USENIX

THE ADVANCED COMPUTING
SYSTEMS ASSOCIATION

Email Spoofing with SMTP Smuggling: How the Shared Email Infrastructures Magnify this Vulnerability

Chuhan Wang, *Southeast University and Tsinghua University;* Chenkai Wang, *University of Illinois Urbana-Champaign;* Songyi Yang, *Tsinghua University;* Sophia Liu, *University of Illinois Urbana-Champaign;* Jianjun Chen, *Tsinghua University and Zhongguancun Laboratory;* Haixin Duan, *Tsinghua University and Quan Cheng Laboratory;* Gang Wang, *University of Illinois Urbana-Champaign*

https://www.usenix.org/conference/usenixsecurity25/presentation/wang-chuhan

This paper is included in the Proceedings of the
34th USENIX Security Symposium.

August 13–15, 2025 • Seattle, WA, USA

978-1-939133-52-6

Open access to the Proceedings of the
34th USENIX Security Symposium is sponsored by USENIX.

ARTIFACT EVALUATED
usenix ASSOCIATION
AVAILABLE

Email Spoofing with SMTP Smuggling: How the Shared Email Infrastructures Magnify this Vulnerability

Chuhan Wang [£†], Chenkai Wang[‡], Songyi Yang[†], Sophia Liu[‡],

Jianjun Chen[†¶✉] *, Haixin Duan[§†✉], and Gang Wang[‡]

[£]*Southeast University*, [†]*Tsinghua University*, [‡]*University of Illinois Urbana-Champaign*,
[¶]*Zhongguancun Laboratory*, [§]*Quan Cheng Laboratory*,

Abstract

Email spoofing is a critical technique used in phishing attacks to impersonate a trusted sender. SMTP smuggling is a new vulnerability that allows adversaries to perform email spoofing while *bypassing* existing authentication protocols such as SPF and DMARC. While SMTP smuggling has been publicly disclosed since 2023, its impact has not been comprehensively evaluated and the effectiveness of the community's mitigation strategies is yet unknown. In this paper, we present an in-depth study of SMTP smuggling vulnerabilities, supported by empirical measurements of public email services, open-source email software, and email security gateways. More importantly, for the first time, we explored how to perform measurements on *private* email services ethically, with new methodologies combining user studies, a DKIM side channel, and a non-intrusive testing method. Collectively, we found that 19 public email services, 1,577 private email services, five open-source email software, and one email gateway were still vulnerable to SMTP smuggling (and/or our new variants). In addition, our results showed that the centralization of email infrastructures (e.g., shared SFP records, commonly used email software/gateways) has amplified the impact of SMTP smuggling. Adversaries can spoof highly reputable domains through free-to-register email accounts while bypassing sender authentication. We provided suggestions on short-term and long-term solutions to mitigate this threat. To further aid email administrators, we developed an online service to help self-diagnosis of SMTP smuggling vulnerabilities.

1 Introduction

Email services are still the cornerstone of communication for individuals and organizations today [5]. With billions of emails sent daily, ensuring the security and integrity of email systems is paramount. The Simple Mail Transfer Protocol (SMTP) [23] is the foundation of email transmission and has undergone numerous enhancements to address security concerns. SMTP extensions such as SPF (Sender Policy Framework) [22], DKIM (DomainKeys Identified Mail) [9], and DMARC (Domain-based Message Authentication, Reporting & Conformance) [24] have been developed to bolster email security and mitigate threats such as phishing [16, 33] and spoofing [19, 50, 52].

Despite these advancements, vulnerabilities continue to surface, exposing email systems to new attacks [7, 27, 44]. One such emerging threat is "SMTP Smuggling", a sophisticated email spoofing technique proposed by researchers from SEC Consult in December 2023 [30]. SMTP smuggling is executed by embedding SMTP commands within the email body, allowing attackers to send multiple emails in a single SMTP session and spoof trusted entities without being detected. This attack exploits the inconsistencies in the processing of the end-of-data indicator of SMTP `DATA` command between the sending and receiving Mail Transfer Agent (MTA) servers.

The implications of SMTP smuggling are profound. It allows attackers to perform email spoofing while *bypassing existing authentication mechanisms* (SPF and DMARC). This enables attackers to impersonate any sender within the same domain (e.g., the CEO's account within a company) to send spoofing/phishing emails. The risk amplifies if the vulnerable sending service is an email service provider, allowing attackers to impersonate all clients of this service. Importantly, since SPF relies on IPs for sender authentication, and many clients incorporate the provider's SPF record into their own [54], spoofed emails (sent from the provider's IP) can pass both SPF and DMARC verification (see details in Section 2.3).

Research Questions. This paper is motivated by two research questions. First, we suspect the impact of SMTP smuggling has been *underestimated*. More specifically, the risk of SMTP smuggling may have been amplified by the shared email infrastructures, which is the hypothesis to be verified. Second, SMTP smuggling attacks have been publicly disclosed for more than eight months. We seek to understand how involved parties, including email services and open-source

*✉ Corresponding authors:{jianjun, duanhx}@tsinghua.edu.cn.

Figure 1: Spoofing Email Sent via SMTP Smuggling: The attacker registers a *free* Gmail account and uses it to send a spoofing email, impersonating `admin@youtube.com`. The spoofing email passed SPF and DMARC authentication.

email software projects, have responded to this vulnerability since its disclosure. As the formation of SMTP smuggling itself is an intricate issue that involves multiple parties, how to correctly mitigate the issue is worth studying and measuring.

Our Study. We made key contributions to understanding and mitigating SMTP smuggling vulnerabilities. *First*, we conducted a comprehensive measurement across a broad spectrum of email infrastructures, including public email services, open-source email software, and security gateways. By exploring both known and *new variants* of SMTP smuggling tactics, we performed testing on 22 public email services, five open-source email software packages, and two commercial email gateways. Our extensive experimentation revealed that (1) 18 public email providers (e.g., Gmail, Outlook, Yahoo) and five email software packages (e.g., Postfix) were still vulnerable on the *sending* side, and (2) eight public email providers (e.g., Sina, Sohu, Yandex), two email software packages (i.e., Haraka, Axigen) and one email gateway (i.e., TrendMicro) were still vulnerable on the *receiving* side.

Second, for the first time, we explored how to ethically perform measurements on *private* email services and proposed a new set of methodologies. By combining user studies (i.e., recruiting users to help with testing) with a DKIM verification side-channel and a non-intrusive testing method (automated testing without user cooperation), we performed SMTP smuggling tests over a large number of private email systems. This allowed us to assess the vulnerability while minimizing the disruption to existing services (IRB-approved; detailed ethical consideration is in Section 9). Our user study revealed that 23 out of 48 university email systems were vulnerable, and the non-intrusive test showed that 1,577 of the Tranco Top 10,000 domains [25] were susceptible to the attack.

Third, we analyzed how the shared email infrastructures had amplified the impact of SMTP smuggling. First, we observed that the shared SPF infrastructures had enabled attackers to spoof well-known domains through *free/public* email services. For example, we found that attackers could spoof `admin@youtube.com` or `admin@zoom.us` through a free Gmail account (similarly, they could spoof `admin@microsoft.com` through a free Outlook account). A proof-of-concept example is shown in Figure 1. In addition, through SMTP banner analysis, we showed that the vulnerable domains were highly associated with a small set of widely used email infrastructures. This included popular email software (e.g., Postfix) and security gateways (e.g., Proofpoint). These insights underscored the urgent need for improved security practices and updates of these core email infrastructures to mitigate the threat.

Responsible Disclosure. We have sent vulnerability reports to all vulnerable domains identified in the non-intrusive test. In addition, we reached out to all other affected parties, including public email services, university email services, email software developers, and email gateway services. A detailed discussion is presented in Section 6.4.

Contributions. We made the following contributions:

- Experiment: We conducted a comprehensive measurement of SMTP smuggling over a wide range of email infrastructures, including public email services, open-source email software, and email gateways.
- Methodology: We proposed new measurement methodologies to test *private* email systems, including a user study, a DKIM verification side-channel, and a non-intrusive test method.
- Results: Our results showed that 19 public email services (combining sending and receiving sides), 1,577 private email services, five open-source email software, and one email gateway were still vulnerable to SMTP smuggling.
- Insights: Our results revealed that the centralization of email services magnified SMTP smuggling attacks. Attackers could spoof well-known domains by exploiting SMTP smuggling and the shared SPF infrastructure.

2 Background

2.1 Simple Mail Transfer Protocol

Simple Mail Transfer Protocol (SMTP) [23] is the fundamental email transfer protocol for today's email systems. Mail Transfer Agents (MTAs) are entities that use SMTP to transport emails (i.e., servers). Mail User Agents (MUAs) are the origins and final consumers of email messages (i.e., clients). SMTP operates under plain-text commands, using Carriage Return and Line Feed (<CR><LF> or "`\r\n`")[1] as the des-

[1] <CR> and <LF> are also represented as `\r` and `\n`. If not otherwise stated, we use `\r` and `\n` for the rest of the paper for consistency.

ignated line delimiter. Figure 5 (in the Appendix) shows a typical SMTP conversation. It begins with the sending MTA issuing the "EHLO" command. To send an email, the sender then sends an SMTP envelope with "MAIL FROM:", "RCPT TO:" and at last, "DATA" which starts transmission of the email body [41]. At the end of data transmission, the sender transmits <CR><LF>.<CR><LF> (i.e., "\r\n.\r\n") as the end-of-data indicator. After that, the sender can choose to send the next SMTP envelope or send "QUIT" to end the conversation.

Pipelining. Command pipelining is an SMTP service extension widely adopted by email services in the real world [14]. With command pipelining, SMTP senders can send multiple commands in batch, eliminating the need to wait for status code replies. Servers that support pipelining include "PIPELINING" in the 250 status code when replying to the client's EHLO command.

2.2 Email Authentication Mechanisms

SMTP [23] lacks built-in authentication features, making it vulnerable to sender spoofing attacks. As countermeasures, multiple email security extensions were introduced in the past two decades, including SPF, DKIM, and DMARC.

SPF. Sender Policy Framework (SPF) [22] authenticates sender domains based on *IP addresses*, using the DNS infrastructure. First, sender domain administrators can publish a set of IP addresses that are allowed to act as senders for their domain. Then, upon receiving connections from the sender, the receiving MTA can check whether the sending MTA's IP address is permitted to send emails under the domain they claimed in EHLO and MAIL FROM.

