CoordMail: Exploiting SMTP Timeout and Command Interaction to Coordinate Email Middleware for Convergence Amplification Attack

Ruixuan Li*†, Chaoyi Lu‡, Baojun Liu*†, Yanzhong Lin§, Qingfeng Pan§ and Jun Shao¶‖
*Tsinghua University, †Beijing National Research Center for Information Science and Technology,
‡Zhongguancun Laboratory, §Coremail Technology Co. Ltd, ¶Zhejiang Gongshang University,
‖Zhejiang Key Laboratory of Big Data and Future E-Commerce Technology
*lirx25@mails.tsinghua.edu.cn, *lbj@tsinghua.edu.cn, †lucy@zgclab.edu.cn,
‡{tim, pqf}@coremail.cn, §¶chn.junshao@gmail.com

***Abstract*—This paper introduces a novel and powerful email convergence amplification attack, named CoordMail. Traditional email DoS attacks primarily send spam to targeted mailboxes, with little ability to affect email servers' operation. In contrast, CoordMail exploits the inherent properties of the SMTP protocol, i.e., long session timeouts and client-controlled interactions, to cleverly coordinate reflected emails from various email middleware and eventually direct them to an incoming mail server simultaneously. As a result, the amplification capabilities of different email middleware are concentrated to form highly amplified attack traffic. From the SMTP session state machine and email reflection behaviors, we identify many real-world email middleware suitable for CoordMail, including 10,079 bounce servers, 584 open email relays, and 6 email forwarding providers. By building SMTP command sequences, CoordMail can maintain prolonged SMTP communications with these middleware at an extremely low rate and control them to reflect emails steadily at any given moment. We show that CoordMail is effective at a low cost: 1,000 SMTP connections can achieve more than 30,000 times of bandwidth amplification. While most existing security mechanisms are ineffective against CoordMail, we propose feasible mitigations that reduce the convergence amplification power of CoordMail by tens of times. We have responsibly reported CoordMail to email middleware and popular email providers, some of which have accepted our recommendations.**

I. Introduction

Email is a vital medium for global online communication, providing authentication functions for various network applications and security mechanisms [1]. Traditionally, email denial-of-service (DoS) techniques are mainly implemented by overwhelming targeted mailboxes with spam messages, known as *EmailBomb* [2], [3]. Various types of email middleware are leveraged for sending spam, such as website forms and subscription lists. Eventually, mailboxes targeted by *EmailBomb* are filled with spam messages, and users are hindered from identifying important or malicious emails.

As the email ecosystem continues to centralize, many domains and users are relying on a limited number of email servers [4]. As a result, compared to bombing mailboxes with spam, generating massive amplification traffic from the email protocol (i.e., SMTP [5]) and reflecting it against email servers, aiming to exhaust their available bandwidth or clog their task queues, will have more severe consequences. When email servers fail to process or even drop incoming messages due to a DoS attack, it can lead to business disruption, compromised authentication mechanisms, or financial losses.

Key observation: SMTP connections can be coordinated to form massive email amplification attacks. By analyzing the SMTP session state machine, we find that the inherent properties of the SMTP protocol can be exploited to coordinate different SMTP connections, including *long session timeout* and *client-controlled interaction*. To elaborate, SMTP is a delay-tolerant protocol for transmitting large texts, typically with long session timeouts to ensure reliable delivery. In addition, it uses a client-controlled model where clients send SMTP commands sequentially to initiate and manage SMTP sessions, while email servers respond correspondingly to inform the session status. Therefore, clients are able to maintain multiple SMTP sessions simultaneously and control their progress and completion times.

Our study: a novel and powerful SMTP amplification attack targeting email servers. Exploiting the key observations, we propose an email convergence amplification attack called CoordMail. The goal of the attack is to flood incoming mail servers with massive email traffic, thus affecting their email reception capabilities. CoordMail establishes numerous SMTP connections with different email middleware at a low rate and coordinates reflected emails from all middleware to reach the victim simultaneously. As a result, the amplification capabilities of all middleware are concentrated on the victim. To launch CoordMail, an attacker simply generates an SMTP command sequence for each middleware, taking into account the attack timing, email reflection interval, and the

Network and Distributed System Security (NDSS) Symposium 2026
23 - 27 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.241414
www.ndss-symposium.org

middleware's SMTP session state machine.

We identify three types of commonly-deployed email middleware which can be exploited for CoordMail: bounce servers, open email relays, and email forwarding providers. These middleware share a common feature: after an original email is received via SMTP, it will be "fanned out" into one or more reflected emails by the middleware to specified targets. In addition, such middleware may produce amplification traffic by themselves, such as by including larger email content in the reflected email than the original email, or replicating one original email into multiple reflected ones to different targets.

Through large-scale measurement, we identify email middleware in the wild that are suitable for CoordMail. Such middleware must meet two requirements: supporting long SMTP session timeouts and fanning out reflected emails steadily. We find 10,079 bounce servers, 584 open email relays, and 6 email forwarding providers that can be exploited for attacks. They allow SMTP session timeouts exceeding 10 minutes, and the variance of the reflection interval is less than 500 microseconds. In particular, bounce servers are most effective for CoordMail because they can reflect larger email content.

Evaluation and practical considerations. The larger the number of coordinated email middleware and the longer the SMTP connection accumulation time (attack period), the higher the bandwidth amplification of CoordMail. For ethical reasons, we deploy dedicated servers to conduct small-scale experiments in the real world to demonstrate attack feasibility. Using only 20 SMTP connections with email middleware and 140 seconds as the attack cycle, the bandwidth concentration efficiency [6] of CoordMail reaches 3,801x. In our controlled environment experiment, CoordMail achieves 2.5 Gb/s of traffic burst through 1,000 SMTP connections and a 100-second attack cycle, with a concentration efficiency of 33,145x.

We also analyze how existing security mechanisms are mostly ineffective against CoordMail, including email sender authentication, host reputation checks, and rate limit. For example, reflected emails from 77.89% of bounce servers can pass SPF [7] or DKIM [8] verification, popular email blocklists (e.g., Spamhaus [9]) only include 5.3% of bounce servers, and outlook.com receives 87.77% of reflected emails. Exceptionally, DMARC [10] can intercept about 95% of reflected emails, making it an effective mechanism for blocking CoordMail traffic. However, due to the low deployment of DMARC in the real world [11], email vendors typically do not make it mandatory for senders [12], [13].

Mitigation and disclosure. Based on our analysis of the CoordMail principle, we suggest that the email middleware add a random delay in the process of reflecting emails. Experimental evaluation shows that increasing the random delay from 0 to 30 seconds can reduce the amplification capability of CoordMail by about 15 times. We have responsibly reported CoordMail to the affected email middleware and 14 popular email providers with mitigation recommendations. We have received valid responses from 91 email middleware, with 22 administrators indicating that issue fixes are in progress or complete. In addition, 8 email providers acknowledged CoordMail, and proton.me plans to improve its email service.

Contributions. Contributions of this paper are as follows:

- *Novel attack*. By exploiting the SMTP timeout and command interaction, we propose a powerful email convergence amplification attack, which can achieve 30k+ times bandwidth concentration efficiency.
- *Measurement and evaluation*. We collect email middleware on a large scale and analyze their attack applicability. We comprehensively assess the defense effect of existing security mechanisms against CoordMail.
- *Mitigation and Responsible Disclosure*. We provide mitigation measures to defend CoordMail and disclose vulnerabilities to email middleware and popular email providers.

II. Background and Related Work

This section first describes the SMTP session state machine. Then, we introduce three common types of email middleware and existing security mechanisms to prevent traditional email reflection attacks. Finally, we summarize prevalent reflective DoS attack techniques.

A. SMTP Session State Machine

SMTP is an Internet standard protocol used for email transmission [5]. Figure 1 depicts the typical process of email delivery. To begin, the sender delivers the email to the outgoing mail server through SMTP (❶). Then, the outgoing mail server negotiates an SMTP session with the incoming mail server to transmit the email (❷). Finally, the recipient obtains the email from the incoming mail server (❸). Considering that the attack we proposed requires establishing and managing SMTP sessions with the email middleware via SMTP commands, we provide an in-depth analysis of the SMTP session state machine in the following.

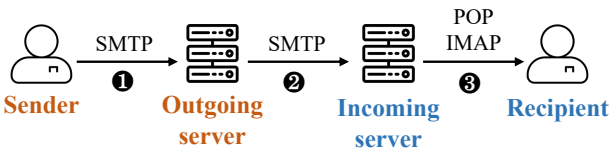

Figure 1. Typical process of email delivery.

The SMTP communication process is a **client-controlled interaction**. Specifically, the client sends SMTP commands sequentially to initiate and manage the SMTP session. Accordingly, the server responds with status messages to inform the SMTP session's state, which mainly consists of two types: successful action (e.g., 250 OK) and non-delivery report (e.g., 550 User does not exist). The server maintains a state machine for each SMTP client and transitions to the corresponding state upon receiving different SMTP commands. Figure 2 shows a simplified SMTP session state machine for the SMTP server, beginning after the completion of the TCP handshake. The SMTP server's state can be categorized into two types: the *Necessary* and *Temporary* SMTP session states. When the

server receives a *Mandatory* command, it enters the *Necessary* state; when the server receives a *Non-mandatory* command, it enters the *Temporary* state.

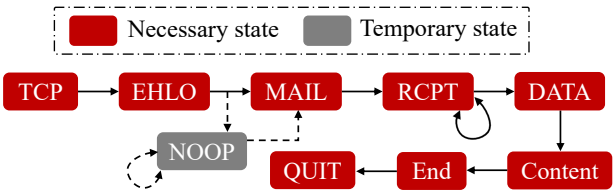

Figure 2. Simplified SMTP session state for SMTP servers.

Mandatory commands are SMTP commands that the client must send to complete the SMTP session, in the following order: `EHLO`, `MAIL`, `RCPT`, `DATA`, and `QUIT`. After receiving the `DATA` command, the server accepts the email content (`Content` state) until the client sends the termination signal (`\r\n.\r\n`), at which point the server enters the `End` state. The client can send multiple `RCPT` commands to specify different recipients for an email. After receiving the `QUIT` command, the server determines that the SMTP session is over. *Non-mandatory* commands are categorized into four main types [14]: optional (e.g., `NOOP`), obsolete (e.g., `TURN`), private (e.g., `XADR`), and invalid (e.g., `ABCD`). Typically, the client can send *Non-mandatory* commands at any time when the server is not in the `DATA` or `Content` states. The server may limit the number of *Non-mandatory* commands a client can send during a single SMTP session.

Furthermore, SMTP is a large text transmission protocol with high compatibility and latency tolerance. To ensure reliable delivery of email content, SMTP servers typically support **long session timeouts**. The server sets a timeout for each SMTP session state, with the duration varying between different servers. In addition, the server may impose a total timeout on the SMTP session to prevent excessive resource consumption. RFC 821 [5] recommends a long timeout duration (e.g., 5 minutes) for each SMTP session state.

B. Email Middleware and Email Reflection

The definition of email middleware in this paper is: an entity that automatically sends emails to the specified target after receiving emails from the originator. Figure 3 illustrates the email reflection process of email middleware. There are two *deployment types* for email middleware: *separation* and *integration* mode. In *separation* mode, the front node of the middleware is responsible for receiving original emails from the originator through the SMTP session. The back node establishes the SMTP connection with the target server and sends the reflected emails. Front nodes are typically associated with MX records for middleware domains, while back nodes are selected by the middleware. To ensure the performance and stability of the email service, large providers typically deploy multiple back nodes [15]. In *integration* mode, the middleware handles both receiving and sending emails through a single node. In the following, we introduce three common types of email middleware.

