. Huber, "No need for black chambers: Testing tls in the e-mail ecosystem at large," in *2016 11th International Conference on Availability, Reliability and Security (ARES)*, 2016, pp. 10–20.

[33] A. Melnikov, "Updated Transport Layer Security (TLS) Server Identity Check Procedure for Email-Related Protocols," RFC 7817 (Proposed Standard), RFC Editor, Fremont, CA, USA, Mar. 2016. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7817.txt

[34] A. Melnikov and B. Leiba, "Internet Message Access Protocol (IMAP) - Version 4rev2," RFC 9051, aug 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc9051

[35] A. Melnikov (Ed.) and K. Zeilenga (Ed.), "Simple Authentication and Security Layer (SASL)," RFC 4422 (Proposed Standard), RFC Editor, Fremont, CA, USA, Jun. 2006. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4422.txt

[36] Microsoft, "Pop, imap, and smtp settings for outlook.com." [Online]. Available: https://support.microsoft.com/en-us/office/pop-imap-and-smtp-settings-for-outlook-com-d088b986-291d-42b8-9564-9c414e2aa040

[37] Microsoft Learn, "Autodiscover service in Exchange Server — learn.microsoft.com," https://learn.microsoft.com/en-us/exchange/architecture/client-access/autodiscover?view=exchserver-2019, [Accessed 28-09-2024].

[38] B. Moeller and A. Langley, "TLS Fallback Signaling Cipher Suite Value (SCSV) for Preventing Protocol Downgrade Attacks," RFC 7507 (Proposed Standard), RFC Editor, Fremont, CA, USA, Apr. 2015. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7507.txt

[39] K. Moore and C. Newman, "Cleartext Considered Obsolete: Use of Transport Layer Security (TLS) for Email Submission and Access," RFC 8314 (Proposed Standard), RFC Editor, Fremont, CA, USA, Jan. 2018. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8314.txt

[40] K. Moriarty and S. Farrell, "Deprecating tls 1.0 and tls 1.1," Mar. 2021.

[41] playsearch.kaki87.net, "Kplaysearch," 2022. [Online]. Available: https://git.kaki87.net/playsearch.kaki87.net/v1

[42] D. Poddebniak, F. Ising, H. Böck, and S. Schinzel, "Why TLS is better without STARTTLS: A security analysis of STARTTLS in the email context," in *30th USENIX Security Symposium (USENIX Security 21)*. Vancouver, B.C., Canada: USENIX Association, Aug. 2021, pp. 4365–4382.

[43] S. Pourali, X. Yu, L. Zhao, M. Mannan, and A. Youssef, "Racing for TLS certificate validation: A hijacker's guide to the android TLS galaxy," in *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024,

pp. 683–700. [Online]. Available: https://www.usenix.org/conference/usenixsecurity24/presentation/pourali

[44] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446 (Proposed Standard), RFC Editor, Fremont, CA, USA, Aug. 2018. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8446.txt

[45] P. Saint-Andre and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)," RFC 6125 (Proposed Standard), RFC Editor, Fremont, CA, USA, Mar. 2011. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6125.txt

[46] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP," RFC 6960 (Proposed Standard), RFC Editor, Fremont, CA, USA, Jun. 2013, updated by RFC 8954. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6960.txt

[47] D. Schinazi and T. Pauly, "Happy Eyeballs Version 2: Better Connectivity Using Concurrency," RFC 8305 (Proposed Standard), RFC Editor, Fremont, CA, USA, Dec. 2017. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8305.txt

[48] L. Song, Q. Wang, S. Jia, J. Lin, L. Lu, and Y. Fu, "You cannot fully trust your device: An empirical study of client-side certificate validation in wpa2-enterprise networks," in *2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2022, pp. 266–273.

[49] Svpsiva, "Authenticate an imap, pop or smtp connection using oauth," 2023. [Online]. Available: https://learn.microsoft.com/en-us/exchange/client-developer/legacy-protocols/how-to-authenticate-an-imap-pop-smtp-application-by-using-oauth

[50] K. F. Tang, K. L. Wu, and S. Y. Chau, "Investigating tls version downgrade in enterprise software," in *The 14th ACM Conference on Data and Application Security and Privacy*, ser. CODASPY '24. ACM, 2024.

[51] W. Venema, "The postfix home page," 2011. [Online]. Available: http://www.postfix.org

[52] L. Waked, M. Mannan, and A. Youssef, "To intercept or not to intercept: Analyzing tls interception in network appliances," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, 2018, pp. 399–412.