DKIM. DomainKeys Identified Mail (DKIM) [9] cryptographically signs email messages using public-key cryptography, with a public key hosted in the DNS. DKIM signatures are sent in email messages as a header, which signs a selected list of header fields as well as part of the message body. The DKIM signature headers also contain fields that lead to the DNS record where the sender's public key is hosted. The MTA or MUA that verifies the signature would first look up the DNS record for the public key and then perform the cryptographic verification.

DMARC. Domain-based Message Authentication, Reporting and Conformance (DMARC) [24] is built on top of SPF and DKIM, allowing domains to publish policies on how recipients should handle authentication failures and report results back to senders. DMARC also verifies *domain alignments* between the domain name in the "From:" field of the email header and the domain name in the DKIM/SPF authentication results (i.e., the "MAIL FROM"). This ensures the user-visible sender address in "From:" is consistent with the authenticated one. For DMARC to pass, it requires two things to be true: (1) DKIM or SPF passes the authentication, and (2) the authenticated domain aligns with the From: domain.

Figure 2: SMTP Smuggling Attack Example: The attacker (alice@a.com) spoofs a trusted sender (admin@a.com) to send an email to the victim (bob@b.com). This is done by embedding the second (attack) email within the first (legit) email's body and placing a malicious end-of-data indicator \n.\n at the beginning of the second email.

2.3 SMTP Smuggling Attack

The above security extensions, if configured correctly, can detect email sender spoofing attacks. However, researchers recently discovered *SMTP smuggling*, which can circumvent anti-spoofing mechanisms. SMTP smuggling was introduced by researchers of SEC Consult in December 2023 [30]. This attack exploits the inconsistent processing of SMTP commands between the sending and receiving MTA servers. The key idea is to embed SMTP commands into the body of an email, tricking the receiving MTA into misinterpreting the data (email body) as code (SMTP command). This allows the attacker to send two emails within a single SMTP session, "smuggling" a malicious email (i.e., the second email) to perform spoofing.

SMTP smuggling exploits the "end-of-data" indicator in the STMP DATA command. The RFC documents [23] mandate that the end-of-data indicator should be in the form of "\r\n.\r\n". However, historical differences in operating systems and email server implementations have led to variations in line break handling. Some email systems accept other formats (e.g., \n.\n) for compatibility, creating an opportunity for SMTP smuggling attacks.

An Attack Example. We use Figure 2 to explain how it works. In this example, the attacker (alice@a.com) sends an email to the victim (bob@b.com). In the email body, the attacker embeds malicious content (in plain text), which is highlighted in red. Note that the first line of the malicious content is \n.\n. When the receiving MTA receives the message, for compatibility reasons mentioned above, it may accept \n.\n as a legitimate end-of-data indicator. As a result, the

receiving MTA is led to believe that the first email ends here. More importantly, it continues to read the rest of the content as the SMTP command for sending a second email, due to the SMTP pipeline mechanism. As a result, the original email is split into two emails. The first email (green) is sent legitimately, but the second email (red) can be used to conduct malicious attacks.

Why is This a Serious Threat? Using the smuggled email (the second email), the attacker can perform email spoofing *while bypassing existing anti-spoofing protocols* including both SPF and DMARC. As shown in Figure 2, the receiving server would view the smuggled email as if it was legitimately sent by `a.com` and will use its IP address for SPF authentication. This allows the attacker to spoof any account under `a.com` (e.g., `admin@a.com`) to send emails to the victim. The smuggled email will bypass SPF because the attacker is using an allowed IP of `a.com` for sending. It can bypass DMARC as well because DMARC is marked "pass" if (1) SPF is passed and (2) the authenticated domain name is aligned with that in the `From:` email header (both are `a.com` in this case). This is a mandatory workflow defined by the RFC standard [24] (not a configurable policy of email servers).

The implication is profound. For example, for an enterprise organization (e.g., `apple.com`), any contractors/interns can send emails using their CEO/CFO's sender address to anyone within/outside of the organization, without being detected by anti-spoofing protocols. For email providers, e.g., `gmail.com`, attackers can spoof any user's Gmail account. In addition to spoofing an arbirary account under the same domain name, the original sender and the spoofed address can be under different domain names too. More specifically, as many different domains/organizations have shared SPF infrastructures [54] (e.g., the same IP is allowed to send emails on behalf of *gmail.com*, *youtube.com*, and a wide range of organizations that use Gmail's mailing service), attackers thus can spoof email addresses of another connected organization (see Section 3.3 for more detailed analyses).

The original vulnerability report [30] showed examples of attack payload (e.g., line break indicators that can be misinterpreted) without performing extensive tests on its variants. The testing scope is limited to a small set of *public email services*. More importantly, after the individual disclosure in 2023, it is unclear what actions email providers have taken to mitigate this threat and whether these mitigation strategies are effective. In the rest of the paper, we fill these gaps with extensive measurements of both public and *private* email services and also explore the sources of the vulnerabilities by analyzing the SPF infrastructures, email gateways, and email software.

3 Public Email Service Experiments

To evaluate the impact of SMTP smuggling attacks, we first conducted experiments against public email services.

ID	Payload	ID	Payload
A_1	`\n.\n`	A_8	`\n\x00.\n`
A_2	`\n.\r`	A_9	`\n\x00.\r`
A_3	`\r.\n`	A_{10}	`\r\x00.\n`
A_4	`\r.\r`	A_{11}	`\r\x00.\r`
A_5	`\n.\r\n`	A_{12}	`\n\x00.\r\n`
A_6	`\r.\r\n`	A_{13}	`\r\x00.\r\n`
A_7	`\r\n\x00.\r\n`		

Table 1: SMTP Smuggling Payload List.

3.1 Experiment Methodology

Attack Requirements. A successful SMTP smuggling attack requires that (1) the receiving MTA *misinterprets* the end-of-data indicator and splits the email into two messages, and (2) the sending MTA *ignores* the specially crafted end-of-data indicators placed in the email body by the attacker. Thus, our SMTP smuggling test also includes two parts: a receiving MTA test and a sending MTA test.

Receiving MTA Test. The goal is to analyze how the receiver MTA handles end-of-data indicators other than "`\r\n.\r\n`", and whether the receiver MTA will be misled to split the original email into two separate emails.

Receiving MTA tests require (1) a self-built sending MTA, configured with SPF, DKIM, and DMARC correctly; (2) an account of the target receiving email service. We use our sending MTA to send test emails to the target recipient. We can check the results using the MUA of the target receiving service. If our email has been interpreted as two emails, the target receiving service is considered vulnerable.

Sending MTA Test. The goal is to test whether sending MTA "ignore" the inserted end-of-data indicators in the email body or if they have implemented any defenses. Sending MTA tests require (1) a fully controlled MUA, which can send emails containing testing payload to the sending MTA; (2) a self-hosted receiving MTA, which is used to capture the original data sent by the sending MTA; (3) an account of the target sending email services, which allows us to connect to the sending MTA and send emails.

The test process contains the following steps: (1) We use our custom MUA, written in Python, to connect to the sending MTA. (2) After the sending MTA has verified the account identity, we will send emails containing the identified payload to the target sending MTA, specifying the target recipient as an email address of our self-hosted email server. (3) We will monitor and capture the traffic sent to port 25 of the self-hosted receiving MTA. (4) We will conduct an analysis of the received traffic data to determine whether the sending MTA has made any changes to the payload we constructed. If the sending MTA transmits the original email data *without filtering or modification*, it is considered vulnerable.

Attack Payload Selection. Payload selection is a crucial step in testing for SMTP smuggling vulnerabilities. While

Email Service	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}	A_{13}
qq.com	✗	✗	✗	✗	✗	✗	✂	✂	✂	✂	✂	✂	✂
163.com	✂	✂	✂	✂	✂	✂	✗	✗	✗	✗	✗	✗	✗
sina.com	✓*	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
sohu.com	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
aliyun.com	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
tom.com	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
139.com	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
gmail.com	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
yahoo.com	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
zoho.com	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
aol.com	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
yandex.ru	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
mail.ru	✗	✗	✗	✗	✗	✗	✂	✂	✂	✂	✂	✂	✂
outlook.com	✗	✗	✗	✗	✗	✗	✂	✂	✂	✂	✂	✂	✂
icloud.com	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
runbox.com	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
fastmail.com	✗	✗	✗	✓	✗	✓	✓	✗	✗	✗	✓	✗	✓
cock.li	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
daum.net	✓	✗	✗	✗	✓	✗	✓	✓	✗	✗	✗	✓	✗
rambler.ru	✓	✂	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
naver.com	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
freemail.hu	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗

✓: Vulnerable. ✗: Not Vulnerable. ✗: Rejected. ✂: Truncated.
* : Sina will only display the smuggled email and drop the first legitimate email.

Table 2: Receiving MTAs: Testing Results of 22 Public Email Services. The attack payloads A_1–A_{13} are presented in Table 1.

the original technical report provided some example payloads [30], it lacked a comprehensive assessment. To address this gap, we expanded our testing by fuzzing the most popular open-source email software (e.g., Postfix, Sendmail, Exim) locally within a sandbox environment. Our character fuzzing tests covered a broad range of characters, including all ASCII and extended ASCII characters (`\x00-\xff`) and most Unicode characters using UTF-8 encoding (`\u0000-\uFFFF`). By analyzing how these email software processed various inputs, we identified 13 payloads that effectively exploit SMTP smuggling vulnerabilities. Notably, payloads A_8–A_{13} are new and were not mentioned in the original vulnerability report. Our expanded set of payloads offers a more comprehensive foundation for testing SMTP smuggling vulnerabilities.

To ensure our emails can be delivered to the target services, we have extra considerations, as detailed in Appendix A.

3.2 Testing Public Email Services

Public email services were selected as the primary targets for our experiment due to their wide use and critical role in the global communication infrastructure. We selected 22 email services that offer free use with open registration based on previous work [44]. These 22 mail services collectively provide email services to more than 1 billion users. Additionally, some of these services also offer email services for other domains, and thus their security is important to measure.

Ethics. We are mindful of the ethical implications of our tests and have taken great care to avoid negatively impacting the services/users. Our test is highly controlled: we only send emails to *researcher-owned email accounts*. In addition, all the *spoofed* accounts and domains are also owned/controlled by the researchers. For a given test, we only sent 13 emails per service, at a rate of one email every 30 seconds. The full ethical discussion is in Section 9.1.

