Automatic Insecurity: Exploring Email Auto-configuration in the Wild

Shushang Wen*, Yiming Zhang†✉, Yuxiang Shen*, Bingyu Li‡, Haixin Duan†§, Jingqiang Lin*✉
*School of Cyber Science and Technology, University of Science and Technology of China, China
†Tsinghua University, China
‡School of Cyber Science and Technology, Beihang University, China
§Zhongguancun Laboratory, China
{sswen, yuxiangshen}@mail.ustc.edu.cn, {zhangyiming, duanhx}@tsinghua.edu.cn,
libingyu@buaa.edu.cn, linjq@ustc.edu.cn

***Abstract*—Email clients that support auto-configuration mechanisms automatically retrieve server configuration information, such as the hostname, port number, and connection type, allowing users to log in by simply entering email addresses and passwords. Auto-configuration mechanisms are being increasingly adopted. However, the security implications of these mechanisms, both in terms of implementation and deployment, have not yet been thoroughly studied. In this paper, we present the first systematic analysis of security threats associated with email auto-configuration and evaluate their impacts. We summarize 10 attack scenarios, covering 17 defects (including 8 newly identified ones), along with 4 inadequate client UI notifications. These attack scenarios can either cause a victim to connect to an attacker-controlled server or establish an insecure connection, putting the victim's credentials at risk. Moreover, our large-scale measurements and in-depth analysis revealed serious insecurity of auto-configuration applications in the wild. On the server-side, we discovered 49,013 domains, including 19 of the Top-1K popular domains, were misconfigured. On the client-side, 22 out of 29 clients were vulnerable to those threats. Moreover, 27 out of 29 clients exhibited at least one UI-notification defect that facilitates silent attacks. These defects arise from misconfiguration, mismanagement, flawed implementation and compatibility. We hope this paper raises attention to email auto-configuration security.**

I. Introduction

Email has emerged as a crucial communication channel globally [16]. Email clients are widely used for their customization options and convenience, with 57.8% of all emails opened through mobile or desktop clients in 2021 [36]. However, email clients typically require users to specify the configuration information of the mail server (e.g., hostname, port number, connection type) and establish a connection with the target server based on the above parameters. This complicates the login process and impairs usability.

To improve the usability of email clients, Microsoft and Thunderbird proposed their auto-configuration mechanisms, *Autodiscover* [46] and *Autoconfig* [51] in 2007 and 2008, respectively. Both mechanisms are designed to retrieve server configuration information automatically, allowing a user to log in by simply entering the email address and password. Subsequently, in 2011, the IETF released a standard [23] for client auto-configuration, defining the use of DNS SRV records to locate email submission and access services. In a typical login scenario that supports auto-configuration, server administrators publish configuration information on web servers. The client requests this configuration information by constructing specific URLs using the user's email address. Then, it establishes a connection with the mail server based on the server hostname, port, and connection type parameters in the configuration information.

Although auto-configuration mechanisms significantly improve the usage convenience of email clients, they also introduce new attack vectors for the email system. Recent studies [56], [8] have identified flawed Autodiscover implementations, which can cause email users to inadvertently connect to attacker-controlled servers, thereby exposing their credentials. To the best of our knowledge, no research has been done to systematically analyze the security of email auto-configuration mechanisms and evaluate their real-world impacts. While some threats have been discussed in the non-academic community [23], [14], they focus mainly on protocol design and lack practical evaluation. Furthermore, auto-configuration remains an area without harmonized standards. Microsoft's Autodiscover and Thunderbird's Autoconfig serve merely as industry references. Email vendors may implement their own defined or customized mechanisms, such as built-in lists, heuristic guessing, and default settings. Those mechanisms remain unclear to the public, and the security implications are also understudied.

Question. Our investigation is guided by two questions: *What security threats exist in email auto-configuration? If defects are present, how extensive is their impact on email services?* We focus our exploration on the standard email protocols (i.e., IMAP, POP3, and SMTP), with an emphasis on potential threats that could facilitate user credentials theft. In this paper, we define the *web server* as the entity responsible for publishing the configuration file to the client, and the *mail server* as the server to which the client is expected to log in. In a *client-server* communication scenario with auto-configuration, we identified two key security factors: (1) the transmission of configuration information from the web

✉Corresponding author.

Network and Distributed System Security (NDSS) Symposium 2025
24-28 February 2025, San Diego, CA, USA
ISBN 979-8-9894372-8-3
https://dx.doi.org/10.14722/ndss.2025.241078
www.ndss-symposium.org

server to the client should be secured, and (2) the setting of configuration parameters should enforce a secure connection (e.g., STARTTLS or implicit TLS) between the client and the mail server. These factors correspond to two types of attacks presented in our threat analysis.

Approach. To comprehensively assess security issues of email auto-configuration, we analyzed potential threats and identified defects in both servers and clients. Then, we conducted a large-scale inspection of servers across 1,053,469 domains and 29 email clients on 5 operating platforms to evaluate the real-world impact of these threats.

Findings. The auto-configuration process consists of two phases: (1) *configuration information retrieval*, and (2) *parameter parsing and application*. We analyze two threats (or attack goals) in these two phases: victims connecting to attacker-controlled servers (**Type-I**) and leaking credentials (**Type-II**). Based on different attacker capabilities, we summarize 10 attack scenarios on these goals, with 8 newly identified defects.

Our extensive experiments demonstrated that current email auto-configurations in the wild exhibit general defects on both the server and client sides. Among the 1M domains tested, 79,212 supported at least one auto-configuration mechanism. Of these domains, 61.88% (49,013) domains were misconfigured. 55.0% (43,566) had defects leading to Type-I threats, such as delivering configuration files via plaintext HTTP connections, allowing attackers to tamper with configurations and redirect the email connections to servers they controlled. Meanwhile, 14.93% (11,824) of domains were affected by Type-II threats, including defects of configuration (e.g., incorrect parameter settings) and management (e.g., inconsistent parameters among mechanisms), which downgrade the security of connections to mail servers. 19 popular domains in the Top-1K list, including well-known vendors like Yandex and Onet, suffered from these issues.