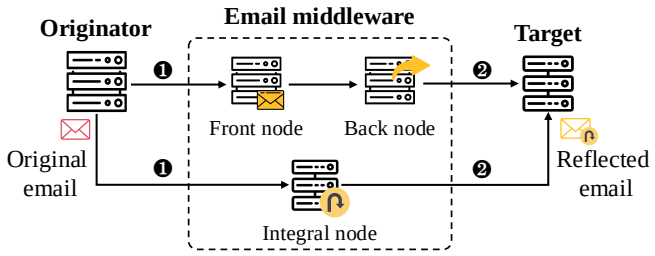

Figure 3. Email reflection process of email middleware.

• **Bounce server**. The traditional expectation within the community is that email should either be delivered or bounced, never silently lost. Therefore, some email systems are improperly configured to *receive any email during the SMTP communication* and then filter out undeliverable emails. Finally, the bounce server sends the original email content with the delivery failure reason to the originator, that is, the email address specified in the `MAIL FROM` command during the original SMTP session [16].

• **Open email relay**. Properly configured email servers should only transmit emails for domains or email addresses within their service scope. However, an open relay receives email from any sender and delivers it to the destination designated by the originator, that is, the email address specified in the `RCPT TO` command during the original SMTP session [17].

• **Email forwarding provider**. Email forwarding is a service that automatically forwards received emails to other email addresses. Users need to configure the forwarding relationship at the provider, specifying the source email address to be forwarded and the forwarding destination [18].

Threat and protection. The auto-reflective nature of the above email middleware is often exploited by attackers to send spam. Over 20 years ago, Bass et al. [19] and Frei et al. [16] provided a detailed analysis of the workflow and hazards of the bounce backscatter attack. In particular, Frei et al. [16] measured amplification metrics for 7,458 bounce servers, pointing out that hundreds of bounced emails can be generated from a single original email containing multiple invalid recipients. Email middleware is also exploited to distribute phishing and extortion emails [20], [21]. In addition, Shen et al. [22] and Liu et al. [18] revealed that forwarding services can be used to implement email sender spoofing attacks.

Currently, several mitigation measures have been proposed to defend against email reflection attacks. We describe three common types of security mechanisms below.

• **Email authenticity**. The authentication mechanisms can detect forged reflected emails, including DomainKeys Identified Mail (DKIM) [8], Sender Policy Framework (SPF) [7], Domain-based Message Authentication, Reporting, and Conformance (DMARC) [10]. DKIM ensures email content integrity through digital signatures, and SPF allows domains to specify which hosts are authorized to deliver email on their behalf. The DMARC mechanism builds on DKIM and SPF, and additionally requires the alignment of authenticated

identifiers, i.e., the identity information in the `From` field must align with the authenticated identifiers of SPF or DKIM.

• **Host reputation**. Email middleware involved in malicious activities usually has a poor reputation. Incoming servers can promptly interrupt SMTP connections with malicious email middleware by querying blocklists. The email server can block connections with email middleware by querying blocklists during SMTP communications. DNS-based blocklist (DNSBL) is a popular type of blocklist that contains malicious hosts and is widely used by email providers [15], [23], such as Yahoo [24] and Outlook [25]. In addition, Greylisting is based on the host delivery behavior [26]. The server can use Greylisting to reject the email when the host first connects and accept it when the host attempts to deliver it again later.

• **Rate limit**. To defend against bursts of email reflection attack traffic, email providers implement two main types of rate limits. On the one hand, email providers can limit the rate at which an IP address sends emails to them [27], i.e., IP sending rate limit. On the other hand, email providers can limit the rate at which a user's mailbox receives emails [28], i.e., mailbox receiving rate limit.

C. Overview of Reflective DoS Attack Techniques

DoS attacks are prevalent and serious network threats that usually flood the target system with large-scale traffic [29], [30]. Attackers often use controlled machines or reflectors to generate massive traffic directed at the victim's side. Controlled machines typically consist of botnets infected with malware [31]. However, botnets are costly to acquire and relatively easy to identify. In contrast, reflectors are usually normal and legitimate devices in the wild [32].

Common reflective DoS techniques. Reflective DoS generates high-magnification attack traffic primarily by increasing the size and number of response packets. Scholars have pointed out that various mechanisms and instructions can be used to increase the size of the response. For example, querying DNS TXT and DNSSEC records [33], [34], and requesting NTP synchronization servers [32], [35]. Regarding the increase in the number of packets, Afek et al. [36] and Moura et al. [37] proposed using DNS NS and CNAME records to generate a high volume of responses, eventually achieving amplification of thousands of times.

To amplify the magnifying power of reflectors, more complex and subtle DoS strategies are being proposed. Pulsing DoS (PDoS) is a powerful variant that subjects the target to intermittent bursts of traffic, similar to periodic pulses [38]. PDoS attacks send packets at a low rate while aggregating attack traffic at the victim's side, producing bandwidth amplification of up to tens of thousands of times. Numerous studies [39]–[41] have shown that PDoS attacks are highly damaging to various network services and can even lead to the permanent disruption of target systems. Guo et al. [42] proposed a PDoS attack based on path delay, called *CDN-Convex*. By exploiting the path delay between different CDN nodes and the target server, *CDN-Convex* concentrates traffic from multiple CDN nodes to the target server within a short time. Li et al. [6] reported the *DNSBomb* attack, which uses DNS response timeouts and query aggregation to cause victims to receive a flood of DNS messages simultaneously. Additionally, researchers proposed DoS techniques for constructing DNS response amplification and query loop links to execute powerful traffic amplification attacks, such as *TsuKing* [43].

Common email DoS attack: targeting user mailboxes. Few studies focused on DoS techniques in the email ecosystem. The most well-known is *EmailBomb*, which floods the victim's mailbox with spam by submitting fake registration requests to website forms or subscription lists. As a result, the victim cannot locate important or malicious emails in the deluge of unsolicited emails [44]. Jakobsson et al. [3] described the construction of *EmailBomb* and analyzed website form reflectors in the wild. They used 2,000 forms to fill the user's inbox with 2MB of data in about an hour. Moreover, Schneider et al. [2] launched *EmailBomb* attacks on their mailboxes through the darknet and analyzed the real-world harm of attacks. They found that new abused form reflectors emerged daily, and most of the reflected emails reached the victim's mailbox the day after the attack.

III. CoordMail Attack and Construction

This paper reports a novel and powerful email amplification attack that can achieve tens of thousands of bandwidth amplification rates. Because the amplification is achieved by coordinating different email middleware, we refer to this attack as **Coord**inate E**mail** (CoordMail). This section first describes the threat model of CoordMail and then details the construction of the attack. Finally, we compare CoordMail with common reflective DoS attack techniques.

A. Threat Model

CoordMail exploits the SMTP session timeout and SMTP command interaction to coordinate reflected emails from different email middleware, causing them to reach the victim simultaneously. Ultimately, CoordMail disrupts the availability of the incoming mail server through explosively amplified traffic. CoordMail is based on the key concept of time convergence, leveraging two inherent properties of the SMTP protocol: long session timeouts and client-controlled interactions. Specifically, after selecting the attack moment, the attacker gradually accumulates SMTP sessions with various email middleware, maintaining all sessions through SMTP command sequences, i.e., the SMTP command sending order and the corresponding sleep intervals. When approaching the attack moment, the attacker completes the SMTP session with each middleware, and reflected emails from all middleware are aggregated to the victim. The attack cycle can be arbitrarily specified by the attacker.

The requirements for an attacker to carry out CoordMail are extremely low. First, the attacker only needs a low-bandwidth SMTP server to send email, which is unlikely to produce detectable network-traffic spikes. To evade cloud provider monitoring that flags high CPU utilization, the attacker can coordinate SMTP connections using their own local

machines. Moreover, attackers can deploy multiple servers to distribute load and reduce the detection risk and per-host resource usage. Second, the attacker collects a batch of email middleware that supports long SMTP session timeouts and stably sends reflected emails.

The victims of CoordMail are IP addresses with email receiving capabilities, such as incoming mail servers of email service providers, forwarding platforms, and websites. CoordMail fills the email processing queue or consumes the available bandwidth of the victim server. Given the high centralization of email services [45], successful DoS attacks on popular incoming servers can impact numerous domains and users.

B. Attack Construction

The amplification factor of traditional email backscatter attacks depends primarily on the capabilities of email middleware [16], making it difficult to generate extremely high-bandwidth traffic that can quickly overwhelm the target server. We recognize that the SMTP session timeout and SMTP command interaction can be exploited to combine the amplification capabilities of different email middleware.

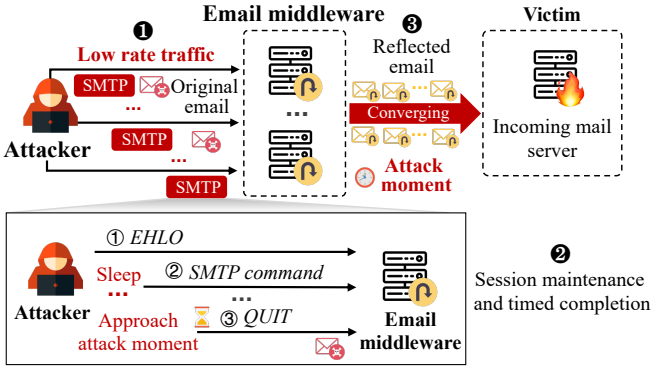

Figure 4. Workflow of the CoordMail attack.

Attack overview. Figure 4 illustrates the workflow of CoordMail, which consists of three key steps. First, the attacker gradually establishes SMTP connections with email middleware. Second, the attacker maintains the SMTP connections and completes the sessions with various middleware at specific times. Third, reflected emails from all middleware reach the victim simultaneously.

• **Low-speed connection to email middleware** (❶). The attacker first selects an arbitrary attack moment and a batch of email middleware. Then, the attacker sequentially establishes SMTP connections with email middleware at a low rate, completing the process before the attack moment. By establishing only one or a few connections with each email middleware, the attack avoids being blocked for connecting too quickly.

• **Session maintenance and timed completion** (❷). The attacker maintains SMTP connections with each email middleware until close to the attack moment. To achieve this, the attacker creates SMTP command sequences to specify the sending order of SMTP commands and their corresponding sleep intervals. Before the SMTP session state machine expires, the attacker sends an SMTP command to refresh the state timeout. Simultaneously, the attacker calculates when to complete the SMTP session with each middleware, considering the attack moment, path latency, and email reflection interval. When the designated moment arrives, the attacker sends the `QUIT` command to each email middleware.

• **Concentrate reflected emails at victim** (❸). After receiving the original emails from the attacker, the email middleware sends reflected emails to the victim. By specifying the time to send the `QUIT` command, the differences in reflection interval and path latency between various email middleware are offset. As a result, all reflected emails are aggregated at the victim side within a short period.

Below, we present the details of CoordMail construction, including *how to construct original emails* sent to middleware and *how to build SMTP command sequences* to manage the SMTP session with email middleware.

Construct original email. The reflection behavior of different email middleware varies [18], so it is necessary to construct different original emails. We elaborate on three types of original emails and the process by which attackers use middleware to reflect emails to the target server. Figure 5 shows examples of original and reflected emails during the attack. Note that the examples we provided do not cover all cases.