Receiving MTA Test Results. Table 2 presents the result of the receiving side. The results indicated that 8 of 22 public email services were vulnerable to SMTP smuggling.

Vulnerable services include popular email providers such as Sina, Sohu, and Yandex. Among these vulnerable email services, Sina dropped the first legitimate email and only displayed the smuggled email, which could be a useful feature for attackers. Fastmail had set a strict policy on bare "`\n`" and rejected some ambiguous emails, but it was still vulnerable to other payloads without bare "`\n`" like "`\r.\r`". Most services such as Gmail, Yahoo, Outlook, and iCloud were found not vulnerable on the *receiving side*. Additionally, some services, such as 163.com and Mail.ru, truncated the smuggled emails, displaying only the first part of the email (before the payload) in the user interface (denoted by a scissor symbol in Table 2).

Sending MTA Test Results. Table 3 presents the result of the sending side. The results indicated that 18 out of 20 public email services were vulnerable to SMTP smuggling

Email Service	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}	A_{13}
qq.com	❍	❍	❍	❍	❍	❍	✂	✂	✂	✂	✂	✂	✂
163.com	✂	✂	✂	✂	✂	✂	✓	✓	✓	✓	✓	✓	✓
sina.com	❑	❍	❍	❍	❑	❍	✓	✓	✓	✓	✓	✓	✓
sohu.com	❑	❑	✓	✓	❑	✓	✓	✓	✓	✓	✓	✓	✓
aliyun.com	❍	❍	✓	✓	❍	✓	✓	✓	✓	✓	✓	✓	✓
tom.com	❑	❑	❍	❍	❑	❍	☆	☆	☆	☆	☆	☆	☆
139.com	-	-	-	-	-	-	-	-	-	-	-	-	-
gmail.com	❍	❍	❍	❍	❍	❍	✓	✓	✓	✓	✓	✓	✓
yahoo.com	❍	❍	❍	❍	❑	❍	✓	✓	✓	✓	✓	✓	✓
zoho.com	❍	❍	❍	❍	❍	❍	✓	✓	✓	✓	✓	✓	✓
aol.com	❍	❍	❍	❍	❑	❍	✓	✓	✓	✓	✓	✓	✓
yandex.ru	❑	☆	❍	❍	❑	❍	✓	✓	✓	✓	✓	✓	✓
mail.ru	☆	☆	☆	❍	☆	☆	✓	☆	☆	☆	✓	☆	✓
outlook.com	✗	✗	✗	✓	✗	✓	✓	✗	✗	✗	✓	✗	✗
icloud.com	☆	☆	☆	☆	☆	☆	✓	✓	☆	☆	☆	✓	☆
runbox.com	☆	☆	☆	❍	☆	❍	✓	☆	☆	☆	✓	☆	✓
fastmail.com	❑	❑	✓	✓	❑	✓	✓	✓	✓	☆	☆	✓	☆
cock.li	❑	❑	☆	☆	❑	☆	✓	✓	✓	☆	☆	✓	☆
daum.net	❍	❍	✓	✓	✂	✓	✓	✓	✓	✓	✓	✓	✓
rambler.ru	❑	❑	☆	☆	❑	☆	✓	✓	✓	☆	☆	✓	☆
naver.com	❍	❍	✓	✓	❍	✓	✓	✓	✓	✓	✓	✓	✓
freemail.hu	-	-	-	-	-	-	-	-	-	-	-	-	-

✓: Vulnerable. Email services send out the original email data with the payload we inserted without filtering.
✓: Vulnerable. Fastmail will send multiple emails after processing.
✗: Rejected. ❍: Dot Stuffing. ❑: Sender Checking. ☆: String Filtering. ✂: Truncated. -: SMTP Service Not Available.

Table 3: Sending MTAs: Testing Results of 22 Public Email Services. The attack payloads A_1–A_{13} are presented in Table 1.

on the sending side[2]. The vulnerable MTAs failed to filter smuggled payloads, allowing malicious emails to be sent without alteration. Among the tested MTAs, 13 email services employed *dot stuffing*, which adds one extra period at the beginning of lines that start with a period (i.e. altering "`\n.\n`" to "`\n..\n`"). These services are denoted by a circle in Table 3. Although dot stuffing was initially designed to prevent confusion with legitimate lines in the message body, it can also serve as a potential mitigation technique against SMTP smuggling. We will discuss dot stuffing as a mitigation technique in Section 6. Certain services like Mail.ru and iCloud implemented string filtering (denoted by a star), adding extra space characters before or after bare "`\n`" and "`\r`". Nine email services interpreted our email as two emails *during the sending phase*, which triggered a sender checking (denoted by a square) on the second email to prevent spoofing.

While many services perform string filtering and dot stuffing on outgoing emails, these methods are not foolproof, leading to vulnerabilities in some instances. We found these methods often fail to filter the `\x00` character, including Gmail and Yahoo (`\x00` is part of the payload of A_7–A_{13}). It is mainly because they treat `\x00` as a string terminator (common for email servers programmed by C). Notably, Fastmail exhibited more severe issues, where emails were split into multiple parts *before sending* (denoted by an orange checkmark). This is beyond typical smuggling and would allow attackers to set different sender and recipient addresses for each email. This means Fastmail can be exploited as an email relay for spam or email bombing attacks.

Collectively, 19 out of the 22 public email services were either vulnerable on the sending side or on the receiving side (or both). Considering that a successful attack requires both the sending and receiving MTAs to be vulnerable, we analyzed the combinations of sending and receiving services to assess the attack potential. By combining Tables 2 and 3, we identified 36 exploitable pairs out of the 440 (20×22) possible sending-receiving combinations. Furthermore, considering these public email services provide email services for thousands of domains, the vulnerabilities on the sending side not only affect the providers themselves but can also extend to the domains they serve.

3.3 Impact of Shared SPF Infrastructure

SMTP smuggling allows attackers to forge emails from different accounts within the same domain. The threat becomes even more severe when combined with shared SPF infrastructure. A previous study [54] has highlighted the prevalence of shared SPF infrastructure in today's email ecosystem. Organizations often share SPF records to simplify email configuration and maintenance, allowing them to delegate email handling to a third-party provider without needing to man-

[2] 139.com and freemail.hu were excluded from sending MTA tests since they did not have available SMTP services.

Email Service	Shared*	Email Service	Shared*
outlook.com	81,718	runbox.com	35
gmail.com	63,225	aol.com	18
yandex.ru	4,838	sina.com	7
zoho.com	4,074	163.com	6
qq.com	1,713	naver.com	5
mail.ru	999	sohu.com	1
fastmail.com	768	aliyun.com	1
daum.net	87	tom.com	1
yahoo.com	62	cock.li	1
icloud.com	47	rambler.ru	1

* : Share column indicates the number of Tranco Top 1 Million domains that are within the shared SPF infrastructure of these public email services.

Table 4: Shared SPF Infrastructure for Public Email Services.

Outlook		Gmail	
Rank	Domain	Rank	Domain
3	microsoft.com	1	google.com
15	instagram.com	8	youtube.com
19	live.com	13	twitter.com
28	bing.com	14	cloudflare.com
31	microsoftonline.com	36	fastly.net
44	github.com	37	netflix.com
46	sharepoint.com	38	googlesyndication.com
53	skype.com	41	googleusercontent.com
56	digicert.com	43	youtu.be
61	msn.com	48	pinterest.com

Table 5: Top 10 Domains within the Shared SPF Infrastructure of Outlook and Gmail.

age SPF settings themselves. This practice, while convenient, significantly increases the risk of email spoofing.

In a shared SPF infrastructure, multiple domains may include the same email provider's SPF records in their own SPF records, sharing the same set of authorized IP addresses. Consequently, any IP address within this shared set is permitted to send emails on behalf of all the domains using that infrastructure. For example, domains like *youtube.com*, *twitter.com*, and *zoom.us* all include Google's SPF record, which allows any IP address within Google's authorized range to send emails on behalf of these domains.

The shared SPF infrastructure significantly amplifies the risks associated with SMTP smuggling, allowing attackers to compromise all domains within the shared SPF infrastructure, thereby undermining the security guarantees that SPF is supposed to provide. Here, we describe a threat model for this threat. We assume: (1) The attacker has an email account with a domain (e.g., `a.com`). (2) The attacker recognizes this email service is vulnerable to SMTP smuggling on the sending side. (3) Both `a.com` and another trusted domain (e.g., `b.com`) use the same shared SPF infrastructure. The attacker can exploit this situation by performing an SMTP smuggling attack from one domain (e.g., `attacker@a.com`) within the shared infrastructure, making the malicious emails appear as if they originate from another trusted domain (e.g., `admin@b.com`). For large email providers, this is particularly dangerous because it allows attackers to impersonate numerous highly trusted domains, increasing the likelihood of successful phishing.

Spoofable Domains. To assess the impact of this threat model, we analyzed the SPF infrastructure used by these public email services. First, we recursively measured the SPF records of 20 public email services (139.com and freemail.hu were excluded due to unavailable SMTP services). We collected the public email services' SPF records as well as all other domains indirectly involved through the "include" and "redirect" mechanisms. These domains collectively form the shared SPF infrastructure of these public email services. Second, we expanded our measurement to the SPF records of the Tranco Top 1 Million domains [25]. By parsing the "include" and "redirect" mechanisms within these SPF records, we mapped out the SPF dependencies between them. This allowed us to identify which domains rely on the shared SPF infrastructure of public email services. Combining these analyses, we counted the number of domains encompassed within the shared SPF infrastructure of public email services.

The results are presented in Table 4. Outlook and Gmail are among the top with the highest number of shared domains (81,718 and 63,225, respectively). These numbers should be viewed as a *lower bound* on the actual situation since our analysis cannot count for certain SPF-sharing scenarios. For example, some email services, such as aliyun.com, use *different domain names* for their business email services, separating them from their free email service's SPF records. However, their clients still share the same SPF infrastructure. This means a free email service account can still spoof their business email domains.

Case Study. Different email services have inconsistent strategies for handling SMTP smuggling payloads, making it possible to launch email spoofing attacks. For instance, Gmail treated the `\r\n\x00\r\n` payload as normal text when sending emails. However, Daum.net recognized `\r\n\x00\r\n` as the end indicator of `DATA` command. Thus, attackers can utilize Gmail as the sending MTA and choose any email account of Daum.net as the target victim.