In the client experiments, we tested 29 clients across 5 operating platforms. Of these, 22 clients, including Thunderbird and Outlook, are affected by at least one threat. Specifically, 13 clients were susceptible to Type-I threats, potentially connecting users to attacker-controlled servers. 19 clients were vulnerable to Type-II threats, resulting in connection downgrades. Further, we examined client UI designs and found that 21 clients did not seek user confirmations when obtaining configurations, enabling silent attacks. We also identified client-side defects that could leak user credentials on a server without auto-configuration mechanisms. For instance, Nextcloud Mail did not verify domain formats when constructing configuration request URLs, mistakenly connecting to domains that could be registered by attackers. This defect could result in users from at least 24,149 domains being hijacked to attacker-controlled servers. We have extensively reported to affected clients and servers, and some of them have confirmed and fixed related issues (see Section VIII-C for details).

Contributions. Our main contributions include:

- We systematically analyzed email auto-configuration defects across protocol design, server deployment, and client implementation. We summarized 10 attack scenarios with 8 new defects and 4 UI issues that resulted in connecting to attacker-controlled servers or leaking credentials.
- We conducted extensive measurements of auto-configuration to evaluate the real-world impact of these threats, revealing widespread flaws in server deployments and client implementation, and discussing the root causes of the flaws.

Table I: Protocols and ports for email services.

Protocol	Port	Defined in	Service	Implicit TLS
SMTP	25	[63]	Relay and submission	No
	587	[29]	Submission	No
	465	[50]		Yes
IMAP	143	[19]	Access	No
	993	[50]		Yes
POP3	110	[55]	Access	No
	995	[50]		Yes

II. Background

A. Email Submission and Access

An email system consists of several components [22] that collaborate to send and receive email messages. *Email submission* [30] and *access* refer to the interactions between users and servers, while *email relay* involves the store-and-forward transmissions of messages between mail servers.

Simple Mail Transport Protocol (SMTP) [39] plays a vital role in transmitting messages across the Internet. Email submission is a communication between a Mail User Agent (MUA), which is also known as an email client (e.g., Thunderbird), and a Mail Submission Agent (MSA). The MSA is responsible for posting a message to the outgoing server, which then delivers the message to the intended recipient through email relay. Although both submission and relay use SMTP, the standard [29] specifies different ports for each service. Internet Message Access Protocol (IMAP) [19] and Post Office Protocol v3 (POP3) [55] are email protocols for accessing messages on servers.

B. STARTTLS and Implicit TLS

Email protocols like SMTP, IMAP, and POP3 were originally designed without encryption, leaving data potentially exposed. STARTTLS [33] was proposed to compensate for this design weakness, which allows upgrading insecure connections to secure ones on existing ports (i.e., 587, 143, and 110). Implicit TLS, however, mandates encryption from the beginning of the connection without requiring an explicit command to initiate it. As it relies on dedicated ports (i.e., 465, 993, and 995), it ensures that all data transmitted over the connection is encrypted by default, reducing the risk of transmitting data in plaintext and eliminating attacks against STARTTLS [62]. Table I lists well-known ports of SMTP, IMAP, and POP3.

Several security issues with STARTTLS [33], [57], [78], [40], [62] have been disclosed. The command injection vulnerability (CVE-2011-0411) [78], which was first discovered in 2011, allows attackers to inject plaintext content into the TCP packet containing the STARTTLS command, leading the server to misinterpret it as part of the TLS session. The primary cause of this vulnerability is that servers mishandle the state transition from unencrypted to encrypted communications, making plaintext commands buffered alongside the STARTTLS negotiation. The complexity of STARTTLS makes it error-prone to implement, and implicit TLS is recommended to be prioritized over STARTTLS for secure connections [50].

C. Related Work

Mismanagement of email protocol deployments. Measurement studies on the deployment of email protocols have been conducted, including sender authentication (e.g., SPF, DKIM, and DMARC) [28], [26], [11], [80], [10], [75], transport encryption (e.g., TLS, DANE, and MTA-STS) [28], [26], [34], [45], [43], [42], [74], [13] and end-to-end encryption (e.g., S/MIME and OpenPGP) [73]. A significant proportion of SPF policies were found to be overly broad [26], weakening their intended protections, while the use of the `include` mechanism in SPF records has brought excessive DNS lookups during SPF evaluation [11]. Misconfigurations have also been found in the DKIM deployment, such as weak keys and signatures [80]. Recently, researchers analyzed the DMARC reporting mechanism and revealed that 26% of DMARC records with external domains lacked proper authorizations, making them vulnerable to reflection attacks [10]. On transport encryption protocols, researchers analyzed the usage of TLS in email ecosystems for the entire IPv4 address space. They discovered that a large percentage of emails were transmitted unencrypted, leaving content vulnerable to interception [34], [45]. Finally, a study on email encryption adoption at a university revealed that only 0.06% of over 80M emails were encrypted, and 32.99% of PGP keys lacked expiration dates [73]. Our work focuses on the deployment of email auto-configuration mechanisms, identifying security issues in configuration and management.

Attacks on email mechanisms. Recent work has exposed various exploitable vulnerabilities in email security protocols, including authentication bypass [44], [70], [17], [35], credential theft [62], signature spoofing [52] and even the decryption of encrypted contents [38], [61], [53]. For example, the difficulties in detecting and mitigating spoofing attacks, make forged emails difficult to handle [35]. Additionally, weaknesses in S/MIME and OpenPGP email signature verification leave 70% of the tested clients susceptible to forgery attacks [52]. A novel attack technique called malleability gadgets [61] was introduced to exploit CBC and CFB modes in S/MIME and OpenPGP encryption, enabling the exfiltration of plaintext through backchannels, and such vulnerabilities were found in 23 S/MIME and 10 OpenPGP clients.

Poddebniak et al. [62] performed the first structured analysis of STARTTLS and uncovered over 40 issues that could be exploited to steal users' login credentials. They developed a test tool and identified over 300,000 hosts vulnerable to the command injection. They concluded that STARTTLS should be replaced with implicit TLS. Our work applied this conclusion to the threat analysis of email auto-configuration.