Original email	Reflected email
MAIL FROM: *exist@victim* ***RCPT TO:*** *non-exist@bounce* ***From:*** *exist@victim* ***To:*** *non-exist@bounce* [Original email body]	***MAIL FROM:*** <> ***RCPT TO:*** *exist@victim* ***From:*** *exist@bounce* ***To:*** *exist@victim* [Bounce body + Original body]

(a) Bounce server

MAIL FROM: *random@attacker* ***RCPT TO:*** *exist@victim* ***From:*** *random@attacker* ***To:*** *exist@victim* [Original email body]	***MAIL FROM:*** *random@attacker* ***RCPT TO:*** *exist@victim* ***From:*** *random@attacker* ***To:*** *exist@victim* [Original email body]

(b) Open email relay

MAIL FROM: *random@attacker* ***RCPT TO:*** *exist@forward* ***From:*** *random@attacker* ***To:*** *exist@forward* [Original email body]	***MAIL FROM:*** *[code]@forward* ***RCPT TO:*** *exist@victim* ***From:*** *random@attacker* ***To:*** *exist@forward* [Original email body]

(c) Email forwarding provider

Figure 5. Examples of original emails (sent to email middleware) and reflected emails (sent to victim server).

• **Bounce server**. The `MAIL FROM` field indicates the "reverse-path" of the email [5], which the bounce server uses to determine the reflection destination. Therefore, the attacker specifies the victim's address in the `MAIL FROM` field of the original email, and sets the `RCPT TO` field to a non-existent user to ensure the original email is deemed undeliverable

by the bounce server. After that, the bounce server sets the RCPT TO field to the victim's address when sending the bounced email. Because the victim server may reject bounced emails destined for non-existent users, the attacker should ensure that the victim's email address is an existing account. Furthermore, many bounce servers receive and reflect emails that fail SPF checks (see Section IV-A), so the original email is not rejected for spoofing the victim's domain. Moreover, bounce servers usually set the MAIL FROM field of the bounced emails to empty (<>) to avoid email delivery loops. Bounced emails typically contain additional email content to explain the reason for rejecting the original email.

• **Open email relay**. The open relay does not change the email routing path, and just delivers emails according to the destination of the original email. Therefore, the attacker can direct the reflected email to the victim server by setting the RCPT TO field of the original email to the victim's address. To ensure that the reflected emails pass authenticity verification, the attacker configures valid DKIM and DMARC records for the domain specified in the MAIL FROM field and includes the IP address of open relays in the SPF record. Similar to the bounce server, the RCPT TO field of the original email should be set to the existing victim's email address.

• **Email forwarding provider**. The forwarding provider determines the destination of the reflected email based on the user's configuration. The attacker can instruct the forwarding provider to send emails to the victim in two ways. One way for the attacker to do this is to obtain a legitimate account with the forwarding provider and configure the received email to be forwarded to the victim. The other is that the attacker exploits the forwarding relationships of the provider's existing account. Therefore, the attacker needs to find email addresses that are configured to forward emails to the victim. After that, the attacker sets the RCPT TO field of the original email to the email address managed by the forwarding provider. Finally, the forwarding provider automatically sends emails to the victim, and the domain in the MAIL FROM field of the reflected email is the forwarding provider's domain.

To increase attack amplification, the attacker can instruct the email middleware to send multiple reflected emails for a single original email. The attacker designates many recipients for the original email by sending multiple RCPT commands, and the middleware will send one reflected email for each recipient. For bounce servers and open relays, recipient addresses can be generated from a batch of random usernames or subdomains. For forwarding providers, recipient addresses correspond to different legitimate accounts.

Build SMTP command sequence. Through SMTP commands, CoordMail can maintain and coordinate SMTP sessions with front nodes of different email middleware. We first analyze the duration of each stage of an email reflection attack, as shown in Figure 6, including the following key moments and durations.

• M_{start}: The moment the attacker completes the TCP handshake with the middleware.

• M_{atta}: The moment when the victim is attacked.

• M_{quit}: The moment the attacker sends QUIT command.

• D_{prod}: The duration of the middleware to produce the reflected email.

• D_{orig}: The path latency between attacker and middleware.

• D_{refl}: The path latency between middleware and victim.

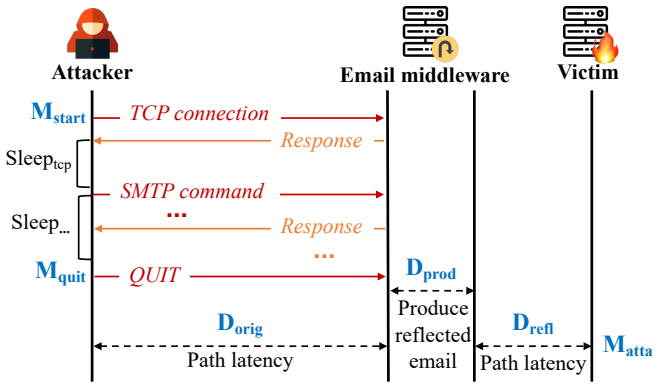

Figure 6. Time diagram of an email reflection attack.

The values of M_{start} and M_{atta} are specified arbitrarily, M_{quit} can be calculated, D_{prod}, D_{orig}, D_{refl} can be measured. To calculate M_{quit}, the attacker first measures the path latency between its server and the email middleware, i.e., D_{orig}. Then, the attacker sends an email to the middleware and directs the reflected email to the controlled server. The attacker records the moment of sending the QUIT command (M_1) and the moment of receiving the SMTP connection from the middleware (M_2). D_{prod} is calculated as follows: $D_{prod}=(M_2-M_1)-(2*D_{orig})$. In terms of measuring the path latency between the email middleware and the victim (D_{refl}), the attacker can utilize the King method [46], which is also used in other DoS attacks [6], [42] to approximate the latency between Internet hosts. Then, the attacker calculates the M_{quit} as follows: $M_{quit}=M_{atta}-D_{refl}-D_{prod}-D_{orig}$.

After determining M_{quit}, the attacker needs to maintain the SMTP session with the email middleware through the SMTP command sequence. To ensure that the QUIT command is sent in time at the M_{quit}, the attacker sends all the email content before the M_{quit} and sleeps until the M_{quit} arrives. Algorithm 1 in Appendix A illustrates the method for building an SMTP command sequence, including the SMTP commands sent to the email middleware and the corresponding sleep time after each sending. The key parameters of email middleware are as follows:

• T_{targ}: Target maintenance time of SMTP session.

• T_{nece}: Total timeout of *Necessary* SMTP session states.

• D_{nece}: Max duration of the *Necessary* SMTP session state.

• D_{temp}: Max duration of the *Temporary* SMTP session state.

• N_{temp}: Max number of consecutive *Temporary* SMTP session states.

In the above parameters, T_{targ} is defined according to the attack requirements ($M_{quit}-M_{start}$), while the other parameters are different for each email middleware. The attacker prioritizes maintaining the SMTP session by sending *Mandatory*

SMTP commands. In particular, the attacker assigns sleep time for each *Mandatory* command according to the ratio of the max duration of the corresponding *Necessary* SMTP session states to T_{nece}. If T_{nece} is less than T_{targ}, the attacker supplements the SMTP session duration by *Non-mandatory* commands until T_{targ} is satisfied. Note that the sleep time and number of all commands are within the requirements of D_{nece}, D_{temp}, and N_{temp}.

For example, D_{nece} is {TCP: 30, EHLO: 30, MAIL: 30, RCPT: 30, DATA: 60, Content: 60, End: 30}, D_{temp} is {NOOP: 60}, N_{temp} is {NOOP: 30}. If T_{targ} is 180, the SMTP command sequence is {(TCP, 20), (EHLO, 20), (MAIL, 20), (RCPT, 20), (DATA, 40), (Content, 40), (End, 20)}; if T_{targ} is 280, the SMTP command sequence is {(TCP, 30), (EHLO, 30), (MAIL, 30), (RCPT, 30), (NOOP, 10), (DATA, 60), (Content, 60), (End, 30)}. Following the SMTP command sequence, the attacker sends SMTP commands in order and then sleeps for the corresponding time. When the M_{quit} arrives, the attacker immediately sends the QUIT command to the email middleware.

C. Comparison with Common Reflective DoS Techniques

Table I presents a comparison of the attack technique of CoordMail and common reflective DoS attacks. Traditional reflective DoS attacks require spoofing IP addresses to target the victim server, so attackers typically use stateless protocols, such as DNS and NTP, to transmit the attack traffic [32], [35]. The idea of CoordMail is inspired by PDoS attacks (time convergence), enabling it can achieve a high bandwidth amplification effect. Compared to traditional email-based reflection attacks, such as bounce backscatter and *EmailBomb*, CoordMail can generate explosive traffic bursts against email servers.

Furthermore, PDoS attacks utilizing DNS and CDN depend on specific software behaviors (e.g., DNS timeout) and path delays (e.g., HTTP transmission). For example, *DNSBomb* and *CDN-Convex* attacks typically require accumulating connections to middleware (DNS resolvers or CDN nodes) over a period, usually within 60 seconds [6], [42]. CoordMail is a time-convergent reflective pulsing DoS whose novelty lies in its carrier and techniques. First, DoS risks in the email ecosystem have not been thoroughly explored, and previous pulsing DoS carriers mainly include DNS, CDNs, etc. Our work demonstrates a new attack surface where SMTP can be abused for PDoS. Second, email middleware is diverse and independently managed, which makes traffic coordination challenging. CoordMail utilizes SMTP's built-in interaction commands and timeouts to manage session state machines of various middleware. Unlike previous time-convergent pulsing attacks that rely on path delays [38], [42], CoordMail enables finer-grained, more stable coordination.

Table I
Comparison of common reflective DoS attack techniques.

Attack	Carrier	w/o IP spoof[1]	Amplification
Traditional reflection [34], [35]	DNS NTP	✗	<1K BAF[2]
TsuKing [43]	DNS	✔	1K-40K PAF[2]
CDN-Convex [42]	CDN	✔	1K-5K BCE[2]
DNSBomb [6]	DNS	✗	1K-30K BCE
Bounce backscatter	SMTP	✔	50 BAF[3]
CoordMail	SMTP	✔	30K BCE[3]

[1] ✗ means IP address spoof is required; ✔ means no need.
[2] BAF: bandwidth amplification factor. PAF: packet amplification factor. Bandwidth centralization efficiency (BCE) is used to evaluate the effectiveness of PDoS attacks and is also often referred to as BAF [6].
[3] See Section V-A for evaluation experiments.

IV. Collecting Exploitable Email Middleware

In this section, we first collect email middleware in the real world through large-scale scanning. Then, we examine the attack metrics of email middleware and select those that are suitable for constructing CoordMail.

A. Finding Email Middleware in the Wild

CoordMail attack coordinates SMTP sessions with front nodes of email middleware and cannot directly select back nodes. In this paper, we use front nodes to count the number of bounce servers and open email relays, and use the domain names to count the number of forwarding providers. Moreover, the number of back nodes we found is only a lower bound.

Bounce server. In order to complete the SMTP session with the bounce server, we need to obtain the domains they serve. As such, we first extensively collect domains, then send original emails to them and monitor reflected emails.

We obtain domains from popular domain lists and leaked email datasets. Specifically, we downloaded three Top 1M popular domain lists on March 1, 2024, including Tranco [47], Umbrella [48], and Majestic [49]. We then removed domains without MX records and finally collected 1,064,761 unique domains. Furthermore, we obtained the Adobe email leakage dataset [50] and extracted 9,208,073 domains. After removing domains without MX records, we obtained 5,027,648 unique domains. Then, we purchased a domain name and deployed the email service on a dedicated cloud server. Following this, we send an original email to each collected domain. The recipient of the original email is a non-existent user under the target domain, and the sender is the email address containing *random identifiers* under our domain. To identify bounce servers that receive emails from spoofed sender domains, we configure incorrect SPF, DKIM, and DMARC records for our domain. Because recipients of bounced emails correspond to senders of original emails, we can identify bounce servers by matching *random identifiers*.