Note that all domains that rely on the same shared SPF infrastructure are susceptible to exploitation. This means attackers, with a free Gmail account, can spoof any domain that uses Google's mail service—at least 63,225 domains within the Tranco Top 1 Million domains. To demonstrate this, we performed a proof-of-concept experiment by sending emails to *our own Daum.net account*. We can successfully spoof the identity of *admin@youtube.com* and send an email to *victim@daum.net*, as shown in Figure 1. The spoofed email passed both SPF and DMARC verification without triggering any alerts. There are other attack cases shown in Figure 8 in the Appendix where we

spoofed admin@microsoft.com, admin@openai.com, and admin@github.com. We also listed the top 10 domains within the shared SPF infrastructure of two leading email providers, Gmail and Outlook in Table 5, which included many high-profile targets such as Microsoft, Google, Twitter, Netflix, Pinterest, Skype, and Instagram.

4 Private Email Service Experiments

In this section, we extend our experiments to *private email services* and present novel measurement methodologies for ethical and non-intrusive tests. Previous research on email spoofing attacks [7,18,44] has been primarily focused on public email services because they allow researchers to register their own accounts for controlled testing without affecting other users. However, this method cannot be used to obtain insights into private email systems. In this section, we combine user studies, DKIM side channels, and a non-intrusive test method to perform our experiment. Our test will be focused on *receiving MTAs* since the receiving end holds a higher responsibility to interpret the end-of-data indicator correctly.

4.1 User Study

We start with a user study where we recruit users who own an account of a private email service to help with the testing. There are two purposes. First, we seek to understand whether these private email services are vulnerable to SMTP smuggling. Second, we use this opportunity to collect "ground truth" data to explore non-intrusive and scalable methods to test a broader range of private emails (for Section 4.2).

We recruited participants from 48 universities who owned institutional email accounts and provided consent for us to send testing emails. After receiving the testing emails, users then reported the results back, using an online questionnaire.

Ethics. Our user study has been reviewed and approved by the Institutional Review Board (IRB). No personally identifiable information (PII) was collected except for email addresses (only used for testing). Formal consent was obtained from participants before any emails were sent. We explain our full ethical considerations in Section 9.2.

User Study Process and Questionnaire. We hosted the user study online via Qualtrics [39]. Participants began by reading a consent form (which contained detailed information about our study). After providing consent, participants were asked to input their organizational email address. We then sent 13 testing emails to the provided email address, each containing a different variant of the SMTP smuggling payload listed in Table 1. Participants were instructed to wait about 5 minutes while our sending service dispatched all test emails.

Figure 6 (Appendix) shows the participants' view of their inbox during the experiment. If the email service was vulnerable to a specific payload, the corresponding testing email would be split into two, with the second smuggled email titled "*Successful! Please report it to us*." If their email service was not vulnerable to this payload, there would be only one email with the subject "*Normal; Please Ignore*." Participants reported the results back to us by answering a multiple-choice question on the questionnaire. They were asked to select the subject lines (and the email IDs) they saw in the inboxes that indicated attack success. In this way, we can identify which testing email (and attack payload) has reached the receivers.

Figure 3: The Workflow of DKIM Verification Side Channel.

DKIM Verification Side Channel. The above method relies on users to *manually* report the testing results back to researchers. There are two drawbacks: (1) it requires user cooperation and is difficult to scale; (2) user-reported results may contain errors due to potential misunderstanding of the question or misreading the email subjects. For these considerations, we took this user study as an opportunity to experiment with a side channel to determine attack success attacks *without user involvement*.

The idea is to use *DKIM verification* as a side channel to distinguish whether a testing email has been split into two emails due to SMTP smuggling. Figure 3 shows the workflow. ❶ We host an authoritative DNS server for DKIM verification. In this example, we host a subdomain of a.com (*_domainkey.a.com*) and set up wildcard resolution of TXT records for any subdomains. ❷ We add two DKIM signatures to the test email before sending it out. The first DKIM signature covers the entire email, and the second signature covers the smuggled part. As highlighted in Figure 3, we set different *selectors* [3] to differentiate between the two DKIM signatures.

[3] DKIM selector is used to distinguish between multiple keys published in a single domain's DNS records.

These two DKIM signatures are both valid signatures, which help the email pass through email authentication and reach users' inboxes regardless of the server's vulnerability status. ❸ When the email is received, if the target email MTA is vulnerable, it will split the email into two separate emails and query the DKIM records for both DKIM signatures. If the target email MTA is not vulnerable, then it will only query the DKIM record for the first DKIM signature. These queries are logged by our DNS server. ❹ The receiving MTA validates incoming emails and transmits them to the MUA.

As shown in Figure 3, we use the DKIM domain selector to encode the testing email's ID and target MTA domain name. For example, `b_com_1_second` represents test email #1 sent to the target MTA `b.com`. By analyzing the logs of our DNS server, we can distinguish whether the target MTA (`b.com`) has requested the second signature, and thereby tell whether it is vulnerable to SMTP smuggling (with a specific payload).

This side channel, if verified reliable, can eliminate the need for receiver cooperation in SMTP smuggling tests. A precondition is the target email services would perform DKIM verification for received emails. Recent measurements show that DKIM is widely adopted [55] and this precondition is met by most of the major email services.

User Study Results. We recruited participants from 48 universities in five countries, with 40 located in the U.S. and the rest in the U.K., Germany, China, and Australia. Due to the space limit, we present the detailed testing results in Table 10 in the Appendix. Out of the 48 private email services tested, we identified SMTP smuggling vulnerabilities in 23 services. Most of them are vulnerable to more than one type of attack payload. Among different types of payload, A_1 (`\n.\n`) and A_5 (`\n.\r\n`) are the most impactful. This is consistent with the observation in public email services. Notably, one email service (U_1) was vulnerable to all the payloads tested.

We also cross-compared the user-reported results with the DKIM side channel results (all tested email services supported DKIM verification). We found that the DKIM side channel's result matched the user reports 100%, confirming its effectiveness. This side channel is useful for implementing SMTP smuggling vulnerability tests without the receiver's cooperation, and can automatically generate vulnerability reports. We later used it to implement a self-diagnosis tool for our responsible disclosure. We recognized the potential bias in our user study due to the focus on university email services and tested more private email systems in the next section.

4.2 Large-scale Non-intrusive Test

Although the DKIM side channel can automatically detect vulnerable SMTP servers without receiver cooperation, it still requires delivering testing emails into the receivers' inboxes. If most email services being tested are not vulnerable, sending emails to their inboxes would be undesirable. To perform large-scale tests, an even less intrusive method is needed.

(a) If NOT vulnerable: it automatically terminates the SMTP session earlier without sending any email.

(b) If vulnerable: it automatically sends a vulnerability report to their admin account.

Figure 4: Examples of the Non-intrusive Test.

Here, our idea is to combine *vulnerability test* with *vulnerability notification* in a single SMTP session: it delivers an email (in the form of a vulnerability report) to the email administrator's inbox *only if their server is vulnerable*; otherwise, the receiver will not receive any emails.

Testing Method. Figure 4 demonstrates this idea. We design an email that includes a carefully crafted payload. In this example, the special payload is highlighted in red: `\n.\n` is the attack payload to trigger SMTP smuggling in the email data, and then we append `MAIL FROM`, `RCPT TO`, `DATA` commands after this payload. Note that the red payload ends with "`\n`" instead of the actual end-of-data indicator "`\r\n.\r\n`", which is on purpose (explained later). The main body of the email (i.e., the blue text) is a vulnerability report message where we explain what SMTP smuggling vulnerability is, and how it can be used for spoofing. We also explain that we have found their email service vulnerable to SMTP Smuggling and invite them to perform further self-diagnosis.

Given a target private email service, we send this email to their administrator account (or other email accounts reserved for security reporting). As shown in Figure 4(a), when the target email server is not vulnerable, it will not interpret "`\n.\n`" as the end-of-data indicator of the SMTP `DATA` command. Since the receiving MTA never sees an actual end-of-data indicator "`\r\n.\r\n`", it will assume the email data has not been fully transmitted and will continue waiting for the end indicator. If our sending service does not receive any reply within ten seconds, we will proactively end the SMTP session. In this case, the email receiver will not receive any email. In other words, if the target email server has no identified vulnerability, no email will be sent, and we can completely avoid disrupting the administrator account.

If the target email service is vulnerable, Figure 4(b) shows the subsequent interactions. The email server will interpret "`\n.\n`" as the end-of-data indicator, treat it as the end of the first email, and start to process the commands for the second email (i.e., the red commands inserted after `\n.\n`). If the server correctly responds to the second `DATA` command with a 354 status code, it indicates this server is vulnerable. At this point, we send `QUIT` to terminate the SMTP session to avoid sending the second email. However, the first email (which contains the vulnerability report) has already been sent to the administrator account. In other words, if the email server is vulnerable, a vulnerability report is automatically sent to the administrator's account to notify them.

Attack Payload Selection. To balance the need for thorough testing and ethical considerations (e.g., to avoid spamming the receiver accounts), we limited the large-scale testing to using a single attack payload. We selected A_1 (`\n.\n`) because it is the most effective in our tests on public and private email services thus far. This allows us to identify more vulnerable private email services without spamming email administrators (who receive a single vulnerability report email only if their server is vulnerable). Recall that in our vulnerability report, we included a link to our online service where email administrators can conduct more detailed self-tests to determine whether their service is affected by other payloads.

Ground-truth Validation. Compared with the method in Section 4.1 (user study, DKIM side channel), this non-intrusive test further avoids sending smuggled emails to the inboxes. We first want to verify this non-intrusive method can correctly detect vulnerable services. To do so, we run the non-intrusive test on the email services involved in our user study. Then, we use the user study results as the "ground truth" to cross-compare with the results of non-intrusive tests. We find the two sets of results are perfectly consistent, indicating the reliability of this method.

Ethics. The above discussion covers most of our ethical considerations. Note that we included our contact information including our email and our website URL in every vulnerability report such that email admins can ask follow-up questions. Extended discussion of ethics is presented in Section 9.2.

Results. We conducted a non-intrusive test for the Tranco top 10,000 domains [25]. To avoid any disruption to regular users, we selected five email addresses (security@, abuse@, postmaster@, support@, and info@), which are commonly associated with administrative functions according to RFC 2142 [8] and Stock et al. [46].