III. Email Auto-configuration

Email auto-configuration automates the configuration process in email clients that retrieves server settings, thus simplifying email account setup for users. Figure 1 shows the typical workflow of email auto-configuration. 1) Email administrators first publish server configuration information by

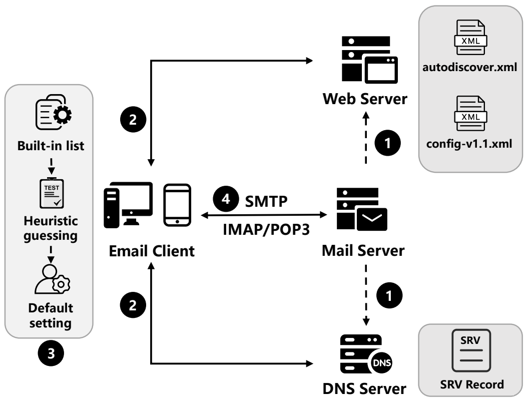


Figure 1: Email auto-configuration.

placing configuration files at specified URLs or by adding SRV [32] records to DNS. 2) When a user enters an email address, the client that supports auto-configuration generates a list of candidate URLs based on the email address and attempts to retrieve server configurations from these URLs. The client can also query SRV records to obtain these settings. 3) If configurations are unavailable, the client resorts to a built-in list containing the settings of popular providers or heuristically guesses the settings. If this fails, the client usually fills in some default parameters (e.g., the connection type) and prompts the user to enter the hostname manually. 4) After the user confirms the settings, the client uses them to log in to the mail server.[1]

A. Autodiscover

Autodiscover was first introduced in Exchange for Outlook [46] and later extended to support others. This paper focuses on Autodiscover for standard email protocols [47], [48]. According to the specification [48], Autodiscover requires a client to send an HTTP POST request with the email address for which the configuration information will be retrieved. The Autodiscover response contains configuration information related to the mail server, including settings for IMAP, POP3, and SMTP services. For example, the connection type parameter (i.e., `SSL` or `Encryption` element) specifies whether encryption is required to connect to servers (more details in Appendix A).

A client that supports Autodiscover performs as below: it firstly constructs a list of Autodiscover URLs based on the email address entered by a user [47]; then, the client sequentially requests each URL until it successfully obtains the configuration information. Table II shows an example and the user's email address is *user@example.com*, patterned as <local part>@<domain part> [20]. The client initially extracts the domain part (marked in red) and constructs an ordered list of candidate URLs. The client then sends POST requests to the URLs listed in Steps 1 or 2 to retrieve the configuration information. If the previous requests fail, in Step 3 the client queries the SRV record to obtain the destination hostname (e.g., *target.com*) of the Autodiscover server. The client then constructs a new URL using that hostname (marked in blue) and sends a POST request. Finally, in Step 4, if all preceding attempts fail, the client sends an insecure (non-SSL) GET request to the URL from Step 2. If the server responds with a

[1]In this paper, the discussion of mail servers is limited only to those involved in the processes of email submission and access, excluding email relay.

Table II: An example of configuration information retrieval via Autodiscover.

Step	Candidate URL	Request method
1	http://example.com/autodiscover/autodiscover.xml	HTTP POST
2	https://autodiscover.example.com/autodiscover/autodiscover.xml	HTTP POST
3.1	_autodiscover._tcp.example.com. IN SRV 0 0 443 target.com.	DNS SRV request for Autodiscover server
3.2	https://target.com/autodiscover/autodiscover.xml	HTTP POST
4	http://autodiscover.example.com/autodiscover/autodiscover.xml	HTTP GET for initial request, POST for redirection

Table III: An example of configuration information retrieval via Autoconfig.

Step	Candidate URL	Request method
1	https://autoconfig.example.com/mail/config-v1.1.xml?emailaddress=user@example.com	HTTP GET
2	https://example.com/.well-known/autoconfig/mail/config-v1.1.xml	HTTP GET
3	http://autoconfig.example.com/mail/config-v1.1.xml	HTTP GET
*4	https://autoconfig.thunderbird.net/v1.1/example.com	HTTP GET
5.1	example.com. IN MX 0 mx.backoff.target.com.	DNS MX request for mail provider
5.2	https://autoconfig.backoff.target.com/mail/config-v1.1.xml?emailaddress=user@example.com	HTTP GET
5.3	https://autoconfig.target.com/mail/config-v1.1.xml?emailaddress=user@example.com	HTTP GET
*5.4	https://autoconfig.thunderbird.net/v1.1/backoff.target.com	HTTP GET
*5.5	https://autoconfig.thunderbird.net/v1.1/target.com	HTTP GET
6	%USER_CONFIGURATION_DIR%/isp/example.com.xml	Local import

* Retrieve configuration information through the public centralized database ISPDB [3], which maintained by Thunderbird.

302 redirect, the client then attempts to resend an HTTP POST request to the URL specified in the Location header of the response. The specification [47] suggests that this step should only be used for redirection and not for querying settings.

B. *Autoconfig*

Although Autoconfig [15] has been proposed for over a decade, formal specifications are still not finished. We refer to the only known authoritative source, the draft "Mail Autoconfig" [14], to explain its workflow. Unlike Autodiscover, Autoconfig allows the client to use the email address directly as a query parameter in an HTTP GET request. The Autoconfig response also contains a `socketType` element that specifies whether encryption is required (see Appendix A for details on the Autoconfig response elements).

Table III shows an example of Autoconfig workflow, with each step ordered by priority. In Steps 1-3, the client extracts the domain part (i.e., *example.com*) of the email address and constructs an order list of candidate query URLs. The client then sends HTTP GET requests for each URL in the list. To allow mail providers to provide user-specific configurations, the full email address may be included as a query parameter (e.g., the URL in Step 1). In Step 4, the client accesses a public centralized database ISPDB [3], to retrieve configuration information for most mail providers.

If all of the above steps fail, Autoconfig provides a fallback mechanism. In Step 5, the client reconstructs requests based on the MX hostname (marked in blue) of the email domain. Since MX hostnames may have multiple levels of subdomains, Autoconfig provides two mechanisms for extracting the input to construct the candidate URLs: 1) the parent domain of the MX hostname (i.e., `%MXFULLDOMAIN%`), 2) the effective second-level domain (i.e., eTLD + 1) of the MX hostname (i.e., `%MXMAINDOMAIN%`). For example, the `%MXFULLDOMAIN%` of *mx.backoff.target.com* is *backoff.target.com* and the `%MXMAINDOMAIN%` is *target.com*. Here, we refer to [8] and define this query process as a "back-off" query. Finally, in Step 6, when the client cannot retrieve the configuration through the above requests, it can read a specific directory on the local disk to check for the presence of a configuration file.

Note that some of the candidate steps (i.e., Step 4 and Steps 5.4-5.5) actually retrieve the configuration information from the ISPDB. In this paper, we consider the ISPDB to be a kind of "built-in provider list" (another auto-configuration mechanism discussed in Section III-D). Therefore, we excluded these steps in the subsequent analysis related to Autoconfig, and analyzed the configuration information of the ISPDB separately in Section VII (A8.1).