As shown in Table II, we collect 19,184 bounce servers that served 25,821 domains and deployed 11,114 back nodes. By analyzing the MX records of bounce domains, we find that they are mainly hosted by four Japanese email providers

Table II
STATISTICS ON IDENTIFICATION AND ATTACK SUITABILITY OF EMAIL MIDDLEWARE IN THE REAL WORLD.

Email Middleware	Front	Back	CoordMail number	CoordMail suitability[1]
Bounce server	19,184	11,114	10,079	●
Open email relay	1,299	1,279	584	◐
Forwarding provider	10	9,581	6	◐

[1] ● means high suitability; ◐ means moderate suitability.

and two security organizations, which together account for 40.75%. Moreover, 112 of Tranco top 10K domains offer bounce services. Regarding the content of bounced emails, most explain the reasons for delivery failures, with 22,359 (86.59%) bounce domains including the original email content in bounced emails. Furthermore, 17,998 (93.82%) bounce servers set the `MAIL FROM` field of bounced emails to empty.

We further analyze the back nodes responsible for sending bounced emails. By querying the IP geodatabase [51], we find that the back nodes of bounce servers are widely distributed globally, covering 171 countries and 4,728 autonomous systems (ASes). The top three ASes are *AS8075 Microsoft Corporation*, *AS16509 Amazon.com, Inc*, and *AS16276 OVH SAS*. To understand the infrastructure of the back nodes, we analyze their domains and software through the `Received` headers in bounced emails. Among the 6,768 back nodes that provide valid domain information, 612 are deployed on Outlook, and 334 are deployed on Proofpoint. The software running on 4,684 (42.14%) back nodes is Postfix. In addition, we analyze the *deployment type* of bounce servers. For 21,771 bounce domains that we can obtain front and back node domains, 15,970 (73.56%) of them belong to the *integration mode*, i.e., the front and back nodes are deployed in the same second-level domain (SLD).

Open email relay. Open relays transmit the original email directly, so we do not need to collect the domains they serve. We first scan IP addresses with open TCP/25 ports and then send the original emails to them.

On March 10, 2025, we used Zmap [52] to collect IP addresses with open TCP/25 ports on the Internet. Next, we send the original email to each IP address, and the sender's and recipient's domains are deployed on two servers that we control. The original emails sent to different IP addresses are identified by *unique* email addresses. Finally, we analyze incoming emails on our recipient server and collect open relays by matching the *unique* email addresses.

In total, we collected 1,299 open email relays, corresponding to 1,279 back nodes. The front and back nodes of 931 (71.67%) open relays share the same IP address. The back nodes are distributed across 57 countries, with China and the United States accounting for 35.41%. Furthermore, 742 (58.01%) back nodes are running the Postfix software. Almost all emails sent by open relays only add `Received` headers on the basis of the original emails.

Email forwarding provider. The email forwarding relationship depends on the user's configuration, that is, <*forwarding initiator, forwarding destination*> pair. The initiator is the valid email address of the forwarding provider, and the destination is the email address to which emails received by the initiator are forwarded. Referring to previous work [15], [22], we investigated ten popular email forwarding providers.

We first analyze the possibility of an attacker directly setting the victim's email address as the forwarding destination. As shown in Table VI of Appendix B, six providers allow users to set any email address as the forwarding destination without requiring authentication. These providers include outlook.com, hotmail.com, icloud.com, 163.com, 126.com, and yeah.net. The remaining four providers send verification emails to the forwarding destination to request the activation of the forwarding service.

Furthermore, the attacker can leverage existing forwarding relationships in the real world to execute CoordMail. This avoids the need to manually acquire a large number of accounts and can bypass the provider's bulk registration detection and forwarding verification. By analyzing forwarding behavior, we observe that 9 email providers use custom templates to generate `MAIL FROM` fields for forwarded emails. In collaboration with Coremail [53], a large email service provider in China, we identified 858,785 forwarding relationship pairs from one year's email reception logs. In particular, two business domains are associated with more than 10K forwarding relationships. Moreover, we observe that 9 forwarding providers deploy 9,581 back nodes by analyzing Coremail's email logs. Appendix B provides a detailed description of the identification process and analysis results for the forwarding relationships. In addition to using large-scale email logs, attackers can leverage email-tracking technology to identify forwarding relationships, as documented in [54].

B. Analyzing Attack Metrics for Email Middleware

Amplification capability. The amplification capability of email middleware is primarily reflected in two aspects.

- N_{pack}: The magnification of the total packet size in the reflected email session compared to the original email session, mainly due to the larger content in the reflected email body.

We find that the N_{pack} of bounce servers is relatively large, while the N_{pack} of open relays and forwarding providers is typically less than 2. Specifically, the N_{pack} of 8,834 (46.05%) bounce servers is more than 5 times larger compared to the original email communication. The packet magnification of 609 bounce servers exceeds 20, as they include a lot of unnecessary information in the content of bounced emails, such as attachments and images.

- N_{rcpt}: The number of reflected emails generated by email middleware for one original email, which is mainly achieved by specifying multiple recipients for the original email.

We send original emails with different recipients through the `RCPT` command to the email middleware and analyze the number of reflected emails we receive within 10 seconds. We find that most email middleware generates reflected emails for each different recipient in one original email. The difference

is that bounce servers usually consider email addresses with different usernames (before @) as different recipients, while most open relays send only one email to recipients with the same fully qualified domain name (FQDN). Therefore, the attacker needs to specify email addresses with different FQDNs (after @) as recipients in the original email. To achieve this, the attacker can generate many random subdomains for their domain and point the MX records of all subdomains to the victim's server. For forwarding providers, the attacker needs to set the recipients to different legitimate email accounts.

Because CoordMail cannot control all reflected emails corresponding to one original email reaching the victim at the same time, the excessive `Nrcpt` does not enhance the attack effect. We choose `Nrcpt` values of 5 and 10 for measurement. The results show that 4,218 bounce servers support `Nrcpt` of 5, and 3,719 support `Nrcpt` of 10; 868 open relays support `Nrcpt` of 5, and 398 support `Nrcpt` of 10; all 10 forwarding providers support `Nrcpt` of 10.

SMTP session timeout. CoordMail utilizes SMTP command sequences to maintain and coordinate SMTP communication. To this end, we measure the SMTP session timeouts for front nodes of email middleware. During communication with email middleware, we send SMTP commands and then hold the connection for a specified *duration*. If we successfully complete the SMTP session with the middleware afterward, we consider the corresponding SMTP session state timeout to exceed the specified *duration*. Note that we test the allowable duration for only one SMTP session state in each communication. Specifically, we test seven *Necessary* SMTP session states: `TCP`, `EHLO`, `MAIL`, `RCPT`, `DATA`, `Content`, `End`; and six *Temporary* SMTP session states: optional (`NOOP`, `HELP`, `VRFY`), obsolete (`TURN`), private (`XADR`), invalid (`ABCD`). Considering the cost of testing, we specify seven *durations* for testing: 5, 10, 30, 60, 120, 180, and 300 seconds. We also test the number of consecutive *Temporary* SMTP session states permitted by email middleware, up to 30 times.

We find that most email middleware supports long SMTP sessions. The maintenance time of each SMTP session state for 14,365 (74.88%) bounce servers and 1,004 (77.29%) open email relays exceeds 300 seconds. The SMTP session state timeout allowed by the 10 popular forwarding providers varies: gmail.com, outlook.com, hotmail.com, and icloud.com support session state timeouts longer than 300 seconds, while the remaining six providers typically set timeouts shorter than 60 seconds. The timeout for yeah.net is particularly short, set to only 10 seconds.

Figure 7 illustrates the number of consecutive *Temporary* SMTP session states supported by email middleware. `NOOP` is the most highly supported *Non-mandatory* command for email middleware. We discover that 85.38% of middleware allows clients to send at least 30 consecutive `NOOP` commands during an SMTP communication. In contrast, obsolete, private, and invalid commands are allowed in relatively small numbers. The allowable number of *Temporary* SMTP session states varies significantly among the 10 forwarding providers. All providers support 30 consecutive `NOOP` commands, and 139.com only accepts the `NOOP` command. The allowed number of consecutive times of other *Temporary* SMTP session states is typically no more than 10. A comprehensive list of SMTP session timeouts for email middleware is provided in Appendix C.

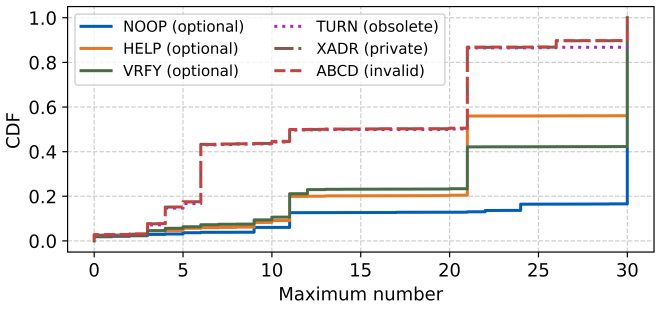

Figure 7. Maximum number of consecutive times for *Temporary* SMTP session states allowed by email middleware.

Additionally, we build SMTP command sequences as described in Section III-B to measure the total SMTP session timeout permitted by email middleware. Our measurements show that 13,425 (69.98%) bounce servers, 831 (63.97%) open email relays, and all ten email forwarding providers allow an SMTP session duration of more than 10 minutes. Overall, our results remain well below the maximum timeout and command count limits permitted for SMTP session states. Nevertheless, we demonstrate that a client can maintain SMTP communication with the email middleware for a sufficiently long period by leveraging SMTP command interaction.

Email reflection interval. CoordMail requires the email middleware to return the reflected email within a consistent and predictable interval. To evaluate this, we sent 20 original emails to each email middleware, then calculated the average and standard deviation of the reflection intervals. Our results show that 12,465 (64.98%) bounce servers, 753 (57.97%) open email relays, and 6 email forwarding providers exhibit a mean reflection interval of less than 5 seconds and a standard deviation below 500 milliseconds. Therefore, these email middleware systems are capable of sending reflected emails to target servers with relatively high temporal stability.

C. Selecting Suitable Email Middleware for CoordMail

CoordMail relies on time convergence to generate high-bandwidth attack traffic. It is essential to select appropriate email middleware to construct attacks. Specifically, we select middleware that meets the following two criteria: 1) support maintaining an SMTP session for more than 10 minutes and 2) exhibit an average reflection interval of less than 5s with a standard deviation below 500 milliseconds. Ultimately, we identify 10,079 (52.53%) bounce servers, 584 (44.96%) open email relays, and 6 email forwarding providers (gmail.com, qq.com, 163.com, 126.com, 139.com, sina.com) suitable for constructing CoordMail.

Furthermore, bounce servers are the most suitable email middleware for constructing CoordMail. First, due to the larger size of reflected emails, bounce servers offer

higher amplification potential. Second, unlike email forwarding providers, which require the attacker to actively register or discover forwarding relationships, the inherent backscatter behavior of bounce servers can be directly exploited. Third, the population of exploitable bounce servers is significantly greater than that of other types of email middleware.

V. Evaluation and Practical Consideration

This section first evaluates the traffic amplification effect of CoordMail in both real-world and controlled environments. Considering that certain practical factors may reduce the damage of attacks, we also analyze the effectiveness of mainstream security mechanisms in defending against CoordMail.