To begin, we first checked each domain for the presence of an MX record. Among the top 10,000 domains, 2,503 domains did not have an MX record, indicating that they did not accept emails. For the remaining 7,497 domains with valid MX records, we attempted to send emails to each of the five addresses listed above. By analyzing the SMTP session log, we identified which administrative accounts are reachable.

Account	Security	Abuse	Postmaster	Support	Info	Total
Domains	3,583	5,046	5,189	4,056	3,795	6,917

Table 6: Common Administrative Email Account Names.

Rank	Domain	Infrastructure
5	amazonaws.com	Amazon
10	akamai.net	Proofpoint
12	akamaiedge.net	Proofpoint
14	cloudflare.com	Postfix
15	instagram.com	Proofpoint
25	akadns.net	Proofpoint
27	amazon.com	Amazon
30	wikipedia.org	Postfix
39	wordpress.org	Postfix
67	yandex.net	Yandex

Table 7: Top 10 Domains Influenced by SMTP Smuggling Identified by Non-Intrusive Tests.

As shown in Table 6, 6,917 domains had at least one address configured and accepting emails.

In total, out of 6,917 domains tested, 5,280 are deemed *not vulnerable* since they did not react to the inserted payload. Their SMTP sessions were terminated earlier to avoid sending emails to administrators. For the remaining 1,637 domains, they all recognized `\n.\n` as the end-of-data indicator. However, 60 of them did not react to our following commands normally (i.e., the second `DATA` command for smuggling). There are two common reasons. First, some domains rejected our second emails due to the greylisting mechanism (for spam control). It is difficult to determine whether they are vulnerable to SMTP smuggling. Second, some domains disabled the pipeline mechanisms, which prevented sending multiple emails in one SMTP session. To this end, we excluded these 60 domains and determined that the remaining 1,577 domains were vulnerable to SMTP smuggling. These domains' administrators also received our vulnerability report emails.

Among the 1,577 vulnerable domains, 20 domains are ranked within the top 100 domains on the Tranco list [25] (198 are ranked within the top 1,000 domains). This finding underscores the widespread risk across a range of high-profile websites and organizations. Table 7 lists the top 10 domains that we identified as being affected by SMTP smuggling.

5 Exploring the Sources of the Vulnerability

Given the widespread vulnerability, it is crucial to investigate where these vulnerabilities originate and why they remain unaddressed. This section seeks to explore the underlying causes of SMTP smuggling vulnerabilities across different email infrastructures.

SMTP Banner Analysis. We first conducted an SMTP banner analysis to identify the underlying infrastructures used by

Test	Type	Name	Version	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}	A_{13}
Receiving MTA Test	Software	Postfix	3.9.0	✂	✂	✗	✗	✂	✗	✗	✗	✗	✗	✗	✗	✗
		Sendmail	8.18.1	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
		Exim	4.97.1	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
		Haraka	3.0.3	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
		Axigen	10.5.25	✓*	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	Gateway	TrendMicro	-	✓	✓	✗	✗	✓	✗	☆	☆	☆	☆	☆	☆	☆
		Sophos	-	☆	☆	☆	☆	☆	☆	☆	☆	☆	☆	☆	☆	☆
Sending MTA Test	Software	Postfix	3.9.0	✂	✂	☆	☆	✂	☆	✓	✓	✓	☆	☆	✓	☆
		Sendmail	8.18.1	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗
		Exim	4.97.1	☆	☆	❍	❍	☆	❍	✓	☆	☆	☆	✓	☆	✓
		Haraka	3.0.3	✓*	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
		Axigen	10.5.25	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Gateway	TrendMicro	-	✂	☆	❍	❍	✂	❍	✂	✂	✂	✂	✂	✂	✂
		Sophos	-	✂	☆	❍	❍	✂	❍	✂	✂	✂	✂	✂	✂	✂

✓: Vulnerable. For sending MTA tests, it means email services send out the original email data with the payload we inserted without filtering.
✓: Vulnerable. Email servers will send multiple emails after processing. ✗: Rejected. ✗: Not Vulnerable. ❍: Dot Stuffing. ☆: String Filtering. ✂: Truncated.
* : Axigen will only show the smuggled email and drop the first email. Haraka will only send the smuggled email out.

Table 8: Testing Results of Open-source Email Software and Email Gateways.

Infrastructure	Type	# Vul Domain
Proofpoint	Gateway	512
Postfix	Software	316
Cisco	Gateway	77
Yandex	Service	70
Sendmail	Software	36
Exim	Software	32
Amazon	Service	29
Bytedance	Service	10
Netease	Service	8
Forcepoint	Gateway	7

Table 9: The Top 10 Vulnerable Email Infrastructures.

vulnerable services. The SMTP banner often contains information about the email server software (and its version) and/or the gateways used by a given domain. We collected SMTP banners from all the vulnerable email services identified in our non-intrusive tests, and extracted the second-level domains and keywords. By manually aggregating the results based on prior knowledge, (e.g., *gpphosted.com* and *pphosted.com* both belong to Proofpoint), we identified a distinct fingerprint for each infrastructure. This allowed us to pinpoint shared infrastructures susceptible to SMTP smuggling. Subsequently, we matched the vulnerable email services to our identified fingerprints, allowing us to assess the extent to which these infrastructures were being utilized by vulnerable domains.

The SMTP banner analysis revealed three main types of email infrastructures: *Email Software*, *Email Security Gateways*, and *Email Services*. Table 9 highlights the top 10 infrastructures associated with the highest number of vulnerable domains. Although email gateways are typically deployed to protect email systems, we found that their widespread usage (e.g., Proofpoint) has inadvertently contributed to the spread of SMTP smuggling vulnerabilities. Open-source email software such as Postfix, which was linked to 316 vulnerable domains, also played a critical role. Popular email services such as Yandex also contributed to the issue. These findings demonstrate that the most commonly used infrastructures not only fail to mitigate SMTP smuggling vulnerabilities but amplify their impact. To gain deeper insights into the vulnerabilities in open-source email software and email gateways, we further tested instances that were accessible to us.

Open-source Email Software. Open-source email software is a cornerstone of many organizations' email services. We selected five widely used open-source email software (based on [56]) for our testing: Postfix [36], Exim [12], Sendmail [43], Haraka [15], and Axigen [3]. Although some of these software developers have claimed that they addressed SMTP smuggling, we tested their *latest versions* to verify whether the issues have been fully resolved.

As shown in Table 8, on the receiving side, Postfix, Sendmail, and Exim mitigated SMTP smuggling vulnerabilities. However, Haraka and Axigen were still vulnerable to certain payloads. On the sending side, all tested software packages were still susceptible. In particular, Haraka and Axigen lacked filtering for all 13 payloads we tested.

Moreover, considering that many email services in use today are based on *historical versions* of these open-source email servers, we extended our testing to include older versions of the three most popular email software: Postfix, Sendmail, and Exim. The results, shown in Table 11 (Appendix), indicated that these vulnerabilities were widespread in earlier versions (e.g., Postfix before version 3.9.0, Sendmail before version 8.18.1, and Exim before version 4.97.1). Given that email services may not update their software frequently, the threat can remain pervasive.

Additionally, we analyzed the source code of the vulnerable software and discovered that these email programs intentionally support "\n" as a line terminator. According to an explanation by the Postfix project maintainer [51], this issue was originally introduced by Sendmail decades ago, and both Postfix and Exim chose to support it in order to maintain compatibility with Sendmail.

Email Security Gateway. Email services used email security gateways to protect against email-borne threats [40]. These gateways filter and inspect incoming and outgoing emails, acting as the primary barrier against malicious activities such as phishing, spam, and malware attacks. Here, we examine how email security gateways handle SMTP smuggling attacks. We selected 15 email security gateways referenced in prior research [40], and attempted to obtain test accounts by registering on their official websites and sending request emails. Only two, TrendMicro [31] and Sophos [45] provided us with a test account. Most of these gateways primarily cater to business clients, and due to budget constraints, we were unable to test them directly.

Both TrendMicro and Sophos offer cloud-based filtering services. To evaluate these gateways, we set up our own email services, pointed our domain's MX record to the email gateways, and configured them to forward emails to our service. The results of these tests are presented in Table 8. Our findings revealed that TrendMicro is vulnerable to A_1 and A_5 attacks on the receiving side but is not vulnerable on the sending side. Sophos demonstrated no vulnerability.

Additionally, based on our previous SMTP banner analysis, we inferred gateways from Proofpoint, Cisco, and Forcepoint were likely susceptible. The rationale is their association with vulnerable domains is strong. For example, for Proofpoint, there were 561 domains in our testing set using Proofpoint, and 512 (91.1%) were found vulnerable[4]. The vulnerable rates for Cisco and Forcepoint are 81.1% and 100% respectively.

It is important to emphasize that our findings are restricted to email *spoofing* vulnerabilities (instead of the *overall security* of these vendors). Other mechanisms implemented by email security gateways, such as spam filters, URL/attachment scanning, and brand impersonation detection, may provide mitigation against real-world email threats. That being said, we argue that the ability to systematically bypass spoofing detection using SMTP smuggling is still a concern, which allows attackers to impersonate high-value targets.

6 Discussion

6.1 Lessons Learned

Distinguishing data from commands. One of the root causes of SMTP smuggling is that, as a plain-text protocol, SMTP does not distinguish data from commands. This allows attackers to inject malicious commands into SMTP data for exploitation. Distinguishing data from commands is a fundamental concept in computer security, particularly in the context of preventing attacks such as code injection. SMTP smuggling is one recent instance of this attack category.

The inconsistent processing logic between the sending and receiving MTAs causes SMTP smuggling vulnerabilities. As discussed in Section 3.2, SMTP smuggling attacks hinge on the differing interpretations of the `DATA` command's end indicator between sending and receiving MTAs. Even if the sender MTA behaves in full compliance with RFC standards (i.e., only recognizing "`\r\n.\r\n`" as the end-of-data indicator), the discrepancy in how the receiving MTA interprets the DATA command can be manipulated by attackers to carry out email spoofing. This demonstrates that even compliant systems (i.e., senders) can be vulnerable when their interaction with other systems introduces unforeseen security gaps.

The shared email infrastructures have magnified SMTP smuggling vulnerabilities. The centralization of email services is a double-edged sword. On the positive side, it can reduce deployment costs and simplify the management of email services. Large email service providers often invest more resources into security, which can enhance the overall security of their customers. However, this study highlights the negative side of email service centralization, particularly concerning SMTP smuggling.