Table IV: SRV labels for email submission and access.

Service label	Port	Alias	Encryption support
_submission._tcp	587	SUBMISSION	Plaintext or STARTTLS
_imap._tcp	143	IMAP	
_pop3._tcp	110	POP3	
_submissions._tcp	465	SUBMISSIONS	implicit TLS
_imaps._tcp	993	IMAPS	
_pop3s._tcp	995	POP3S	

C. *SRV Record for Locating Services*

DNS SRV resource records (RR) [32] are widely used to locate servers for specific services. RFC 6186 [23] and RFC 8314 [50] define the use of SRV records for locating email submission and access services without the "auto-discover" process (e.g., Autodiscover or Autoconfig) as described above. Table IV lists all the SRV service labels for email submission and access. Since a domain may have multiple mail services, the query name (QNAME) of an SRV record is a combination of the service (e.g., IMAP, POP3, or SUBMISSION), protocol (i.e., TCP or UDP), and domain name. For example, to request an SRV record for an IMAP server of *example.com*, QNAME is formatted as `_imap._tcp.example.com`.

The SRV response contains one or more SRV RRs, each of which consists of the following fields (details in [32]): `Priority`, `Weight`, `Port`, and `Target`. The `Priority` and `Weight` fields determine the order of preference among the listed servers. Servers with lower `Priority` values are more preferred, while among servers with the same `Priority`, those with higher `Weight` values are preferred.

D. Built-in Provider Lists

Clients may contain a built-in list of configuration information for popular providers. The sources of these settings include actively searching and discovering from the Internet [9], email development frameworks [6], or central databases maintained by third parties (e.g., ISPDB [3]). When a client cannot retrieve settings for a domain through real-time queries, it refers to built-in lists to retrieve settings.

E. Heuristic Guessing and Default Settings

Clients may use heuristic methods to guess the mail server settings, typically by prefixing the domain name with a relevant protocol (e.g., "smtp", "imap" or "pop3") and attempting to connect to the constructed hostname via common ports (as listed in Table I). Clients that only assist users in filling out hostnames on login forms, without connection attempts (e.g., Gmail and iOS Mail), are excluded from this definition.

If none of the above mechanisms return settings, a user has to manually enter the hostname and other configuration parameters. In such cases, most clients preset the default value for connection type (e.g., STARTTLS) and authentication method (e.g., password-cleartext) to minimize the user's input.

IV. Attack Analysis around Email Auto-configuration

In this section, we analyze the security of communications between clients and servers that support auto-configuration from two perspectives: (1) *Is the configuration information transmitted securely?* For example, through an encrypted HTTPS connection. (2) *Does the server-provided configuration instruct the client to establish secure connections with a mail server?* For example, setting the connection type to an encrypted option. For the former, attackers could potentially tamper with the configuration information; for the latter, they could sniff the connections to mail servers.

Starting from these two aspects, we analyzed the detailed steps of each email auto-configuration mechanism, including (1) configuration published by a web server, (2) configuration retrieved and parsed by a client, and (3) configuration applied to clients. We analyzed potential threats at each step based on various possible attacker capabilities and outlined each attack scenario. Then, for every auto-configuration mechanism, an attack scenario is instantiated into specific attack cases. We also conducted proof-of-concept experiments for some attack scenarios, demonstrating practical attack cases.

A. Threat Model

We assumed an attacker with two goals: (**Type-I**) *Induce a victim user to connect to attacker-controlled servers, enabling the manipulation of the victim's mailbox*, and/or (**Type-II**) *Steal the victim's credentials, such as passwords*. We explain these attackers' capabilities as below:

Type-I Attacker. Such attackers target the auto-configuration process, where various request methods are used to retrieve configuration information that determines which mail servers a client connects to. A Type-I attacker requires one of the following capabilities, depending on the client's request method: (1) Tampering with TCP packets, on-path attackers (e.g., those sharing a WiFi network) can modify TCP packets to alter configuration information transmitted in plaintext; (2) Domain squatting, attackers can register and control domains that are used in the auto-configuration process. These domains often appear unrelated to mail services, making them overlooked by administrators.

Type-II Attacker. Type-II attackers focus on the connection between a client and the mail servers after configuration information has been retrieved from the web server. Depending on whether the connection is plaintext or encrypted, a Type-II attacker requires one of the following capabilities: (3) Sniffing, on-path attackers can steal credentials by sniffing traffic; (4) Delaying or dropping packets, attackers can disrupt connections by dropping packets; (5) Hacking STARTTLS, an attacker who can tamper with TCP packets as in Capability (1), can inject plaintext contents into TCP packets sent to vulnerable servers (e.g., CVE-2011-0411), allowing them to steal credentials. Details about hacking STARTTLS are discussed in Section II-B.

B. Attack Scenarios and Cases

Building on the above threat analysis and attacker capabilities, we summarize 10 possible attack scenarios in Table V, each containing one or more specific attack cases. An *attack scenario* refers to a category of cases that follow the same attack pattern, while a *case* (denoted as A$i.j$) is a specific attack vector that may arise from configuration or management defects on servers or implementation defects on clients. In each attack scenario, either the client or the server has a defect, and sometimes both are necessary, as in A4. We also indicate which mechanisms are vulnerable to these attack scenarios. For scenarios related to the Type-II attacks, we defined two levels of *downgrade*: (1) the server configuration information allows encrypted-only (i.e., implicit TLS) or STARTTLS connection type, but the client uses plaintext; (2) the server configuration information includes encrypted-only connection type, but the client uses STARTTLS. Depending on the result of the downgrade, the capabilities required for an attacker vary. In the first case, the attacker only needs to sniff packets. In the second, the attacker must hack STARTTLS.

While some attack scenarios have been discussed from a protocol design perspective in the non-academic community (e.g., A1 in the Autoconfig draft [14]), our analysis also considers attacks caused by client implementation and server deployment. Overall, we identified 7 new cases (involving 8 defects, marked with ⋇ in Table V) that have not been discussed before. We categorize these attack scenarios into four groups.

1) Broken external connection for configuration information retrieval: Autodiscover and Autoconfig request configuration files using specific URL formats. The security of these

Table V: Attack scenarios of email auto-configuration.