A. Evaluating the Traffic Amplification Effect

We first introduce metrics to assess the attack effect. Drawing on previous PDoS attack studies [6], [42], we use the bandwidth concentration efficiency (BCE), also often referred to as bandwidth amplification factor (BAF), to quantify the effect of CoordMail. BCE indicates the ability of the attack to aggregate traffic on the victim's side, i.e., the multiple of the peak attack traffic bandwidth over the victim's required bandwidth. The following metrics are defined to evaluate both the attacker's and the victim's sides:

- N_{midd}: The number of email middleware the attacker selects.
- N_{rcpt}: The number of different recipients of the original email sent to the email middleware.
- S_{atta}: The size of packets exchanged in an SMTP session between the attacker and the email middleware.
- S_{vict}: The size of packets exchanged in an SMTP session between the email middleware and the victim.
- T_{atta}: Time used when accumulating SMTP connections, i.e., attack cycle.
- T_{vict}: Time of arrival of reflected emails to the victim.

In the above metrics, we can specify the values of N_{midd}, N_{rcpt}, T_{atta}. For S_{atta}, S_{vict}, and T_{vict}, we measure them for each email middleware by simulating both the attacker and the victim. It is important to note that both outgoing and incoming packets are counted. The size variation of S_{atta} primarily depends on the number of `RCPT` commands sent to the middleware. By using the same metrics as previous studies evaluating PDoS attacks [6], [42], we define the BCE calculation of CoordMail as follows:

$$BCE = \frac{\sum_{i=1}^{Nmidd}(Svict_i \times Nrcpt_i)}{\sum_{i=1}^{Nmidd} Satta_i} \times \frac{Tatta}{Tvict}$$

By controlling the time when different email middleware return emails, T_{vict} is usually several hundred milliseconds. As such, the greater the number of coordinated middleware and the longer the SMTP connection accumulation time can result in the higher the BCE. We present the theoretical BCE of the CoordMail in Appendix D.

We realize that experiments targeting production email servers of large providers and real organizations would better illustrate CoordMail's practical impact. However, in accordance with research ethics, we cannot perform DoS attacks in the wild because the resulting large-scale traffic may disrupt victims' operations. For assessing PDoS, prior studies have established peer-accepted, ethically compliant procedures [6], [38], [42], mainly by measuring BAF/BCE through controlled victims. Following these works, we implement CoordMail in both real-world and controlled environments to assess the amplification effects. We restrict our use to a limited number of real-world email middleware to demonstrate the practical feasibility of the attacks. In addition, we simulate traditional bounce backscatter attacks in controlled networks to compare the impact of CoordMail.

Attacks in the real world. At first, we registered two cloud servers to serve as the attacker and victim, respectively. Each server is equipped with an 8-core, 2.5 GHz CPU, 8 GB of RAM, and a 15 Mb/s network bandwidth. Subsequently, we launch CoordMail against our servers, with attack metrics of N_{midd} of 20 and N_{rcpt} of 5. We prioritize constructing attacks through email middleware with high amplification capabilities, and use tcpdump [55] to capture the packets exchanged by our servers.

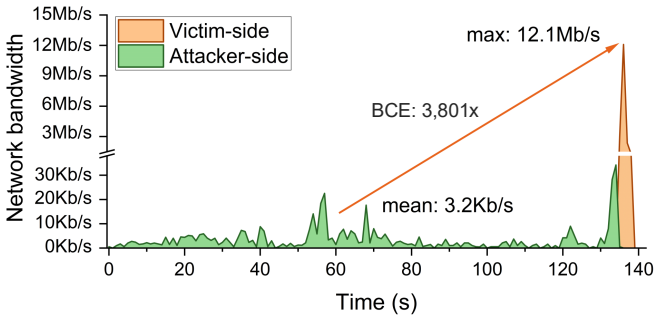

Figure 8. CoordMail experiment using email middleware in the real world (N_{midd} = 20, N_{rcpt} = 5).

Figure 8 presents the experimental results of the real-world CoordMail. We achieve a BCE of 3,801 by utilizing 20 email middleware and a 140-second attack cycle. The average bandwidth on the attacker's side is measured at 3.2 kb/s, while the victim's side can instantly reach 12.1 Mb/s. However, the BCE of the real-world CoordMail is lower than the theoretical BCE, which is 7,150. Such a discrepancy arises due to the fact that the victim completes the SMTP session interactively with the email middleware in stages. CoordMail can not guarantee that all packets in an SMTP session generated by email middleware reach the victim immediately.

Attacks in the controlled environment. We set up two Linux servers in our laboratory network, designating one server as the attacker and the other as the victim. Due to the limited number of physical machines, we deploy 1,000 instances across 10 Linux servers using Docker to simulate bounce servers. We use Postfix software [56] to implement the email bounce function. The bounce server is configured to generate one bounce email for each distinct recipient in the original email. The amplification of packet size for a bounce communication is 5x. All controlled servers and instances are limited to a network bandwidth of 10 Gb/s. We configure N_{rcpt} to 10 and use

1,000 bounce servers to execute CoordMail. To simulate a traditional bounce backscatter attack, we send original emails to bounce servers at a rate of 10 emails per second. The bounce backscatter attack involves 1,000 bounce servers, with each original email containing 10 different recipients.

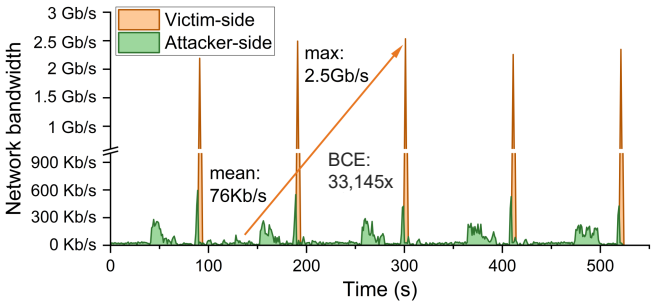

Figure 9. CoordMail experiment using email middleware in the controlled environment (`Nmidd` = 1,000, `Nrcpt` = 10).

As shown in Figure 9, with an average bandwidth of 76 kb/s and a 100-second attack cycle, CoordMail generates an instantaneous bandwidth of 2.5 Gb/s on the victim's side, yielding a BCE of 33,145. Numerous studies have demonstrated that such periodic burst traffic can lead to packet loss and downtime of network services [6], [40]–[42]. In contrast, our experiments show that the BAF of a traditional bounce backscatter attack is only 48, whereas CoordMail can amplify it by several orders of magnitude. Furthermore, CoordMail using hundreds of SMTP connections can achieve a more powerful convergence effect than previous PDoS attacks. Specifically, the *DNSBomb* attack achieves a BCE of about 1K through most public DNS servers [6], and the *CDN-Convex* attack requires 30 minutes of aggregation time to achieve more than 1K BCE [42].

B. Analyzing the Defense Effect of Security Mechanisms

The attack effect of CoordMail depends on whether the attack traffic can be successfully aggregated at the target incoming mail server. If existing security mechanisms can identify and block the attack traffic effectively, the impact of CoordMail is mitigated. We first conducted a systematic review of previous studies and summarized the common protective measures that can be employed to defend against CoordMail, as shown in Table III. Then, we investigate the effectiveness of various security mechanisms in defending against CoordMail. Finally, we evaluate the practical defenses employed by popular email providers to protect against reflected emails. Our results indicate that the DMARC mechanism can effectively block reflected emails generated by CoordMail, while the effectiveness of other security mechanisms is relatively weak. Many popular email providers receive reflected emails from most email middleware.

Email authenticity. The incoming mail server can verify email authenticity through SPF, DKIM, and DMARC, then reject emails that fail verification. Currently, email authenticity verification mechanisms are widely adopted by many

Table III
Survey of common email security mechanisms.

Type	Security mechanism	Defense[1]
Email authenticity	SPF [7]	◐
	DKIM [8]	◐
	DMARC [10]	●
Host reputation	DNSBL [23]	○
	Greylisting [26]	◐
Rate limit	IP sending rate limit [27]	○
	Mailbox receiving rate limit [28]	○

[1] ● means effective against CoordMail; ◐ means partially effective; ○ means ineffective.

domains [57], [58], especially SPF and DKIM. We examine whether the reflected emails from email middleware can pass SPF, DKIM, and DMARC verification. Because the sender domains of reflected emails from open relays can be controlled by attackers, these emails can pass the email authenticity verification. If the bounce server sets the `MAIL FROM` field to empty, we perform SPF inspection using the domain in the `EHLO` field [7].

As shown in Table IV, among the 10,079 bounce servers selected to construct CoordMail, the sender domains of reflected emails from 5,834 (57.9%) bounce servers lack SPF records (*Missing*), and authentication failed for 489 (4.8%) of them (*Fail*). Most bounced emails lack support for DKIM and DMARC mechanisms. Forwarding email providers generally support SPF and DKIM, but the two providers (139.com and sina.com) do not deploy DMARC mechanisms. Furthermore, no email providers include "p=reject" in their domain's DMARC record, which causes incoming mail servers to not reject reflected emails that fail DMARC validation.

Overall, the incoming mail server can only block CoordMail traffic by *forcibly rejecting* reflected emails that fail DMARC validation. Nevertheless, email providers typically only require email to pass SPF or DKIM verification at present [12], [13], and do not mandate DMARC, as its deployment rate in the real world is relatively low [11]. We discover that emails from 7,851 (77.89%) bounce servers could be verified by SPF or DKIM.

Host reputation. The incoming mail server can interrupt connections with email middleware of poor reputation during SMTP communications. We analyze the effectiveness of two types of email blocklists against CoordMail.

• **DNSBL.** Based on measurements from previous work [15], we select the two most popular DNSBLs: *Spamhaus* [9] and *Spamcop* [59]. Because the back node is responsible for sending attack traffic to the victim, we query DNSBLs daily for a month to check the reputation of back nodes. The results show that about 40% of back nodes of open relays are included in DNSBLs, while fewer back nodes from bounce servers and forwarding providers are listed. In particular, only 536 (5.3%) of the back nodes of bounce servers were listed in DNSBLs over the course of a month. Therefore, popular DNSBLs offer limited protection against CoordMail.

Table IV
STATISTICS ON THE DEFENSE OF SECURITY MECHANISMS AGAINST REFLECTED EMAILS FROM SELECTED EMAIL MIDDLEWARE.

Security mechanism	Bounce server	Open relay	Forwarding provider
SPF	10,079	584	6
Missing	5,834 (57.9%)	0	0
Fail	489 (4.8%)	0	0
DKIM	10,079	584	6
Missing	7,652 (75.9%)	0	0
Fail	388 (3.8%)	0	0
DMARC	10,079	584	6
Missing	9,654 (95.8%)	0	2
Fail	112 (1.1%)	0	0
p=reject	42 (0.4%)	0	0
Email blocklist	8,756	367	3,936
Spamhaus	536 (5.3%)	234 (40.1%)	511 (13.0%)
Spamcop	179 (1.8%)	27 (4.6%)	169 (4.3%)

• **Greylisting.** The Greylisting rejects the tuple (sender IP address, sender email address, receiver email address) for the first delivery attempt and later accepts email from the same tuple after a delay (e.g., 5 minutes). The incoming mail server can use Greylisting to block a round of COORDMAIL traffic. However, if the attacker later uses the same tuple to construct COORDMAIL, the Greylisting cannot block it. If the number of back nodes is large and the tuple retention time is short, Greylisting can block a large part of the attack traffic. Specifically, for email middleware with a large number of back nodes, it is difficult for attackers to use the same tuple to reflect emails. As shown in Table II, most bounce servers and email relays have few back-end nodes, with about 80% showing a one-to-one front–back mapping. In contrast, large forwarding providers have many back-end nodes, so Greylisting can only mitigate part of the COORDMAIL traffic that traverses large forwarders. Attackers can further weaken Greylisting through an "attack warm-up" phase, in which middleware is pre-used to send batches of emails to victims. Regarding the tuple retention time, open-source greylisting whitepapers report a tuple cache lifetime of up to 36 days [60], as shorter periods disrupt legitimate email. Overall, Greylisting cannot fully defend against COORDMAIL.