[53] K. Wang, Y. Zheng, Q. Zhang, G. Bai, M. Qin, D. Zhang, and J. S. Dong, "Assessing certificate validation user interfaces of wpa supplicants," in *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, 2022, pp. 501–513.

[54] K. L. Wu, M. H. Hue, N. M. Poon, K. M. Leung, W. Y. Po, K. T. Wong, S. H. Hui, and S. Y. Chau, "Back to school: On the (in)security of academic VPNs," in *32nd USENIX Security Symposium (USENIX Security 23)*. Anaheim, CA: USENIX Association, Aug. 2023, pp. 5737–5754.

[55] K. L. Wu, M. H. Hue, K. F. Tang, and S. Y. Chau, "The devil is in the details: Hidden problems of client-side enterprise wi-fi configurators," in *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec'23)*. New York, NY: ACM, 2023, (Best Paper Award from ACM WiSec '23).

[56] K. Zeilenga (Ed.), "The PLAIN Simple Authentication and Security Layer (SASL) Mechanism," RFC 4616 (Proposed Standard), RFC Editor, Fremont, CA, USA, Aug. 2006. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4616.txt

[57] ZeroSSL, "What type of ssl certificates does zerossl provide?" 2020. [Online]. Available: https://help.zerossl.com/hc/en-us/articles/360060119453-What-Type-of-SSL-Certificates-Does-ZeroSSL-Provide

APPENDIX

A. Support for outdated TLS versions

During the course of our experiments, we discovered that some Windows clients claim to support deprecated versions of TLS in their email connection settings. For instance, nPOP and Becky! on Windows allow users to choose SSLv2 and SSLv3 for their email connections. We further tested these options and found that those old TLS versions are actually no longer supported. Instead, these clients will use TLSv1.0 if either SSLv2 or SSLv3 is selected. According to RFC8996 [40], both TLSv1.0 and TLSv1.1 are generally deprecated, in part due to their dependence on weak cryptographic algorithms and lack of support for recommended cipher suites. In the context of email protocols, RFC8314 [39] also recommends TLSv1.2 or newer should be used. To help users improve the overall security posture of their email connections, it is advisable for email clients to remove these configuration options as well as support for outdated versions of TLS.

TABLE X. TLS VERSIONS DISPLAYED BY CLIENTS

Clients	TLS versions displayed	Supported?	Remarks
W11 nPOP	SSL 2.0	✗	/
	SSL 3.0	✗	/
	TLS 1.0	✓	/
	TLS 1.1	✓	/
	TLS 1.2	✓	/
W11 Becky!	SSL 2.0	✗	/
	SSL 3.0	✗	Prohibited
	TLS 1.0	✓	Not recommended
	TLS 1.1	✓	/
	TLS 1.2	✓	/
	TLS 1.3	✓	/

B. Certificate revocation test cases and findings

As mentioned in RFC7817 [33], all email clients must check for the understanding of server's identity against the identity presented in server's certificate, which the details are listed in Section 6 of RFC5280 [9]. Although RFC5280 specified the checking on certificate revocation status, it stated that one of the method is to obtain the appropriate Certificate Revocation List (CRL) to retrieve the status information. However, in reality, some commercial CAs do not support CRL checking. Instead, they use Online Certificate Status Protocol (OCSP) [46] to determine the revocation status of a digital certificate, which the use of OCSP on checking certificate revocation status is not mentioned in Section 6 of RFC5280.

To further test with this discrepancy, we purchased two certificates from ZeroSSL and GoDaddy, and revoked them immediately. We waited for two days for GoDaddy to update the CRL, so that the revocation status of the certificate will be update to "revoked". We denoted the certificate purchased from ZeroSSL as C_5 since ZeroSSL only supports OCSP for certificate revocation [57], while the certificate purchased from GoDaddy (C_6) supports both CRL and OCSP methods. [18]

The result is listed on Table XI. Among 49 email clients, 35 email clients accepted either C_5 or C_6, or both of them. None of the tested Android clients performed certificate revocation status checking. Only 3 clients in Windows 11 rejected the revoked certificates. For Windows Outlook, if the configuration is done on an unstable network, it will accept both of the certificates. Otherwise, it will reject the certificates directly under a stable network connection. Therefore, we can only mark it as inconclusive result. This shows that most mail servers do not have a strict checking on certificate revocation.