Attack goal	Attack scenario	Attacker capability	Attack case[3]	Applicability[4]	Client defect[5]	Server defect[5]	
						Web	DNS
Type-I: Connecting to attacker-controlled servers	A1: Client requests configuration information in plaintext	Tampering with TCP packets	A1.1	AC	Plain request	Plain response[6]	∅
			A1.2	AD	Plain request	Plain response[6]	∅
			⁎A1.3	AC/AD	∅	Redirection to HTTP	∅
	A2: Client does not enforce eTLD verification	Domain squatting	⁎A2.1	AC	Inadequate eTLD check	∅	∅
Type-II: Leaking credentials	A3: Server sets only the plaintext connection type	Sniffing	A3.1	AC/AD	∅	Plain-only connection	∅
		Sniffing or hacking STARTTLS[1]	A3.2	SR	∅	∅	Plain or STARTTLS connection
	A4: Client fails to parse and defaults to plaintext	Sniffing	⁎A4.1	AC/AD	Plain fallback on parser error	Incorrect connection type	∅
	A5: Client fails to auto-configure and defaults to plaintext	Sniffing	A5.1	AC/AD/SR/BL	Plain default	∅	∅
	A6: Client implements Autodiscover inadequately	Sniffing or hacking STARTTLS[2]	⁎A6.1	AD	Ignoring Encryption element	∅	∅
	A7: Client prioritizes SRV records incorrectly		⁎A7.1	SR	Non-compliant SRV sorting	∅	∅
	A8: Client maintains an outdated built-in list.		⁎A8.1	BL	Outdated built-in list	∅	∅
	A9: Server prefers insecure connection type		A9.1	AC/AD/SR/BL	∅	∅	Insecure SRV priority
			A9.2			Insecure connection priority	∅
	A10: Server sets inconsistent connection type	Delaying or dropping packets and, sniffing or hacking STARTTLS[2]	⁎A10.1		∅	Inconsistent connection types	

[1] When the server adds only SRV records for non-encrypted-only services, the capability required for an attacker depends on client implementations.
[2] When a downgrade occurs, the capability required for an attacker depends on the downgrade result.
[3] ⁎ Newly identified cases.
[4] AC - Autoconfig. AD - Autodiscover. SR - SRV service discovery. BL - Built-in lists.
[5] ∅ means no defect with the client implementation or server deployment in the corresponding scenario.
[6] In A1.1 and A1.2, only client defect is required. Whether or not the server returns a plain response is irrelevant since a MITM attacker can manipulate the response.

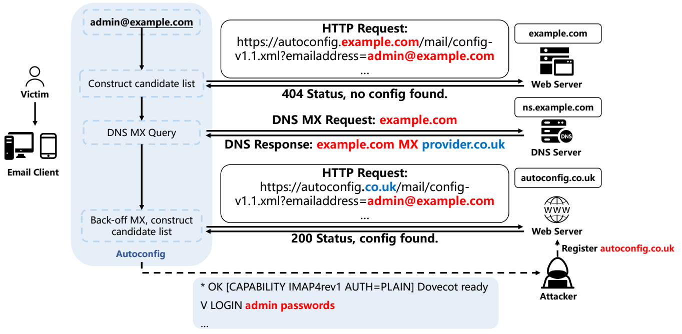
connections depends on the use of TLS protocol [66], [65]. Any data transmitted over plaintext HTTP is vulnerable to modification by manipulator-in-the-middle (MITM) attackers and should be avoided. Additionally, clients must verify that the query URLs are properly formatted; otherwise, they may connect to incorrect or irrelevant domains. Security threats related to DNS resolution (e.g., cache poisoning or hijacking) are not considered, as they pertain to the DNS system itself and are not specific to email auto-configuration (see Section VIII-D for more discussions).

A1: Client requests configuration information in plaintext. When the configuration file is transmitted in plaintext, an attacker can modify the `Server` or `hostname` element to a fake server, causing the victim to send credentials directly to the attacker. Based on the request paths in Tables II and III, we consider whether the client sends plain requests and whether the server transmits configuration information over an unencrypted connection, and identify two distinct cases for Autoconfig (A1.1) and Autodiscover (A1.2). We also found a third case where the server redirects HTTPS client requests to HTTP (A1.3). In A1.1 and A1.2, as long as the client-side defect exists (i.e., sending HTTP requests), a MITM attacker can tamper with the configuration regardless of the server's responses. Nevertheless, since the protocol (e.g., Autodiscover [47], [48]) requires servers to respond over HTTPS even to plaintext requests, we also treat plaintext responses from the server as a server-side defect (related but unnecessary) for this attack.

A2: Client does not enforce eTLD verification. Autoconfig allows clients to request configuration files from the mail provider. As detailed in Step 5 of Table III, a client generates multiple candidate URLs based on the MX hostname. If the client fails to verify whether the extracted `%MXFULLDOMAIN%` or `%MXMAINDOMAIN%` is an eTLD, it may mistakenly use the TLD as the domain name to construct the URLs. This allows attackers to register domains like *autoconfig.tld* in bulk to gain control of these domains. As shown in Figure 2, for an email address *admin@example.com*, the following conditions are required for this attack:

- *example.com* has not deployed Autoconfig.
- The MX hostname for *example.com* is an *eTLD + 1* (e.g., *provider.co.uk*).
- The attacker (can and) has registered *autoconfig.co.uk*.

When the user enters their email address, the client first extracts *example.com* and constructs the query URL based on Steps 1-4 in Table III. If the query fails, the client queries the MX record for *example.com* and extracts the parent domain of the MX hostname to construct a new URL (detailed in Section III-B). At this point, if the client does not verify whether the extracted hostname is a TLD and queries *autoconfig.co.uk*, it could connect to an attacker-controlled server.


admin@example.com
Victim
Email Client
Construct candidate list
DNS MX Query
Back-off MX, construct candidate list
Autoconfig
HTTP Request:
https://autoconfig.example.com/mail/config-v1.1.xml?emailaddress=admin@example.com
...
404 Status, no config found.
DNS MX Request: example.com
DNS Response: example.com MX provider.co.uk
HTTP Request:
https://autoconfig.co.uk/mail/config-v1.1.xml?emailaddress=admin@example.com
...
200 Status, config found.
example.com
Web Server
ns.example.com
DNS
DNS Server
autoconfig.co.uk
WWW
Web Server
Register autoconfig.co.uk
Attacker

```
* OK [CAPABILITY IMAP4rev1 AUTH=PLAIN] Dovecot ready
V LOGIN admin passwords
...
```

Figure 2: An example of the back-off query attack (A2).