Rate limit. Rate limiting is a common strategy employed by email providers to mitigate DoS attacks. Based on previous research [15], [22], we select 14 popular email providers that we can register for email accounts to investigate. Below, we analyze the defensive effects of two common email rate limits employed by popular providers against COORDMAIL.

• **IP sending rate limit.** The incoming mail server can impose a maximum rate at which an IP address is allowed to send emails. Thus, the attacker must avoid repeatedly connecting to the victim via the same email middleware. However, the core strength of COORDMAIL lies in coordinating different email middleware, making IP sending rate limits ineffective against the attack.

To assess the frequency limit that an attacker can use the same middleware to attack a victim, we measure the IP sending rate limit of popular email providers, which are generally undisclosed. We configure an experimental email server capable of successfully delivering emails to providers. We then send emails to providers at a rate of 60 emails per minute (the limit published by Gmail [27]) for no more than 10 minutes. As shown in Table V, six providers reject emails after approximately one minute due to our server's high sending rate. Therefore, attackers should avoid connecting to a victim using the same middleware more than 60 times per minute.

• **Mailbox receiving rate limit.** The incoming mail server can set the maximum rate at which a mailbox can receive emails. Therefore, attackers need to avoid frequently sending multiple emails to the same mailbox of the victim. Additionally, attackers can collect numerous real email addresses to facilitate the construction of attacks. According to statistics [15], [61], attackers can harvest billions of email addresses from public resources, such as leaked datasets and websites. Therefore, the mailbox receiving rate limit does not significantly impact the effectiveness of COORDMAIL.

The exact size of the receiving rate limit is usually not disclosed. We actively test the mailbox receiving rate limits of popular email providers to assess how frequently an attacker can use the same victim email address to launch COORDMAIL. Specifically, we send emails to our mailboxes at a rate of 60 emails per minute over a period of 10 minutes. Simultaneously, we send one email every 20 seconds from our server to each mailbox to test if it can receive new emails. As shown in Table V, we find that the mailboxes of gmail.com, yahoo.com, icloud.com, proton.me, and naver.com could no longer receive any emails after about one minute. Reviewing online materials [62], we discover that outlook.com and hotmail.com limit the number of emails received per hour to 3,600. In conclusion, attackers should limit the use of the same victim email address to no more than 60 times per minute.

Practical protection. Email providers can integrate various security mechanisms to intercept malicious emails. If reflected emails from middleware fail to pass the email provider's security checks, the attack traffic of COORDMAIL will decrease. We investigate how many email middleware can successfully send emails to 14 popular providers, i.e., reflected emails to inboxes or spam boxes.

We instructed each selected email middleware to send one email to our mailbox, totaling 10,669 emails. As shown in Table V, more than 90% of reflected emails appear in the mailboxes of 11 popular email providers. Most reflected emails are in spam boxes, except for naver.com and cock.li, where about 9K reflected emails reach their inboxes. Overall, the protection systems of the most popular email providers do not directly intercept reflected emails, meaning COORDMAIL can have a practical impact in the real world.

VI. DISCUSSION

This section discusses the mitigation measures against COORDMAIL, and evaluates the mitigation effect through

Table V
STATISTICS ON THE DEFENSE OF 14 POPULAR EMAIL PROVIDERS AGAINST COORDMAIL.

Provider	Sending rate limit[1]	Receiving rate limit[2]	Reflected email[3] Inbox	Spam
gmail.com	●	●	458	4,428
yahoo.com	●	●	2,184	5,807
outlook.com	○	◐	932	8,432
hotmail.com	○	◐	726	7,928
icloud.com	●	●	609	234
qq.com	○	○	196	178
163.com	○	○	2,551	7,245
126.com	○	○	2,715	7,304
139.com	●	○	3,019	6,971
sina.com	●	○	3,921	6,335
yeah.net	○	○	2,081	7,537
proton.me	○	●	4,218	1,914
naver.com	○	●	8,941	682
cock.li	●	○	9,815	104

[1] ● means the IP sending rate cannot exceed 60/s.
[2] ● means the mailbox receiving rate cannot exceed 60/s; ◐ means cannot exceed 3,600/h.
[3] Number of reflected emails from 10,669 selected middleware that appear in the inbox and spam box.

controlled experiments. In addition, we report the results of our responsible disclosure of COORDMAIL.

A. Mitigation

COORDMAIL exploits the inherent properties of the SMTP protocol and the reflection behavior of email middleware, unlike traditional vulnerabilities. Blocking all emails from email middleware would disrupt normal email services. Our idea to mitigate COORDMAIL is to avoid all reflected email converging on the victim in a short period.

Email middleware. We propose a general mitigation solution: add *random delay* in the process of producing reflected email. In particular, the process of producing reflected email is completely controlled by middleware, and the attacker cannot intervene. Moreover, email is a delay-tolerant service and does not have extremely stringent requirements for timeliness. Therefore, it is reasonable to add a random delay to the process of producing the reflected email, which makes it difficult for the attacker to coordinate all reflected emails to reach the victim at the same time.

We design controlled experiments to evaluate the mitigation effect of *random delay*. Based on the experimental environment in Section V-A, we configure bounce servers to introduce a random delay ranging from 0 to 30 seconds before sending reflected emails. As shown in Figure 10, after implementing the random delay, the attack traffic is distributed over time, reaching the victim server more gradually. This results in a reduction of the BCE by approximately 15x. Furthermore, increasing the range of random delays can further weaken the amplification capability of COORDMAIL.

Introducing delays is a common mitigation against time-convergent pulsing DoS [38], and we acknowledge that our random delay scheme affects email service performance to some extent. Based on study [63], the mean end-to-end email delivery latency in real business is 19.37 seconds. Therefore, the 0–30 second random delay we propose would, on average, roughly double the email forwarding delay. Given that email is generally delay-tolerant and the SMTP command timeout recommended by RFC 821 [5] is usually 5 minutes, this increased delay should not substantially degrade user experience.

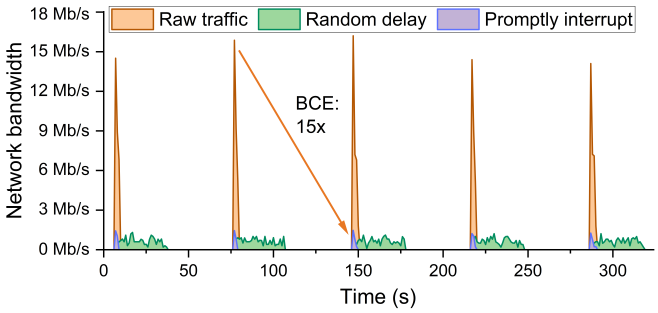

Figure 10. The effect of mitigation measures for COORDMAIL.

For bounce servers, it is crucial to inform the sender as much as possible about the reason for email delivery failure during the SMTP session, rather than relying on bounced emails. Email service administrators should rigorously review server configurations to prevent becoming open relays that could be exploited by attackers. Email forwarding providers should enforce forwarding configuration validation to prevent attackers from redirecting forwarded emails to arbitrary victims. Finally, we recommend that email middleware limit the number of reflected emails for each original email and minimize the size of reflected emails.

Email provider. According to the analysis of Section V-B, most of the reflected emails cannot meet the requirements of the DMARC mechanism. Additionally, many reflected emails display distinct characteristics, such as an empty `MAIL FROM` field or use of specific templates (see Appendeix B). Email providers can *promptly interrupt* reflected emails based on the above traits, especially when dealing with large volumes of traffic. Our experimental evaluation shows that if the incoming mail server disconnects the SMTP session with email middleware after receiving the `RCPT` command, the BCE of attack traffic of COORDMAIL can be reduced by approximately 20 times, as illustrated in Figure 10.

B. Responsible Disclosure

We have responsibly disclosed COORDMAIL and its mitigation measures to most email middleware and 14 popular email providers. Following the vulnerability notification guidelines from previous studies [64], [65], we collected contact emails for email middleware. By sourcing email addresses from WHOIS and domain web pages, we compiled contact information for 9,238 bounce servers, retaining only those email addresses with the same SLD as the email middleware. For the remaining bounce servers, we generated contact information using the four most common usernames found in the contact addresses of 9,238 bounce servers, i.e., “info”, “support”,

"abuse", and "contact". For open email relays, we attempted to reach their administrators through reverse DNS records and AS information of the IP addresses. To minimize disruption to administrators, we limited notification messages to a maximum of five email addresses per email middleware.

So far, we have received replies from 872 email middleware administrators, with 781 automated responses. Among the valid responses, 49 administrators indicated they were awaiting confirmation, 13 mentioned that their email service was managed by hosting providers, and 22 stated they planned to resolve the issue. We further engaged with Coremail, which operates as an email provider for 460 bounce servers. Coremail explained that their cloud mail gateway is only responsible for relaying emails to customer servers and lacks sufficient capacity to determine email deliverability. Consequently, customer servers can only notify delivery errors by sending bounce emails. Coremail has committed to improving the gateway strategy. Among the 14 popular email providers, 8 have acknowledged the threat posed by CoordMail, but most stated that they were not significantly affected. In addition, proton.me plans to improve their mail service implementation.

VII. Ethics Consideration

Our research refers to previous papers related to DoS attacks [6], [42] and adheres to research ethical principles and best network measurement practices [66]–[68]. In constructing and evaluating CoordMail, we carefully consider the following ethical factors.

• **Measure email middleware.** We identify email middleware in the wild by sending emails to non-existent users from our domain, ensuring no emails reach real users' mailboxes. We strictly control the scanning rate to avoid burdening the target network. Our scanning rates were as follows: 3 SMTP connections per second for email service probing, 5 DNS packets per second for MX record probing, and a port-scanning rate of 1M bits/second. The emails we send include experimental explanations and contact information, allowing server administrators to opt out if desired. Importantly, we avoid testing the extreme limits of attack metrics for ethical reasons, focusing only on measuring the relative upper limits. For instance, when testing the value of `Nrcpt` supported by email middleware, we limit the maximum setting to 10. Furthermore, we analyze forwarding relationships of popular email providers on Coremail's internal security server, counting the data only at the domain level without obtaining users' email addresses.

• **Evaluate amplification effect.** To avoid affecting email middleware and cloud server providers, we conducted only small-scale experiments in the real world. Specifically, we used a maximum of 20 email middleware to construct CoordMail, ensuring that the aggregated bandwidth on our controlled servers remained well below the provider's limits. As a complement, we evaluated the amplification effect of CoordMail in a controlled environment, ensuring that no real entities were impacted.

• **Practical considerations of attacks.** Rather than executing DoS attacks, we demonstrate the real threat posed by CoordMail through an analysis of existing security mechanisms. Furthermore, we responsibly reported the risks associated with CoordMail, ensuring relevant entities could benefit from our findings. We also discussed and evaluated mitigation strategies to help the community enhance email security.

• **Reproducibility and artifacts.** Providing code that can be run directly to launch attacks could be exploited to carry out malicious activities. To support ethical security testing and research, we published simulated attack scripts, code to identify email middleware and measure attack metrics, and a sampled dataset of email middleware in the wild at https://github.com/RUI-XUAN-LI/CoordMail.