C. Usage of DANE in mail servers

DNS-based Authentication of Named Entities (DANE) allows a server certificate to be bounded to the domain names

TABLE XI. REVOKED CERTIFICATE CHECKS AGAINST CLIENT APPLICATIONS

Client	Implicit TLS			STARTTLS		
	SMTP	IMAP	POP3	SMTP	IMAP	POP3
Android						
Boxer - Workspace ONE	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6
Blue Mail - Email & Calendar	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6
Email Aqua Mail - Fast, Secure	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6
Edison Email - Fast & Secure Mail	C_5-C_6	C_5-C_6	–	C_5-C_6	C_5-C_6	–
Gmail	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6
K-9 Mail	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6
mail.ru	–	C_5-C_6	–	C_5-C_6	–	–
Microsoft Outlook	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6
myMail: for Outlook & Yahoo	C_5-C_6	C_5-C_6	–	–	–	–
Samsung Email	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6
Spark Mail	C_5-C_6	C_5-C_6	–	C_5-C_6	C_5-C_6	–
Type App mail - email app	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6
iOS						
Apple Mail	✓	✓	✓	–	–	–
Blue Mail - Email & Calendar	✓	✓	✓	✓	✓	∞
Boxer - Workspace ONE	C_5-C_6	C_5-C_6	–	C_5?, C_6	C_5-C_6	–
Email Aqua Mail - Fast, Secure	✓	✓	✓	✓	C_5-C_6	✓
Email - Edison Mail	✓	C_5-C_6	–	✓	–	–
Gmail	✓	✓	–	✓	✓	–
Mail Master by Netease	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6
mail.ru	–	C_5-C_6	–	C_5-C_6	–	–
Microsoft Outlook	✓	✓	–	✓	✓	–
Spark Mail	✓	✓	–	✓	C_5-C_6	–
Type App mail - email app	✓	✓	✓	✓	✓	∞

Client	Implicit TLS			STARTTLS		
	SMTP	IMAP	POP3	SMTP	IMAP	POP3
macOS						
Apple Mail	✓	✓	✓	–	–	–
Blue Mail - Email | Calendar	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6
Edison Mail	✓	✓	–	✓	✓	–
eM Client	C_5-C_6	C_5-C_6	C_5-C_6	–	–	–
Foxmail	C_5-C_6	–	C_5-C_6	C_5-C_6	–	–
Mail Master by Netease	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6
Microsoft Outlook	✓	✓	✓	✓	✓	✓
SeaMonkey (formerly Netscape)	✓	✓	✓	✓	✓	✓
Spark Classic - Email App	✓	✓	–	✓	C_5-C_6	–
Spark Desktop	✓	✓	–	✓	C_5-C_6	–
Sylpheed	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6
Thunderbird	✓	✓	✓	✓	✓	✓
TypeApp	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6
W11						
Becky!	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6
Blue Mail - Email & Calendar	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6
Claws Mail	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6
eM Client	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6
Foxmail	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	–	–
Microsoft Outlook	?	?	?	?	?	–
nPOP	C_5-C_6	–	C_5-C_6	C_5-C_6	–	C_5-C_6
SeaMonkey (formerly Netscape)	✓	✓	✓	✓	✓	✓
Spark Email for Windows	C_5-C_6	C_5-C_6	–	C_5-C_6	C_5-C_6	–
Sylpheed	C_6	C_6	C_6	C_6	C_6	C_6
Thunderbird	✓	✓	✓	✓	✓	✓
TypeApp for Windows	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6	C_5-C_6
Windows Mail	✓	✓	✓	✓	✓	✓

✓: Not vulnerable –: Does not support the protocol ?: Inconclusive ∞: Infinite loop C_5: ZeroSSL certificate (OSCP only) C_6: GoDaddy certificate

using DNSSEC, so that a client can authenticate a particular server without CA. It is a `TLSA` resource record published in DNS and its standard is defined in RFC7671 [15]. To investigate whether the usage of DANE is common in mail servers, we wrote a short Python script and queried for the DNS `TLSA` records. We found that only 4 universities have deployed DANE with `TLSA` records found. This implies the adoption of DANE is not common in educational providers.