2) Insecure parameter settings and parsing: Configuration files list the services supported by the server and the specific parameter settings associated with those services, which clients rely on to establish a connection (we provide detailed parameter definitions in Appendix A). For instance, the connection type is determined by the `SSL` and `Encryption` elements in Autodiscover and the `sockettype` element in Autoconfig. Besides the parameters in configuration files, the service labels in SRV queries (as shown in Table IV) can also be used to distinguish between encrypted-only [50] and STARTTLS connections [23]. In this paper, we focus on parameters related to the connection type (i.e., whether encryption is used), and ignore others.

A3: Server sets only the plaintext connection type. When a server sets only the plaintext connection type in the configuration file (e.g., `socketType` set to *plain*), a client establishes an unencrypted connection, allowing attackers to steal credentials through traffic sniffing (A3.1). For SRV service discovery, if a server does not add records for implicit TLS services, it implies that it supports plaintext and/or STARTTLS. In such cases, it is up to the client implementation to decide which connection type to use for the connection. We consider such servers as insecure if they only add SRV records for services without implicit TLS (A3.2). If a client connects using plaintext, attackers can steal credentials through traffic sniffing. Even if the client upgrades the connection using STARTTLS, attackers may further hack STARTTLS.

A4: Client fails to parse and defaults to plaintext. Server administrators must correctly set the `SSL`, `Encryption`, or `socketType` elements in the configuration file [48], [14]. If the client cannot interpret a value (e.g., `SSL` is set to *starttls*, which is a valid Autoconfig setting but invalid in Autodiscover), it may initiate a plaintext connection by default.

A5: Client fails to auto-configure and defaults to plaintext. If a client cannot retrieve the server's configuration through external requests, built-in lists, or heuristic guessing, it typically defaults to pre-set parameters for user convenience, only requiring the user to enter the hostname. In cases where the client defaults to a plaintext connection type, users (especially those unfamiliar with security) may accept the insecure default settings, risking credentials exposure.

A6: Client implements Autodiscover inadequately. Autodiscover introduces an `Encryption` element, which overrides the `SSL` element. A properly implemented client should check for `Encryption` first and choose the connection type accordingly. If a server sets the `Encryption` element but leaves `SSL` unset, the client that does not strictly follow the specification and ignores the `Encryption` element may default to a plaintext connection. In this case, even if the server is correctly configured, the client's faulty implementation leads to a downgrade.

A7: Client prioritizes SRV records incorrectly. Clients supporting SRV records for auto-configuration must prioritize records correctly according to the standard [32]. If a server prioritizes more secure options (e.g., favoring implicit TLS over plaintext or STARTTLS) and correctly sets the `Priority` and `Weight` values for different services, a client incorrectly implementing the SRV record sorting may mistakenly select a service with a higher, and therefore less preferred, `Priority` value. This flaw in SRV record prioritization could lead to connections being established over less secure channels. Similar to A3, the capabilities required for an attacker depend on how a client processes SRV records.

A8: Client maintains an outdated built-in list. When a client cannot retrieve configuration files in real time through external requests, it relies on a built-in list of providers. If this list is outdated, the client may fail to establish a connection using the latest settings.

3) Discouraged priority settings for services: Configuration files can contain settings for multiple services (e.g., IMAP and POP3), and the order in which these services appear implies their priority. Clients typically prioritize the first service listed. Additionally, RFC 6186 allows servers to indicate preferred services using the `Priority` and `Weight` fields in SRV records.

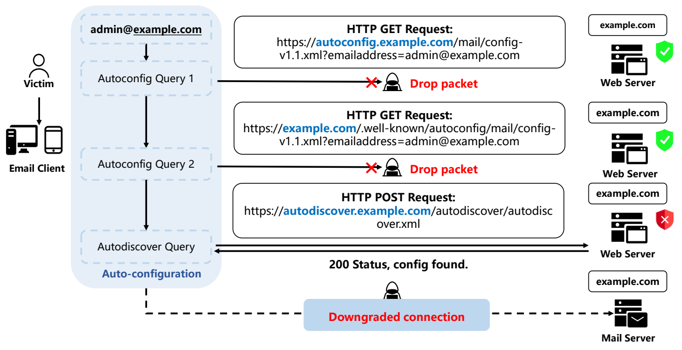


Figure 3: An example of a downgrade attack exploiting inconsistent server configuration information (A10).

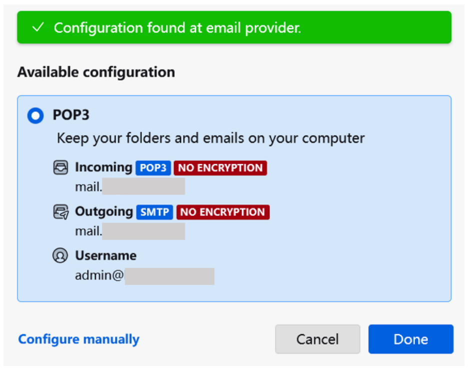
A9: Server prefers insecure connection type. If a server sets STARTTLS or plaintext as the first option, followed by implicit TLS, a compliant client will never establish a connection with the highest security strength. While server administrators may prioritize STARTTLS for compatibility reasons, this is discouraged in the context of email submission and access [62] and is not recommended by RFC standards [25], [50]. Therefore, we also consider it a security defect.

4) Inconsistency of configuration information across and within mechanisms: Clients often support multiple mechanisms simultaneously to enhance user experience. As described in Section III, clients may prioritize certain mechanisms (e.g., prefer Autoconfig over Autodiscover), and there is also an internal priority order for URL paths within a single mechanism. Since these mechanisms differ in URL structures and configuration file definitions, it is crucial for administrators to ensure that configuration information is consistent across all mechanisms. Specifically, considering the connection type settings, inconsistencies in the security strength between mechanisms can lead to a downgrade to less secure connections.

A10: Server sets inconsistent connection types. If a server sets different security strengths for connection types across various auto-configuration mechanisms, an attacker could exploit this by delaying or dropping TCP packets, causing the client to retrieve only the configuration with the least security strength. For example, as shown in Figure 3, this attack could occur if an administrator sets implicit TLS for all Autoconfig configuration files but forgets to update the Autodiscover files, which still use plaintext. If the client prioritizes Autoconfig and sends Query 1 and Query 2 to the server, a MITM attacker could disrupt these requests by dropping packets. The client would then fall back to an Autodiscover query, retrieving a configuration file with a plaintext connection type. As a result, the client would establish an insecure connection, allowing the attacker to steal credentials through traffic sniffing, effectively downgrading the connection from encrypted to plaintext.