VIII. Conclusion

This paper proposes CoordMail, a novel email convergence amplification attack that utilizes the SMTP timeout and SMTP command interaction. By coordinating reflected emails from different middleware to reach the victim at the same time, CoordMail can destroy the server's ability to receive emails. To construct CoordMail, we collect email middleware in the real world on a large scale, then analyze their amplification ability and attack usability. Evaluation experiments demonstrate that CoordMail achieves bandwidth concentration efficiency exceeding 30,000 with 1,000 SMTP connections. In addition, we conduct a comprehensive analysis of existing security mechanisms to prove the feasibility and practical damage of CoordMail. Finally, we provide lightweight mitigations against CoordMail, and conduct responsible vulnerability reporting.

Acknowledgment

We thank all anonymous reviewers for their valuable and constructive feedback. This work is supported by the National Key Research and Development Program of China (No. 2023YFB3105600), and the National Natural Science Foundation of China (Grant No. 62102218, 62272413). Baojun Liu and Jun Shao are both corresponding authors.

References

[1] J. Rijn. (2024) Email is not dead. but email is changing. [Online]. Available: https://www.emailisnotdead.com/
[2] M. Schneider, H. Schulmann, A. Sidis, R. Sidis, and M. Waidner, "Diving into email bomb attack," in *DSN*. IEEE, 2020, pp. 286–293.
[3] M. Jakobsson and F. Menczer, "Untraceable email cluster bombs: On agent-based distributed denial of service," *CoRR*, vol. cs.CY/0305042, 2003.
[4] R. Li, C. Lu, B. Liu, Y. Lin, H. Duan, Q. Pan, and J. Shao, "Understanding and characterizing intermediate paths of email delivery: The hidden dependencies," in *IMC*. ACM, 2025.
[5] J. Postel, "Simple mail transfer protocol," RFC 821, August 1982.
[6] X. Li, D. Wu, H. Duan, and Q. Li, "Dnsbomb: A new practical-and-powerful pulsing dos attack exploiting DNS queries-and-responses," in *SP 2024*. IEEE, 2024, pp. 4478–4496.
[7] S. Kitterman, "Sender policy framework (spf) for authorizing use of domains in email, version 1," RFC 7208, April 2014.
[8] D. Crocker, T. Hansen, and M. Kucherawy, "Domainkeys identified mail (dkim) signatures," RFC 6376, September 2011.
[9] Spamhaus, https://www.spamhaus.org/, 2024.

[10] M. Kucherawy and E. Zwicky, "Domain-based message authentication, reporting, and conformance (dmarc)," RFC 7489, March 2015.

[11] M. Ashiq, W. Li, T. Fiebig, and T. Chung, "You've got report: Measurement and security implications of DMARC reporting," in *USENIX*, 2023, pp. 4123–4137.

[12] Gmail, "Email sender guidelines," https://support.google.com/a/answer/81126, 2024.

[13] Yahoo, "Sender requirements and recommendations," https://senders.yahooinc.com/best-practices, 2024.

[14] J. Klensin, "Simple mail transfer protocol," RFC 5321, October 2008.

[15] R. Li, C. Lu, B. Liu, Y. Zhang, G. Hong, H. Duan, Y. Lin, Q. Pan, M. Yang, and J. Shao, "HADES attack: Understanding and evaluating manipulation risks of email blocklists," in *NDSS*. The Internet Society, 2025.

[16] S. Frei, I. Silvestri, and G. Ollmann. (2004) Mail non-delivery notice attacks. https://techzoom.net/whitepapers/mail_non_delivery_notice_attacks_2004.pdf.

[17] wikipedia. (2025) Open mail relay. [Online]. Available: https://en.wikipedia.org/wiki/Open_mail_relay

[18] E. Liu, G. Akiwate, M. Jonker, A. Mirian, G. Ho, G. Voelker, and S. Savage, "Forward pass: On the security implications of email forwarding mechanism and policy," in *EuroS&P*. IEEE, 2023, pp. 373–391.

[19] T. Bass, A. Freyre, D. Gruber, and G. Watt, "E-mail bombs and countermeasures: cyber attacks on availability and brand integrity," *IEEE Netw.*, vol. 12, no. 2, pp. 10–17, 1998.

[20] Proofpoint. (2015) Dead phish bounce: Alerting to brand risk with email backscatter. https://www.proofpoint.com/us/threat-insight/post/Dead-Phish-Bounce.

[21] Bitdefender. (2022) Backscatter spam attack used to deliver bitcoin extortion messages to eastern europe. https://www.bitdefender.com/en-us/blog/hotforsecurity/backscatter-spam-attack-used-to-deliver-bitcoin-extortion-messages-to-eastern-europe/?srsltid=AfmBOopooPLnWnJXgOOFx4thACPN7T5LLGKqP5CwuPy6BjFdLkqzXFn9%2F%2F%2F.

[22] K. Shen, C. Wang, M. Guo, X. Zheng, C., B. Liu, Y. Zhao, S. Hao, H. Duan, Q. Pan, and M. Yang, "Weak links in authentication chains: A large-scale analysis of email sender spoofing attacks," in *USENIX Security*, 2021, pp. 3201–3217.

[23] C. Lewis and M. Sergeant, "Overview of best email dns-based list (dnsbl) operational practices," RFC 6471, January 2012.

[24] "Yahoo smtp error codes," https://senders.yahooinc.com/smtp-error-codes/, 2024.

[25] Microsoft, https://answers.microsoft.com/en-us/outlook_com/forum/all/550-571-service-unavailable-client-host-51xx2xx1x/79a129be-ab20-413b-abf9-659a93c7eec7, 2024.

[26] M. Kucherawy and D. Crocker, "Email greylisting: An applicability statement for smtp," RFC 6647, June 2012.

[27] Gmail, "Unsolicited rate limit error," https://support.google.com/mail/?p=UnsolicitedRateLimitError, 2024.

[28] Google, "Gmail receiving limits in google workspace," https://support.google.com/a/answer/1366776?sjid=14111895751383822700-AP.

[29] Akamai, "Fighting the heat: Emea's rising ddos threats," https://www.akamai.com/resources/state-of-the-internet/2024-emea-ddos-report, 2024.

[30] Cloudflare, "Ddos reports," https://blog.cloudflare.com/tag/ddos-reports/, 2024.

[31] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the mirai botnet," in *USENIX Security*, 2017, pp. 1093–1110.

[32] M. Kührer, T. Hupperich, C. Rossow, and T. Holz, "Exit from hell? reducing the impact of amplification ddos attacks," in *USENIX Security*, 2014, pp. 111–125.

[33] M. Anagnostopoulos, G. Kambourakis, S. Gritzalis, and D. Yau, "Never say never: Authoritative TLD nameserver-powered DNS amplification," in *NOMS*. IEEE, 2018, pp. 1–9.

[34] R. Rijswijk-Deij, A. Sperotto, and A. Pras, "DNSSEC and its potential for ddos attacks: a comprehensive measurement study," in *IMC*. ACM, 2014, pp. 449–460.

[35] J. Czyz, M. G. Kallitsis, M. Gharaibeh, C. Papadopoulos, M. D. Bailey, and M. Karir, "Taming the 800 pound gorilla: The rise and decline of NTP ddos attacks," in *IMC*. ACM, 2014, pp. 435–448.

[36] Y. Afek, A. Bremler-Barr, and L. Shafir, "Nxnsattack: Recursive DNS inefficiencies and vulnerabilities," in *USENIX Security*, 2020, pp. 631–648.

[37] G. Moura, S. Castro, J. Heidemann, and W. Hardaker, "Tsuname: exploiting misconfiguration and vulnerability to ddos DNS," in *IMC*. ACM, 2021, pp. 398–418.

[38] R. Rasti, M. Murthy, N. Weaver, and V. Paxson, "Temporal lensing and its application in pulsing denial-of-service attacks," in *IEEE Symposium on Security and Privacy*, 2015, pp. 187–198.

[39] L. Gu, J. Zhang, and B. Bensaou, "Unleashing the shrew: a stealth greedy targeted attack on TCP traffic in wireless lans," in *LCN*. IEEE Computer Society, 2014, pp. 337–344.

[40] M. Guirguis, A. Bestavros, I. Matta, and Y. Zhang, "Reduction of quality (roq) attacks on dynamic load balancers: Vulnerability assessment and design tradeoffs," in *INFOCOM*. IEEE, 2007, pp. 857–865.

[41] A. Kuzmanovic and E. Knightly, "Low-rate tcp-targeted denial of service attacks: the shrew vs. the mice and elephants," in *SIGCOMM*. ACM, 2003, pp. 75–86.

[42] R. Guo, J. Chen, Y. Wang, K. Mu, B. Liu, X. Li, C. Zhang, H. Duan, and J. Wu, "Temporal cdn-convex lens: A cdn-assisted practical pulsing ddos attack," in *USENIX Security*, 2023, pp. 6185–6202.

[43] W. Xu, X. Li, C. Lu, B. Liu, H. Duan, J. Zhang, J. Chen, and T. Wan, "Tsuking: Coordinating DNS resolvers and queries into potent dos amplifiers," in *CCS*. ACM, 2023, pp. 311–325.

[44] KrebsonSecurity, "Massive email bombs target .gov addresses," https://krebsonsecurity.com/2016/08/massive-email-bombs-target-gov-addresses/, 2016.

[45] E. Liu, G. Akiwate, M. Jonker, A. Mirian, S. Savage, and G. Voelker, "Who's got your mail?: characterizing mail service provider usage," in *IMC*. ACM, 2021, pp. 122–136.

[46] P. Gummadi, S. Saroiu, and S. Gribble, "King: estimating latency between arbitrary internet end hosts," in *SIGCOMM Internet Measurement Workshop, IMW*. ACM, 2002, pp. 5–18.

[47] Tranco, "Top 1m domains," https://tranco-list.eu/, 2024.

[48] C. Umbrella, "Top 1m domains," https://s3-us-west-1.amazonaws.com/umbrella-static/index.html, 2024.

[49] Majestic, "Top 1m domains," https://majestic.com/reports/majestic-million, 2024.

[50] haveibeenpwned, "Largest breaches," https://haveibeenpwned.com/, 2024.

[51] ip api, "Ip geolocation api," https://ip-api.com/, 2023.

[52] Z. Durumeric, E. Wustrow, and J. A. Halderman, "Zmap: Fast internet-wide scanning and its security applications," in *USENIX*, 2013, pp. 605–620.

[53] Coremail, https://www.coremail.cn/.

[54] A. Chand, N. Nikiforakis, and P. Vadrevu, "Doubly dangerous: Evading phishing reporting systems by leveraging email tracking techniques," in *USENIX Security Symposium*, 2025, pp. 3181–3200.

[55] tcpdump, https://www.tcpdump.org/.

[56] postfix. [Online]. Available: https://www.postfix.org/

[57] S. Czybik, M. Horlboge, and K. Rieck, "Lazy gatekeepers: A large-scale study on SPF configuration in the wild," in *IMC*. ACM, 2023, pp. 344–355.

[58] C. Wang, K. Shen, M. Guo, Y. Zhao, M. Zhang, J. Chen, B. Liu, X. Zheng, H. Duan, Y. Lin, and Q. Pan, "A large-scale and longitudinal measurement study of DKIM deployment," in *USENIX Security*, 2022, pp. 1185–1201.

[59] SpamCop, https://www.spamcop.net/.

[60] "The next step in the spam control war: Greylisting," http://projects.puremagic.com/greylisting/whitepaper.html, 2025.