D. Setup guides from email service providers

While this paper focused on custom email servers deployed by IT admins in the universities, some universities purchase email service from vendors the mail servers are maintained by the service providers, such as Office365 and Google Workspace. Different from the traditional login methods (*e.g.*, `AUTH PLAIN`), those service providers tend to use OAuth2.0 mechanism to login to their own organisation [20], [36]. In Office365 setup guide, it offered manual server settings to user and forced users to use OAuth2.0 to access their account. However, the setup guide does not provide detailed steps on how user should configure their account on a third-party mail client. While for Google setup guide [19], it suggested users to simply look for an option "Sign in with Google", which will redirect users to login with their organisation using OAuth2.0. However, Google also prescribed a detailed setup guide to IT admins and provided detailed steps on how a user should look for the aforementioned option. Nevertheless, Google does not explicitly state the manual server settings, instead it only includes the information under "Set up Gmail with the older version of Outlook". These OAuth-based login instructions have simplified the manual configuration process of the mail account setup in third-party client apps, as they assumed that the security of using OAuth login method is guaranteed.

E. Email clients tested

Table XII shows the names, origins, and version numbers of the email clients tested in this work.

TABLE XII. LIST OF THE TESTED EMAIL CLIENTS

Application	OS	Version	Source
Gmail	Android 13	2023.11.26.586591930.release	Google Play
	iOS 15.7.5	6.0.23.1127	App Store
Samsung Email	Android 13	6.1.90.16	Google Play
Email Aqua Mail - Fast, Secure	Android 13	1.49.2	Google Play
	iOS 15.7.5	1.26.5	App Store
myMail: for Outlook & Yahoo	Android 13	14.95.0.52229	Google Play
K-9 Mail	Android 13	6.603	Google Play
Spark Mail	Android 13	3.7.1	Google Play
	iOS 15.7.5	3.7.3	App Store
Spark Classic - Email App	macOS 12.4	2.11.37.938	App Store
Spark Desktop	macOS 12.4	3.6.5	App Store
Spark Email for Windows	W11	3.12.0.63910	https://sparkmailapp.com/windows
Type App mail - email app	Android 13	1.9.37	Google Play
	iOS 15.7.5	2.0.28	App Store
TypeApp for Windows	W11	3.21.208.483	https://typeapp.com/windows/
TypeApp	macOS 12.4	3.21.208	App Store
Boxer - Workspace ONE	Android 13	23.11.0.5	Google Play
	iOS 15.7.5	23.11	App Store
Microsoft Outlook	Android 13	4.5352.1	Google Play
	iOS 15.7.5	4.2327.0	App Store
	W11	16.0.16924.20150	built-in
	macOS 12.4	16.8	App Store
Blue Mail - Email & Calendar	Android 13	1.9.42	Google Play
	iOS 15.7.5	4.25.3	App Store
	W11	1.137.3.0	MSFT store
Blue Mail - Email | Calendar	macOS 12.4	1.137.3	App Store
Mail Master by Netease	iOS 15.7.5	7.18.3	App Store
	macOS 12.4	4.17.22	App Store
Email - Fast & Secure Mail	Android 13	1.52.0	Google Play
Email - Edison Mail	iOS 15.7.5	1.52.01	App Store
	macOS 12.4	1.24.6	App Store
myMail: for Outlook & Yahoo	Android 13	14.95.0.52229	Google Play
myMail: email app for Gmail	iOS 15.7.5	14.64	App Store
Apple Mail	iOS 15.7.5	iOS 15.7.5	built-in
	macOS 12.4	16.0 (3696.100.31)	built-in
Becky!	W11	2.8.1.5	https://www.rimarts.co.jp/becky.htm#download
SeaMonkey	W11	2.53.18	https://www.seamonkey-project.org/releases/
	macOS 12.4	2.53.18	https://www.seamonkey-project.org/releases/
Thunderbird	W11	115.2.3	https://www.thunderbird.net/en-US/download/
	macOS 12.4	102.14.0	https://www.thunderbird.net/en-US/download/
Windows Mail & Calendar	W11	16005.14326.21768.0	built-in
nPOP	W11	1.2.6	https://nakka.com/soft/npop/index_eng.html
Claws Mail	W11	4.2.0	https://www.claws-mail.org/win32/
Sylpheed	W11	3.7.0	https://sylpheed.sraoss.jp/en/download.html
	macOS 12.4	3.7.0	https://formulae.brew.sh/formula/sylpheed
eM Client	W11	9.2.2157.0	https://www.emclient.com
	macOS 12.4	9.2.2041	https://www.emclient.com