C. Inadequate UI Notifications

We analyzed the UI notifications related to various auto-configuration mechanisms. In particular, a defect concerning user confirmation affects all the scenarios mentioned above and increases the likelihood of successful attacks. For instance, an


✓ Configuration found at email provider.

Available configuration

POP3

Keep your folders and emails on your computer

Incoming POP3 NO ENCRYPTION
mail.[redacted]

Outgoing SMTP NO ENCRYPTION
mail.[redacted]

Username
admin@[redacted]

Configure manually | Cancel | Done

Figure 4: UI notifications (UC and W_P) of auto-configuration in Thunderbird.

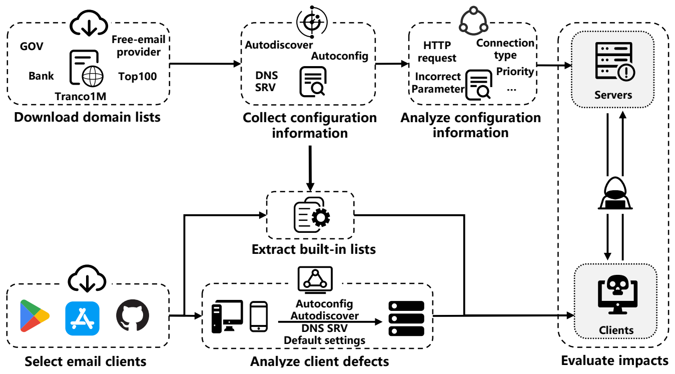


Figure 5: Our methodology to analyze and evaluate defects of servers and clients.

attack can occur silently while the user does not realize it. We summarize 4 UI notifications (including 3 warnings) recommended by public documentations [14], [50], [47], [23] that should appear during the auto-configuration process. Ignoring these notifications could lead to users unknowingly connecting to attacker-controlled servers.

- User confirmation (UC). When a client successfully retrieves the configuration information through any auto-configuration mechanisms as described above, it should first prompt the user to confirm the correctness of the configuration before attempting to log in [14]. If an attacker has tampered with the configuration, the victim's credentials (e.g., passwords) could be silently leaked.
- Plain warning (W_P). In addition to user confirmation, the UI should also emphasize security-sensitive fields, i.e., distinguishing "SSL" and "PLAIN" connection types [50]. The "PLAIN" option should be highlighted to warn users that the connection will be unencrypted, as shown in Figure 4.
- Autodiscover redirect warning (W_{AD}). Autodiscover allows a client to send an HTTP GET request (see Table II), followed by a 302 redirect response containing the destination server URL in the Location header. Since the redirection response stems from an insecure request, attackers could exploit it to intercept configuration information. Therefore, the client should warn the user upon receiving the redirection, and proceed only after the user permits it [47].
- SRV FQDN warning (W_{SR}). RFC 6186 [23] suggests that when the SRV record is not protected by a secure DNS option, the client should verify whether the target FQDN (Fully Qualified Domain Name) of the SRV record matches the queried domain name. Any mismatch should be reported, and let the user decide whether to connect to the host specified in the SRV record.

V. Methodology for Impact Evaluation

In this section, we present the methodology used to evaluate the impact of the attack scenarios discussed above. Our evaluation involved a large-scale measurement of servers and an end-to-end security analysis of clients. Our methodology is illustrated in Figure 5.

Download domain lists. We downloaded the following lists: (1) *Tranco1M*, a list from the Tranco ranking [60] generated on March 25, 2024,[2]; (2) *Top100Provider*, a list of the most popular email domains from Online Email Verification [54]; (3) *GovDomain*, a list of government websites worldwide [71]; (4) *BankDomain*, a list of all bank domains provided by Fondy Payment Service Provider [1]; (5) *FreeDisposableProvider*, a list of free and disposable email service providers (including Guerrilla Mail, Temp Mail) [2]. In particular, we used *PSL* [7] to extract the registrable domain from *GovDomain* and cross-checked the *FreeDisposableProvider* list with data from Hubspot [37] and an open-source list [4]. Our experiments involved a total of 1,053,469 unique domain names.

Collect configuration information. We developed a Golang-based crawler and conducted server scanning for Autodiscover, Autoconfig, and DNS SRV records. The scan was performed from an ECS instance (located in Hong Kong) between March 26 and March 30, 2024. The request paths used are detailed in Appendix B.

- Autodiscover. We generated a list of 10 candidate URLs according to the specifications [48], [47] published by Microsoft. Additionally, to retrieve as many configuration files as possible, we also included plaintext HTTP requests with the GET method. We heuristically set the `EmailAddress` element in the body of the POST request to *info@example.com* by referring to common aliases published by *101domain.com* [81] and RFC 2142 [21].
- Autoconfig. We constructed 6 candidate URLs following the RFC draft [14] and used *info@example.com* as the query parameter in all URLs.
- DNS SRV. We queried all SRV records listed in Table IV and used Unbound [41] as the DNS resolver.

Analyze configuration information. Based on the configuration collected in the previous phase, we analyzed the deployment of auto-configuration across the scanned servers. For configuration files obtained from Autodiscover or Autoconfig, we parsed them based on their respective schemas [48], [14], excluding files with formatting errors. After that, we examined the defects of these servers respectively. Specifically, the analysis approach includes the following aspects:

- Insecure responses. We recorded the URL redirection chain during the request process. We checked if HTTP requests were redirected to HTTPS URLs. For HTTPS requests, we checked if any plaintext URLs appeared in the chain.
- Plain-only connection types. We identified domains that supported only plaintext connection types based on incoming

[2]https://tranco-list.eu/download/Z33PG/1000000

or outgoing server configurations. For SRV records, we checked whether non-encrypted-only service records (e.g., *_submission._tcp*) were added.
- Incorrect parameter settings. We verified whether server-provided connection types adhered to the values defined in the specifications (see more details in Appendix A). In this work, we mainly focus on `socketType`, `SSL`, and `Encryption` elements. Specifically, given the prioritization of services outlined in the configuration file, only the first service listed in the incoming and outgoing servers was evaluated.
- Insecure priority settings. We analyzed if plaintext connections were prioritized over STARTTLS or implicit TLS, or whether STARTTLS was prioritized over implicit TLS.
- Inconsistent connection type. Since each mechanism defines connection types differently, we first unified the configuration parameters and then evaluated consistency across mechanisms (detailed in Appendix E).