The centralization of email services magnifies the attack's impact in two ways. From the sending side, shared SPF records increase the vulnerability of IP address-based authentication chains. A security issue in one email service provider's policy can potentially affect thousands of domains. From the receiving side, the wide use of a small set of popular gateways (e.g., Proofpoint [38]) and open-source email software (e.g., Postfix [36]) amplifies the impact of smuggling vulnerabilities across the email ecosystem.

6.2 Mitigation

Receiving Side. We start with receiving-side mitigation.

Standard End-of-Data Indicator. For receiving MTAs, a fundamental way to eliminate SMTP smuggling vulnerabilities is to use the standard end-of-data indicator (`\r\n.\r\n`) for `DATA` command. However, this fundamental solution may not be easy to fully realize in practice. The key challenge is to update legacy systems to be in compliance. The hazards of using other end indicators have been discussed explicitly in Section 4.1.1.4 of RFC 5321 [23]:

"The custom of accepting lines ending only in <LF>, as a concession to non-conforming behavior on the part of some UNIX systems, has proven to cause more interoperability problems than it solves."

Disabling Pipeline Mechanism. A disabled pipeline mecha-

[4] Not all services that use Proofpoint are vulnerable since it also depends on configurations

nism can serve as a short-term mitigation against SMTP smuggling. With SMTP smuggling, attackers exploit the server's ability to process multiple email commands in a pipeline to send a second email. By disabling pipelining, the server processes each command sequentially, enforcing a stricter adherence to the processing flow, thus reducing the risk of command injection. The drawback is that disabling SMTP command pipelining can hurt the performance (throughput) of email communications. Note that Postfix provides this way as a short-term workaround [51].

Sending Side. For sending MTAs, there are several potential mitigation approaches.

Data Encoding. SMTP smuggling represents a form of command injection attack where attackers exploit the protocol's inability to distinguish between data and commands. To separate data from commands, a widely used method is data encoding, which involves transforming the SMTP body into a different format using algorithms like Base64 [21]. By encoding email content *by default*, the email data is converted into a format that cannot be mistaken for SMTP commands. Implementing this solution only requires setting the `Content-Transfer-Encoding` header to `base64` and encoding the email body with the corresponding algorithm. This is *fully compliant* with relevant RFCs and can ensure compatibility with existing systems. Notably, some implementations, such as Python's "smtplib", have already adopted this encoding by default. We recommend this method as a primary mitigation strategy on the sending side.

Dot Stuffing and Suspicious String Filtering. In SMTP, a single dot (".") on a line signals the end of an email's content but can also appear within the email body to cause confusion. Dot stuffing, as outlined in RFC 5321 [23], addresses this by adding an extra dot at the start of any line that begins with a single dot. While this may reduce the risk of SMTP smuggling (for certain attack payloads), this is *not* a fundamental fix. Our experiments in Section 3.2 show that not all email services consistently enforce dot stuffing, and attackers can bypass it using the null character (`\x00`), which disrupts line recognition. Dot stuffing can be further enhanced by "suspicious string filtering." For example, email services can filter out specific characters (e.g., bare `\r`, bare `\n`, `\x00`). These mechanisms are useful short-term fixes (to block the attacks described in our paper) but may not be a fundamental solution compared with the data encoding method above.

Online Detection Service. We developed an online detection service based on our experimental infrastructure, allowing email administrators to self-diagnose whether their services are vulnerable to SMTP smuggling attacks. This service sends 13 emails containing the payloads (see Section 3.2) to the specified destination email address. The URL to the service is `https://smuggling.breakspf.cloud`. A screenshot is shown in Figure 7 in the Appendix.

6.3 Limitations

Research Scope. The measurement scope of our study is by no means exhaustive. For public email services, we prioritize popular email services that has a large user base. For private email services, we are limited to university email services where we can recruit participating users and top-ranked email domains using non-intrusive tests. More importantly, the private email experiment is restricted to the *receiving side*. Due to the difficulty of obtaining private email accounts for email sending, we are unable to evaluate SMTP smuggling on their sending side. Additionally, for email software and gateways, our study is limited to open-source software and free-to-register gateway services.

Biases in the User Study. For the user study, we acknowledge that user self-reported data may introduce biases or inaccuracies. To alleviate the concerns, we have cross-validated the user self-reported data with the DKIM side-channel results. We have confirmed that all users received emails in their organization inbox, and the reported results were consistent with the DKIM logs. Since user experiments are not suitable for large-scale testing, we further propose a non-intrusive testing method (using the user study results as validation).

Other Security Protection Mechanisms. It is important to acknowledge that security vendors often deploy additional mechanisms, such as spam filters, malware scanning, URL scanning, and brand impersonation detection, to identify and mitigate email threats. While our findings have highlighted the *spoofing* vulnerabilities in specific gateway implementations and email services, these additional security mechanisms may serve as a complementary layer of defense, reducing the practical impact of these vulnerabilities in real-world scenarios. Future work should incorporate a deeper evaluation of these mechanisms to better understand their efficacy in mitigating the risks associated with SMTP smuggling.

6.4 Responsible Disclosure

We have sent vulnerability reports to all vulnerable domains involved in the *non-intrusive test*. We received 1,713 responses, most of which were auto-replies. By manually analyzing these responses, we identified *manual responses* from 117 domains. 20 domain administrators have used our self-diagnosis tools to perform further tests on their email services. We also checked whether there were complaint emails from domain administrators (via keyword search and manual inspection), and did not find such messages. Certain email administrators thanked our efforts and also shared their own experiences with this vulnerability. For example, one German organization administrator mentioned that they had noticed this vulnerability and tried to fix the problem with the help of their vendor. However, our test showed their email services were still vulnerable to certain variants, and they were fixing

these variants after our notification. Another email administrator mentioned that they were considering migrating their email system to Gmail after receiving our report (so that they do not need to worry about the fixes). While this may mitigate the issue on the receiving side, our results show that Gmail has not yet fully mitigated the sending side problem due to the shared SPF infrastructure (we further informed them so).

In addition, we have reached out to *all other parties* affected by this vulnerability based on our measurement results. Our disclosure targets include public email services, university email services, email software developers, and email gateway providers. For public email services, Outlook, Gmail, Yandex, Kakao, Fastmail, and Zoho acknowledged our report. Outlook classified the issue as "moderate severity", shared the report with their product team, and is working on a fix. Gmail marked the report as a duplicate of an existing bug already under investigation. Yandex confirmed the vulnerability on the receiving side and is currently evaluating the sending-side problem. Kakao (daum.net, kakao.com) has confirmed our report as a valid security issue and is coordinating with relevant departments to implement a fix. Fastmail indicated that they have implemented some mitigation schemes and are working on additional measures. Zoho acknowledged the issue but did not commit to implementing a fix. Unfortunately, Yahoo, AOL, and Sina did not respond to or act on our report. For university email services, 7 universities have responded to our report. They thanked us for the information and started investigating. However, some universities declined to respond to vulnerability reports from external sources.

For email software developers, Postfix suggested that features such as `cleanup_replace_stray_cr_lf` and `message_strip_characters` might mitigate the identified issue by handling special stray characters like `\x00` in email bodies. The Axigen team reviewed our findings and stated that while Axigen, by default, does not sanitize in-transit messages, it can be configured to do so using the "Body CR-LF correction" setting under its web admin interface. This configuration ensures proper handling of sequences like "`\n.\r\n`". For email gateway providers, we only received a response from Trendmicro, stating that they submitted the information to the relevant technical team for further validation and replication. We will continue awaiting responses from other parties and offering our help to mitigate the problem.

7 Related Work

Email Spoofing Attacks. Email spoofing remains a critical concern in email systems. Researchers found that despite using authentication protocols, email providers may still allow certain spoofing emails to reach inboxes [18]. More recently, researchers discovered new vulnerabilities that allowed spoofing emails to bypass SPF, DKIM, and DMARC [7, 44]. In addition, email forwarding has also been exploited to help with email spoofing [27, 53]. Compared with the initial report of SMTP smuggling [30], our novelty has been clarified at the end of Section 2.3. Another closely related work is BreakSPF [54] which measured the shared SPF infrastructure. Our new contribution is the measurement of SMTP smuggling with respect to existing defenses and the impact of its new variants (especially on private email services).

The Centralization of Email Services. The shift toward cloud email services has led to the centralization of email infrastructure, with many domain administrators relying on large email providers. This trend is highlighted in [28], noting the move from independently hosted mail servers to a model dominated by third-party providers. This was echoed by the authors of BreakSPF [54] regarding the shared SPF infrastructure. Along this trend, researchers also pointed out the wide use of third-party email filtering services [40]. Our work builds on these findings by demonstrating how SMTP smuggling can be amplified by this centralization, affecting numerous domains dependent on the shared infrastructure.

Other Related Works. Several studies have focused on measuring the adoption and use of email authentication protocols [1, 2, 4, 6, 10, 11, 13, 49, 55]. Our research demonstrates a method for bypassing these authentication protocols *without directly violating them*. Additionally, researchers have worked on security challenges related to end-to-end encryption in email systems [20, 32, 34, 42, 47]. Concerns about email transmission encryption were highlighted in recent works [35, 48]. Finally, researchers have investigated human factors, e.g., by analyzing Gmail's sender identity indicator and its influence on user behavior and comprehension [29]. These are related works but are orthogonal to our focus.

8 Conclusion

In this study, we focus on SMTP smuggling, a new vulnerability that allows adversaries to perform email spoofing without being detected. We conduct a comprehensive measurement to understand its current defense and the vulnerable status of public email services, private email services, open-source email software, and email gateways. Notably, we propose a set of novel measurement methodologies to test *private* email systems ethically. Our results collectively show that the current defense is insufficient. In addition, the vulnerability has been amplified through the centralization of email infrastructures (e.g., shared SPF records) which allows adversaries to spoof a broad range of highly reputable domains. We offer suggestions on mitigating the problem and develop self-diagnosis tools to help email administrators.

9 Ethics Considerations

We are mindful of the ethical implications of our experiments and have taken great care to avoid negatively impacting any online services or their users during our experiment. Our

study has been reviewed and approved by our Institutional Review Boards (IRB) before starting. In the following, we provide details of our ethical considerations for each of the major experiments. We believe our experiments' benefits (e.g., discovering vulnerable email services, informing vulnerable parties, and obtaining new insights into defense strategies) outweigh the risk (which is minimal).