[61] R. Li, Z. Zhang, J. Shao, R. Lu, X. Jia, and G. Wei, "The potential harm of email delivery: Investigating the HTTPS configurations of webmail services," *TDSC*, vol. 21, no. 1, pp. 125–138, 2024.

[62] Microsoft. (2025) Receiving and sending limits. [Online]. Available: https://learn.microsoft.com/en-us/office365/servicedescriptions/exchange-online-service-description/exchange-online-limits#receiving-and-sending-limits

[63] R. Li, S. Xiao, B. Liu, Y. Lin, H. Duan, Q. Pan, J. Chen, J. Zhang, X. Liu, X. Lu, and J. Shao, "Bounce in the wild: A deep dive into email delivery failures from a large email service provider," in *IMC*. ACM, 2024.

[64] F. Li, Z. Durumeric, J. Czyz, M. Karami, M. Bailey, D. McCoy, S. Savage, and V. Paxson, "You've got vulnerability: Exploring effective vulnerability notifications," in *USENIX Security*, 2016, pp. 1033–1050.

[65] B. Stock, G. Pellegrino, F. Li, M. Backes, and C. Rossow, "Didn't you hear me? - towards more successful web vulnerability notifications," in *NDSS*, 2018.

[66] "The belmont report: ethical principles and guidelines for the protection of human subjects of research," United States. National Commission for the Protection of Human Subjects of Biomedical and Behavioral Research. Department of Health, Education and Welfare, 1979.
[67] E. Kenneally and D. Dittrich, "The menlo report: Ethical principles guiding information and communication technology research," 2012.
[68] C. Partridge and M. Allman, "Ethical considerations in network measurement papers," Communications of the ACM, 2016.

APPENDIX

A. Build SMTP Command Sequence

Algorithm 1 describes the process of building an SMTP command sequence. The sending time of *Non-mandatory* SMTP commands does not need to strictly follow the SMTP command sequence and can be sent at any time except during the email content delivery. It should be noted that there are various SMTP command sequences for maintaining SMTP sessions, and Algorithm 1 presents only one of them. However, we have demonstrated that SMTP command sequences can maintain SMTP sessions for more than 10 minutes.

Algorithm 1: Build SMTP command sequence

Input: Target maintenance time T_{targ}; Total timeout of *Necessary* states T_{nece}; Maximum duration of each *Necessary* states D_{nece} = {*TCP*: *T*s, ..., *End*: *E*s}; Maximum duration of each *Temporary* states D_{temp} = {*NOOP*: *N*s, ..., *VRFY*: *V*s}; Maximum number of each *Temporary* states N_{temp} = {*NOOP*: *n*, ..., *VRFY*: *v*}

Output: SMTP command sequence *out_list* = [(*TCP, t1*), ..., (*NOOP, t3*)]

```
out_list ← []
if Tnece ≥ Ttarg then
    // only Mandatory SMTP commands are required
    for key-value pairs (c, t) in Dnece do
        Append (c, Ttarg ×(t/Tnece)) to out_list
    return out_list
else
    // add Mandatory SMTP commands
    for key-value pairs (c, t) in Dnece do
        Append (c, t) to out_list
    Tremain = Ttarg − Tnece
    // supplement Non-mandatory SMTP commands
    for key-value pairs (c, t) in Dtemp do
        for num in Ntemp[c] do
            Tremain ← Tremain − t
            if Tremain ≤ 0 then
                Append (c, t+Tremain) to out_list
                return out_list
            else
                Append (c, t) to out_list
```

B. Identify and Analyze Forwarding Relationships

To reduce the attack cost, the attacker can utilize existing forwarding relationships in the real world to execute CoordMail. We observe that many providers use custom templates to generate the `MAIL FROM` fields for forwarded emails. As shown in Table VI, nine popular providers have unique `MAIL FROM` templates that include forwarding initiators, which provides an opportunity to identify forwarding relationships in the wild. By collaborating with Coremail [53], we extracted all template-matching `MAIL FROM` fields and corresponding `RCPT TO` fields from one year's email reception logs. We then identify the forwarding initiator from the `MAIL FROM` field and the forwarding destination from the `RCPT TO` field. Finally, we deduplicate all forwarding relationship pairs.

In total, we identified 858,785 forwarding relationship pairs, with qq.com accounting for 93.82%. This is primarily because most Coremail users come from China, making qq.com relatively more popular. The attacker can exploit existing forwarding relationships to reflect emails to 11,572 domains. For example, we find that the attacker can forward emails to a business domain via 21,831 email addresses of forwarding providers. Figure 11 illustrates the partial forwarding flow for the top five domains with the highest number of forwarding relationships, including 2 business domains, 2 university domains, and 1 government domain. We acknowledge that analyzing passive data from Coremail reveals only a small fraction of real-world forwarding relationships. However, our results demonstrate the feasibility of utilizing forwarding servers to execute CoordMail.

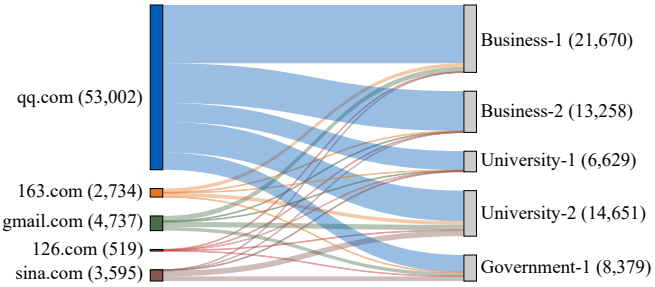

Figure 11. Partial email forwarding relationships of the top five domains with popular providers.

C. Measure SMTP Session State Machine

Table VII presents detailed results for SMTP session timeout times and the number of consecutive *Temporary* SMTP session states for 10 popular email forwarding providers. For each SMTP session state, the maximum timeout time we tested was 300s; for the total SMTP session, the maximum timeout time tested was 10m; and for the number of consecutive *Temporary* SMTP session states, the maximum value tested was 30. Using the SMTP command sequence, we can maintain 10-minute SMTP sessions with all forwarding providers. In particular, the client can refresh the *Temporary* SMTP session

Table VI
STATISTICS OF FORWARDING VERIFICATION, CHARACTERISTICS, RELATIONSHIPS AND BACK NODES OF POPULAR PROVIDERS.

Provider	Verify[1]	MAIL FROM template (ignore case)[2]	Forward relation		Back node
			address	domain	
gmail.com	✔	<e_user>+caf_=<f_user>=<f_domain>@gmail.com	15,208	3,210	1,801
outlook.com	✗	<e_user>+srs=<code>=<o_domain>=<o_user>@outlook.com	1,491	1,173	3,143
hotmail.com	✗	<e_user>+srs=<code>=<o_domain>=<o_user>@hotmail.com	1,952	1,301	2,422
icloud.com	✗	<e_user>@icloud.com	-[3]	-	-
qq.com	✔	<e_user>+auto_=<f_user>=<f_domain>@qq.com	797,303	6,534	897
163.com	✗	auto_<e_user>+<f_user>=<f_domain>@163.com	6,566	1,843	683
126.com	✗	auto_<e_user>+<f_user>=<f_domain>@126.com	2,144	918	155
139.com	✔	rm_mail_<e_user>_auto_forwaed_<f_user>=<f_domain>@139.com	139	12	31
sina.com	✔	<e_user>+==<code>==@sina.com	28,850	1,951	369
yeah.net	✗	auto_<e_user>+<f_user>=<f_domain>@yeah.net	1,116	475	80

[1] ✔ means that the provider verifies ownership of the forwarding destination, ✗ means no verification.
[2] The full path of forwarding the email is: the originator (<o_user>@<o_domain>) to email provider (<e_user>@<e_domain>), and then to forwarding destination (<f_user>@<f_domain>). <code>indicates a coded text customized by the email provider.
[3] The `MAIL FROM` field of the forward email from icloud.com has no unique characteristics, so we cannot discover the forward relationship and back nodes associated with it.

state counter of the email server by sending *Mandatory* commands before reaching the upper limit of the *Temporary* SMTP session state count. This allows the client to send many *Non-mandatory* SMTP commands within a single SMTP session.

D. Theoretical Attack Effect of CoordMail

The bandwidth amplification effect of CoordMail is closely related to The bandwidth concentration efficiency of CoordMail is closely related to the number of email middleware (N_{midd}), SMTP connection aggregation time (T_{atta}), and the number of recipients of the original email (N_{rcpt}). Table VIII shows the theoretical BCE for CoordMail. The average bandwidth of only tens of kb/s on the attacker's side can generate a bandwidth of up to Gb/s level on the victim's side, resulting in a BCE that reaches tens of thousands of times. For example, CoordMail can achieve more than 10K theoretical BCE using 1K SMTP connections. Compared to previous PDoS attacks, the theoretical BCE achieved by *DNSBOMB* attack using 1K DNS queries is 80 to 40K, depending on the DNS software [6].

Table VII
STATISTICS ON SMTP SESSION TIMEOUTS AND TEMPORARY SMTP SESSION STATE NUMBER SUPPORTED BY EMAIL PROVIDERS. BECAUSE THE EXPERIMENT SPECIFIES THE MAXIMUM TEST VALUE, SOME RESULTS CANNOT REPRESENT THE ABSOLUTE UPPER LIMIT.

Provider	gmail.com	outlook.com	hotmail.com	icloud.com	qq.com	163.com	126.com	139.com	sina.com	yeah.net
Maximum timeout for Necessary states (second)										
TCP	300	300	300	300	60	60	60	60	30	10
EHLO	300	300	300	300	60	60	60	60	30	10
MAIL	300	300	300	300	60	60	60	60	30	10
RCPT	300	300	300	300	60	60	60	60	30	10
DATA	300	300	300	300	60	60	60	60	30	10
Content	300	300	300	300	60	60	60	60	5	10
End	300	300	300	300	60	60	60	60	5	10
Maximum timeout for Temporary states (second)										
NOOP	300	300	300	300	60	60	30	60	30	10
VRFY	300	300	300	300	60	30	60	0	30	10
HELP	300	300	300	300	60	30	60	0	30	10
TURN	300	300	300	300	60	30	60	0	30	10
XADR	300	300	300	300	60	60	60	0	30	10
ABCD	300	300	300	300	60	60	60	0	30	10
Maximum number of consecutive times for Temporary states										
NOOP	30	30	30	30	30	30	30	30	30	30
VRFY	10	5	5	30	30	4	4	0	30	4
HELP	30	30	30	30	30	4	4	0	30	4
TURN	10	5	5	30	30	4	4	0	30	4
XADR	10	5	5	30	30	4	4	0	30	4
ABCD	10	5	5	30	30	4	4	0	30	4
Maximum timeout for total SMTP session (minute)										
Command sequence	10	10	10	10	10	10	10	10	10	10

Table VIII
THEORETICAL BANDWIDTH CONCENTRATION EFFICIENCY (BCE) OF COORDMAIL.

N_{rcpt}	**Attacker-side**	**Victim-side**	**BCE**
N_{midd}=100 & T_{atta}=60s			
1	12.1Kb/s	21.9Mb/s	1,816x
5	13.2kb/s	68.8Mb/s	5,219x
20	17.2Kb/s	237.1Mb/s	13,805x
N_{midd} = 1000 & T_{atta} = 300s			
1	24.2Kb/s	128.6Mb/s	5,312x
5	26.3Kb/s	392.9Mb/s	11,4912x
20	34.3Kb/s	1.4Gb/s	40,837x
N_{midd} = 2000 & T_{atta} = 600s			
1	24.2Kb/s	217.8Mb/s	8,996x
5	26.3Kb/s	646.4Mb/s	24,536x
20	34.3Kb/s	2.3Gb/s	66,867x