9.1 Testing Public Email Services

For public email services, the authors registered decided experimental accounts for the tests. This made sure all testing emails were sent to or from these accounts without affecting other users of the target services.

To minimize any potential disruption to the target email servers, both on the sending and receiving sides, we limit our testing to only sending 13 emails (one for each type of payload), at intervals of one email every 30 seconds. This deliberate pacing ensures that our experiments do not interfere with the normal operations of the target services. Finally, all the email messages used for the test do not contain any harmful content (such as malicious URLs or phishing content).

9.2 Testing Private Email Services

For the user study, we explicitly obtained participants' *informed consent* before sending emails to them. We also controlled the total number of emails sent and the sending rate, similar to the public email service tests. The user study was reviewed and approved by our Institutional Review Boards (IRB). We recruited participants from 48 universities who have been validated to own the email account being tested against. We did not collect or store any personally identifiable information (PII) other than participants' email addresses which were only used for the testing. After the testing, we did not store the users' full email addresses (and only kept the domain name part of the address). The testing emails do not contain any harmful content (such as malicious URLs or phishing content). Participants can withdraw their data at any time after completing.

For the non-intrusive test, the experiment method is designed with measurement ethics as the key priority. As described in Section 4.2, if an email service was not vulnerable, we only established an SMTP connection once and broke the session before sending any unsolicited email. For an email service that was detected vulnerable, the testing email will be sent to the *administrator account* (e.g., postmaster) as a vulnerability report to notify them about the vulnerability. This is also part of our responsible disclosure effort. Some domain administrators have thanked us for the vulnerability reports.

9.3 Testing Gateways and Email Software

In Section 5, we used *publicly available* information including DNS records and SMTP banners to identify the use of email software and email gateways of different email services. Our testing method involves setting up *our own email servers* to install the open-source email software and setting up the gateway for testing. This process *does not involve any users*, and thus incurs no risk. The tests were conducted within a controlled environment where the authors/researchers controlled both the sending and receiving sides.

9.4 Spoofing "Admin" Accounts

During the actual experiments (Sections 3–5), we only spoofed accounts and domains under our own control (not "admin" accounts). In addition, all spoofing emails were sent to researcher-controlled accounts. Only for a few case studies in Figure 1 (Introduction) and Figure 8 (Appendix), we performed spoofing tests on "admin" addresses for proof-of-concept to show the realism of the attack. In these case studies, we chose the admin account as the spoofing target because, compared with other email addresses, admin accounts are more difficult for attackers to compromise, making it clear that the email is forged. This choice is also consistent with practices in previous related studies [7, 44]. Although the actual impact is likely minimal, we recommend using non-existent or researcher-owned addresses as the spoofing target for large-scale cross-domain tests. This is is to avoid affecting the sender's reputation of well-known services.

10 Open Science

We are fully committed to complying with the open science policy. For our paper, we make our research artifacts (measurement code[5], datasets[6], and the self-diagnostic tool) available for sharing with other researchers and security industrial practitioners.

Acknowledgments

We thank the anonymous reviewers and our shepherd for constructive feedback to improve this paper. We are grateful for the support from all volunteers who participated in our experiment. We thank the email administrators who responded and fixed the vulnerabilities we reported. This work was in part supported by the National Natural Science Foundation of China (62272265), the Taishan Scholars Program, and the NSF grant (2030521). Any opinions, findings, conclusions, or recommendations expressed in this paper do not necessarily reflect the views of sponsors.

[5] https://zenodo.org/records/14738722
[6] https://zenodo.org/records/14738853

References

[1] Md. Ishtiaq Ashiq, Weitong Li, Tobias Fiebig, and Taejoong Chung. You've got report: Measurement and security implications of DMARC reporting. In *Proc. of USENIX Security*, 2023.

[2] Md. Ishtiaq Ashiq, Weitong Li, Tobias Fiebig, and Taejoong Chung. SPF beyond the standard: Management and operational challenges in practice and practical recommendations. In *Proc. of USENIX Security*, 2024.

[3] Axigen. Axigen: Mail server software. https://www.axigen.com/.

[4] Nathaniel Bennett, Rebekah Sowards, and Casey T. Deccio. Spfail: discovering, measuring, and remediating vulnerabilities in email sender validation. In *Proc. of IMC*, 2022.

[5] Ivan Blagojević. How many email users are there? https://99firms.com/blog/how-many-email-users-are-there/.

[6] Birk Blechschmidt and Ben Stock. Extended hell(o): A comprehensive large-scale study on email confidentiality and integrity mechanisms in the wild. In *Proc. of USENIX Security*, 2023.

[7] Jianjun Chen, Vern Paxson, and Jian Jiang. Composition kills: A case study of email sender authentication. In *Proc. of USENIX Security*, 2020.

[8] Dave Crocker. Mailbox names for common services, roles and functions. *RFC*, 2142:1–6, 1997.

[9] Dave Crocker, Tony Hansen, and Murray S. Kucherawy. Domainkeys identified mail (DKIM) signatures. *RFC*, 6376:1–76, 2011.

[10] Casey Deccio, Tarun Yadav, Nathaniel Bennett, Alden Hilton, Michael Howe, Tanner Norton, Jacob Rohde, Eunice Tan, and Bradley Taylor. Measuring email sender validation in the wild. In *Proc. of CoNEXT*, 2021.

[11] Zakir Durumeric, David Adrian, Ariana Mirian, James Kasten, Elie Bursztein, Nicolas Lidzborski, Kurt Thomas, Vijay Eranti, Michael Bailey, and J. Alex Halderman. Neither snow nor rain nor MITM...: an empirical analysis of email delivery security. In *Proc. of IMC*, 2015.

[12] Exim. Exim internet mailer. https://www.exim.org/.

[13] Ian D. Foster, Jon Larson, Max Masich, Alex C. Snoeren, Stefan Savage, and Kirill Levchenko. Security by any other name: On the effectiveness of provider based email security. In *Proc. of CCS*, 2015.

[14] Ned Freed. SMTP Service Extension for Command Pipelining. RFC 2920, September 2000.

[15] Haraka. Haraka smtp email server · haraka. https://haraka.github.io/.

[16] Grant Ho, Asaf Cidon, Lior Gavish, Marco Schweighauser, Vern Paxson, Stefan Savage, Geoffrey M. Voelker, and David A. Wagner. Detecting and characterizing lateral phishing at scale. In Nadia Heninger and Patrick Traynor, editors, *Proc. of USENIX Security*, 2019.

[17] Lee Howard. Reverse DNS in ipv6 for internet service providers. *RFC*, 8501:1–15, 2018.

[18] Hang Hu and Gang Wang. End-to-end measurements of email spoofing attacks. In William Enck and Adrienne Porter Felt, editors, *Proc. of USENIX Security*, 2018.

[19] Federal Bureau Of Investigation. Business e-mail compromise the 12 billion dollar scam. https://www.ic3.gov/Media/Y2018/PSA180712.

[20] Fabian Ising, Damian Poddebniak, Tobias Kappert, Christoph Saatjohann, and Sebastian Schinzel. Content-type: multipart/oracle - tapping into format oracles in email end-to-end encryption. In *Proc. of USENIX Security*, 2023.

[21] Simon Josefsson. The base16, base32, and base64 data encodings. *RFC*, 4648:1–18, 2006.

[22] Scott Kitterman. Sender policy framework (SPF) for authorizing use of domains in email, version 1. *RFC*, 7208:1–64, 2014.

[23] John C. Klensin. Simple mail transfer protocol. *RFC*, 5321:1–95, 2008.

[24] Murray S. Kucherawy and Elizabeth D. Zwicky. Domain-based message authentication, reporting, and conformance (DMARC). *RFC*, 7489:1–73, 2015.

[25] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. Tranco: A research-oriented top sites ranking hardened against manipulation. In *Proc. of NDSS*, 2019.

[26] Linode. Cloud computing on akamai connected cloud. https://www.linode.com/.

[27] Enze Liu, Gautam Akiwate, Mattijs Jonker, Ariana Mirian, Grant Ho, Geoffrey M. Voelker, and Stefan Savage. Forward pass: On the security implications of email forwarding mechanism and policy. In *Proc. of EuroS&P*, 2023.

[28] Enze Liu, Gautam Akiwate, Mattijs Jonker, Ariana Mirian, Stefan Savage, and Geoffrey M. Voelker. Who's got your mail?: characterizing mail service provider usage. In *Proc. of IMC*, 2021.

[29] Enze Liu, Lu Sun, Alex Bellon, Grant Ho, Geoffrey M. Voelker, Stefan Savage, and Imani N. S. Munyaka. Understanding the viability of gmail's origin indicator for identifying the sender. In *Proc. of SOUPS*, 2023.

[30] Timo Longin. Smtp smuggling - spoofing e-mails worldwide. https://sec-consult.com/blog/detail/smtp-smuggling-spoofing-e-mails-worldwide/.

[31] Trend Micro. Trend micro (us) | industry-leading cyber security platform. https://www.trendmicro.com/en_us/business.html.

[32] Jens Müller, Marcus Brinkmann, Damian Poddebniak, Hanno Böck, Sebastian Schinzel, Juraj Somorovsky, and Jörg Schwenk. "johnny, you are fired!" - spoofing openpgp and S/MIME signatures in emails. In *Proc. of USENIX Security*, 2019.

[33] Adam Oest, Penghui Zhang, Brad Wardman, Eric Nunes, Jakub Burgis, Ali Zand, Kurt Thomas, Adam Doupé, and Gail-Joon Ahn. Sunrise to sunset: Analyzing the end-to-end life cycle and effectiveness of phishing attacks at scale. In *Proc. of USENIX Security*, 2020.

[34] Damian Poddebniak, Christian Dresen, Jens Müller, Fabian Ising, Sebastian Schinzel, Simon Friedberger, Juraj Somorovsky, and Jörg Schwenk. Efail: Breaking S/MIME and openpgp email encryption using exfiltration channels. In *Proc. of USENIX Security*, 2018.

[35] Damian Poddebniak, Fabian Ising, Hanno Böck, and Sebastian Schinzel. Why TLS is better without STARTTLS: A security analysis of STARTTLS in the email context. In *Proc. of USENIX Security*, 2021.

[36] Postfix. The postfix home page. https://www.postfix.org/.

[37] Spamhaus Project. Spamhaus blocklist (sbl). https://www.spamhaus.org/blocklists/spamhaus-blocklist/.

[38] Proofpoint. Proofpoint. https://www.proofpoint.com/us.

[39] Qualtrics. Qualtrics platform. https://qualtrics.com.

[40] Sumanth Rao, Enze Liu, Grant Ho, Geoffrey M. Voelker, and Stefan Savage. Unfiltered: Measuring cloud-based email filtering bypasses. In *Proc. of TheWebConf*, 2024.

[41] Pete Resnick. Internet Message Format. RFC 5322, October 2008.

[42] Jörg Schwenk, Marcus Brinkmann, Damian Poddebniak, Jens Müller, Juraj Somorovsky, and Sebastian Schinzel. Mitigation of attacks on email end-to-end encryption. In *Proc. of CCS*, 2020.

[43] Sendmail. Sendmail open source. https://www.proofpoint.com/us/products/email-protection/open-source-email-solution.

[44] Kaiwen Shen, Chuhan Wang, Minglei Guo, Xiaofeng Zheng, Chaoyi Lu, Baojun Liu, Yuxuan Zhao, Shuang Hao, Haixin Duan, Qingfeng Pan, and Min Yang. Weak links in authentication chains: A large-scale analysis of email sender spoofing attacks. In *Proc. of USENIX Security*, 2021.

[45] Sophos. Sophos: Defeat cyberattacks with cybersecurity as a service. https://www.sophos.com/en-us.

[46] Ben Stock, Giancarlo Pellegrino, Frank Li, Michael Backes, and Christian Rossow. Didn't you hear me? - towards more successful web vulnerability notifications. In *Proc. of NDSS*, 2018.

[47] Christian Stransky, Oliver Wiese, Volker Roth, Yasemin Acar, and Sascha Fahl. 27 years and 81 million opportunities later: Investigating the use of email encryption for an entire university. In *Proc. of IEEE SP*, 2022.

[48] Dennis Tatang, Robin Flume, and Thorsten Holz. Extended abstract: A first large-scale analysis on usage of MTA-STS. In *Proc. of DIMVA*, 2021.

[49] Dennis Tatang, Florian Zettl, and Thorsten Holz. The evolution of dns-based email authentication: Measuring adoption and finding flaws. In *Proc. of RAID*, 2021.

[50] Valimail. Research: Crisis of fake email continues to plague industries worldwide. https://www.valimail.com/newsroom/research-crisis-of-fake-email-continues-to-plague-industries-worldwide-2/.

[51] Wietse Venema. Smtp smuggling. https://www.postfix.org/smtp-smuggling.html.

[52] Maria Vergelis, Tatyana Shcherbakova, and Tatyana Sidorina. Spam and phishing in q1 2019. https://securelist.com/spam-and-phishing-in-q1-2019/90795/.

[53] Chenkai Wang and Gang Wang. Revisiting email forwarding security under the authenticated received chain protocol. In *Proc. of TheWebConf*, 2022.

[54] Chuhan Wang, Yasuhiro Kuranaga, Yihang Wang, Mingming Zhang, Linkai Zheng, Xiang Li, Jianjun Chen, Haixin Duan, Yanzhong Lin, and Qingfeng Pan. BreakSPF: How Shared Infrastructures Magnify SPF Vulnerabilities Across the Internet. In *Proc. of NDSS*, 2024.

[55] Chuhan Wang, Kaiwen Shen, Minglei Guo, Yuxuan Zhao, Mingming Zhang, Jianjun Chen, Baojun Liu, Xiaofeng Zheng, Haixin Duan, Yanzhong Lin, and Qingfeng Pan. A large-scale and longitudinal measurement study of DKIM deployment. In *Proc. of USENIX Security*, 2022.

[56] Wikipedia. List of mail server software. https://en.wikipedia.org/wiki/List_of_mail_server_software.

A Email Delivery Consideration

To protect users from the bulk of spam emails, the receiving MTA will typically perform multi-level actions to authenticate the sending MTA and ensure that the email is legitimate, including (1) IP reputation and blocklist check, (2) reverse DNS (rDNS [17]) lookup, (3) domain reputation, (4) email authentication protocol verification. To ensure our testing emails can be delivered to the destination email addresses, we have taken the following steps. First, we ensured that the IP address of our sending MTA was not listed in any block list. For our experiment, we obtained a cloud server from Linode [26]. Initially, the IP address Linode gave us was listed in some block lists (e.g., Spamhaus [37]). We submitted a request to Spamhaus to remove the IP address from their list. Second, we registered a new domain and configured a valid rDNS record that matches the domain name. Third, we configured authentication protocols (SPF, DKIM, DMARC) for our domain and attached DKIM signatures to all emails we sent.

Figure 5: An Example of SMTP Session.

Figure 6: Testing Emails in the Inbox from User Perspectives.

Figure 7: Online Service for Self-Diagnosis of SMTP Smuggling Vulnerabilities.

(a) A spoofing email sent from Outlook to Kakao.com, impersonating *admin@microsoft.com*.

(b) A spoofing email sent from Outlook to Kakao.com, impersonating *admin@openai.com*.

(c) A spoofing email sent from Outlook to Kakao.com, impersonating *admin@github.com*.

Figure 8: Examples of Email Spoofing Attacks.

ID	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}	A_{13}	Vul*	Infrastructure	Country
U_1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-	*US*
U_2	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	*Gmail*	*US*
U_3	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	*Proofpoint*	*US*
U_4	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	*Proofpoint*	*US*
U_5	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✓	-	*US*
U_6	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	*Outlook*	*US*
U_7	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	-	*US*
U_8	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	*Proofpoint*	*US*
U_9	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	*Outlook*	*US*
U_{10}	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	*Outlook*	*US*
U_{11}	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	*Proofpoint*	*US*
U_{12}	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	*Proofpoint*	*US*
U_{13}	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	*Proofpoint*	*US*
U_{14}	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	*Gmail*	*US*
U_{15}	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✓	*Cisco*	*US*
U_{16}	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	*Outlook*	*US*
U_{17}	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	*Outlook*	*US*
U_{18}	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	*Outlook*	*US*
U_{19}	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	*Outlook*	*US*
U_{20}	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	*Outlook*	*US*
U_{21}	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	*Proofpoint*	*US*
U_{22}	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	*Outlook*	*US*
U_{23}	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	*Outlook*	*US*
U_{24}	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	*Outlook*	*US*
U_{25}	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	*Outlook*	*US*
U_{26}	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	*Sendmail*	*US*
U_{27}	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	-	*US*
U_{28}	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	*Gmail*	*US*
U_{29}	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	*Proofpoint*	*US*
U_{30}	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	*Outlook*	*US*
U_{31}	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	-	*US*
U_{32}	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	*Gmail*	*US*
U_{33}	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	*Proofpoint*	*US*
U_{34}	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✓	-	*US*
U_{35}	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	*Outlook*	*US*
U_{36}	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	*Gmail*	*US*
U_{37}	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	*Outlook*	*US*
U_{38}	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	*Postfix*	*US*
U_{39}	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	*AppRiver*	*US*
U_{40}	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	*Proofpoint*	*US*
U_{41}	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	*Mimecast*	*UK*
U_{42}	✓	✗	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	-	*DE*
U_{43}	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	-	*DE*
U_{44}	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	*Outlook*	*CN*
U_{45}	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	*Coremail*	*CN*
U_{46}	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	*Postfix*	*CN*
U_{47}	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	*Coremail*	*CN*
U_{48}	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✓	*Cisco*	*AU*
Total	**23**	**8**	**5**	**6**	**22**	**6**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**23**	**9**	**5**

* : Vul column indicates whether email services are vulnerable to SMTP smuggling attacks.
✓: Vulnerable. ✗: Not Vulnerable. - : Email providers can not be identified (e.g., self-hosted).

Table 10: Overall Results for Receiving MTAs of 48 University Email Services.

Test	Name	Version	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}	A_{13}
Receiving MTA Test	Apache james	2.3.2.1	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	Citadel	latest+	✂	☆	✗	✗	✂	✗	✂	✂	✂	✂	✂	✂	✂
	Exim	4.97.0	✓	☆	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
	Exim	4.97.1	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	Axigen	10.4.3	✓*	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	Axigen	10.5.25	✓*	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	Haraka	3.0.3	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
	Sendmail	8.14.7	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
	Sendmail	8.15.2	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
	Sendmail	8.17.2	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
	Sendmail	8.18.1	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	Opensmtpd	6.8.0	✗	☆	✗	✗	✗	✗	☆	☆	☆	☆	☆	☆	☆
	Postfix	3.5.8	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
	Postfix	3.9.0	✂	✂	✗	✗	✂	✗	✗	✗	✗	✗	✗	✗	✗
Sending MTA Test	Apache james	2.3.2.1	❍	❍	❍	❍	❍	❍	✓	✓	✓	✓	✓	✓	✓
	Citadel	latest+	✂	☆	❍	❍	✂	❍	✂	✂	✂	✂	✂	✂	✂
	Exim	4.97.0	✓	☆	❍	❍	✓	❍	✓	✓	✓	✓	✓	✓	✓
	Exim	4.97.1	☆	☆	❍	❍	☆	❍	✓	☆	☆	☆	✓	☆	✓
	Axigen	10.4.3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Axigen	10.5.25	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Haraka	3.0.3	✓*	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Sendmail	8.14.7	✓	☆	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Sendmail	8.15.2	✓	☆	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Sendmail	8.17.2	✓	☆	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Sendmail	8.18.1	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗
	Opensmtpd	6.8.0	✗	☆	✓	✓	✗	✓	☆	☆	☆	☆	☆	☆	☆
	Postfix	3.5.8	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Postfix	3.9.0	✂	✂	☆	☆	✂	☆	✓	✓	✓	☆	☆	✓	☆

✓: Vulnerable. For sending MTA tests, it means email services send out the original email data with the payload we inserted without filtering.
✓: Vulnerable. Email servers will send multiple emails after processing.
* : Axigen will only show the smuggled email and drop the first email. Haraka will only send the smuggled email out.
✗: Rejected. ✗: Not Vulnerable. ❍: Dot Stuffing. ☆: String Filtering. ✂: Truncated.
+ : We tested the Citadel software in July 2024.

Table 11: Overall Results for Multi-version Open-source Email Software.