Select email clients. We initially included all 7 clients listed in the RFC draft [14]. Then, we searched online blogs and media sites using keywords like "most popular email clients for Android" or "best email clients for Android" on Google. We further extended the list using popularity rankings from Google Play and the Apple App Store. Finally, we checked whether these candidates were open-source by searching for them on GitHub. We did not test cross-platform clients on different platforms, as auto-configuration runs primarily on the OS-independent application layer. In this work, we assumed that their functionality would be consistent (our tests of Gmail for Android and iOS confirmed this assumption). Overall, we selected 29 clients from Windows, Linux, Android, iOS, and macOS, as listed in Table VII.

Extract built-in lists. For open-source clients, we reviewed their codebases and extracted built-in lists as separate files if they existed. We also considered the ISPDB [3] a built-in list and merged all its configuration information into a single file. For other clients that we could not extract built-in lists, we determined their presence by analyzing whether the login screens offered multiple service providers for selection.

Analyze client defects. Our client analysis involved the following parts: (1) support for auto-configuration, (2) default connection types when auto-configuration fails, and (3) defect analysis. We first built a test platform (detailed in Appendix F) that included a mail server, a web server, and a DNS server. We combined the access logs from these servers to track client requests. Additionally, we used Wireshark to capture packets and verify whether the client requests the ISPDB list or performs heuristic guessing. We then suspended all auto-configuration services on our test servers to observe the default connection type clients used when auto-configuration failed. For Outlook, we used its built-in testing tools [67] to make the results more intuitive and accurate. Our defect analysis approach mainly includes the following:

- Plain requests. We tracked client requests using Wireshark and logs from our test servers.
- Inadequate eTLD checking. We first tested whether the client performs *back-off* queries. We added an MX record with *subdomain.example.xyz* for the test domain *example.com* and published configuration information in the corresponding directories of *example.xyz*. For clients that support back-off queries, we checked if they were vulnerable to A2. Specifically, we registered the domain names *example.xyz* and *autoconfig.xyz* and set the MX record for *example.com* to *example.xyz*, and published configuration information at *https://autoconfig.xyz/mail/config-v1.1.xml*. We set the hostname to 127.0.0.1 to prevent users affected by this vulnerability from connecting to our server.
- Insecure configuration parsing. We tested how clients handle configuration parsing errors by setting the `socketType` and `SSL` elements to "xxx" and observing the default configurations. Since the `Encryption` element overrides `SSL`, we set `SSL` to "on" and `Encryption` to "off" to check if clients supporting Autodiscover correctly interpreted the configuration, i.e., resulting in a plaintext connection. For clients that only support Autodiscover for Exchange [49], we skipped tests related to parameter parsing (A4 and A6).
- Non-compliant SRV sorting. We set SRV records with different `Priority` and `Weight` values according to the specification [32] and verified whether clients correctly prioritized the records.
- Outdated built-in lists. We evaluated whether the extracted built-in lists were outdated by comparing them against the configurations obtained from our server scan results.
- Inadequate UI notifications. We tested if Autodiscover-enabled clients displayed warnings when encountering HTTP GET redirects by returning a 302 redirect to a domain we registered. We also added SRV records where the `Target` field had a different FQDN from the queried domain to check for warnings. For clients that retrieve configuration through auto-configuration, we also verified if users were prompted to confirm settings before login and if highlight warnings were displayed for plaintext connections.

Evaluate impacts. We evaluated the impact of the attack scenarios outlined in Table V. We analyzed defects on both the server-side and the client-side and presented detailed evaluation results in Sections VI and VII, respectively. Although the impact of these defects is evaluated separately, practical attacks sometimes require defects on both sides, such as the attacks in A4.

VI. Server Defects in the Wild

Deployment status. We first present the real-world support for auto-configuration mechanisms. Overall, our scan included 1,053,469 domains, with 79,212 (7.52%) deploying at least one auto-configuration mechanism: Autodiscover, Autoconfig, or SRV records. The adoption of Autodiscover (49,538 domains) and Autoconfig (57,331 domains) was comparable, but only 11,281 domains deployed SRV records. This limited adoption may be due to SRV supporting fewer field types [14], [5], making it less flexible. We provide more details about the deployment result in Appendix C.

Summary. As listed in Table V, we focused on five server-side defects: (1) A1 in Type-I, which could result in the victim's email connection being hijacked to an attacker-controlled server, and (2) A3, A4, A9, and A10 in Type-II, which could result in the victim's credentials being exposed. Table VI summarizes our detection results for these defects. Our analysis discovered a total of 49,013 domains with security defects. For Type-I attacks, we identified defects in 43,566

Table VI: The number of domains affected by different attacks introduced by servers.

Domain list	Type-I			Type-II					
	A1.1	A1.2	A1.3	A3.1	A3.2	A4.1	A9.1	A9.2	A10.1
Tranco1M	36,417	28,902	400	762	7,021	1,225	368	1,827	3,220
Top100Provider	7	7	0	3	19	0	0	0	7
GovDomain	756	773	2	9	71	20	7	18	41
BankDomain	64	69	1	1	3	0	0	1	3
FreeDisposableProvider	553	185	10	81	253	2	1	20	197
Total	43,566			11,824					

(55.0%) domains, including 10 domains in the Tranco Top 1K. This suggests that considerable domains still transmit configuration information over plaintext, allowing configuration tampering and email communication hijacking. For Type-II attacks, 11,824 (14.93%) domains were detected, with 15 listed in the Top 1K. Misconfiguration or mismanagement in these services weakens the security of email connections, downgrading them to insecure protocols. Specifically, 2,273 domains could be downgraded to plaintext connections and 5,120 to STARTTLS. We further examined services susceptible to STARTTLS attacks using a command injection testing tool from previous work [62], and found 128 servers (including Yandex's) had buffering flaws. Attackers can further exploit those servers to compromise the victims' credentials. Next, we detail the specifics of the affected domains in each category.

Plain responses (A1.1/A1.2) + Redirection to HTTP (A1.3). We define a well-established server as one that balances security and compatibility. This included two aspects: (1) for HTTP requests, the server redirects requests to an HTTPS URL, and (2) for HTTPS requests, all response URLs in the redirection chain are HTTPS. We analyzed all scan results for Autodiscover and Autoconfig (Table XII in Appendix D) and found that 60.32% (28,905/47,917) and 66.57% (36,464/54,776) of the domains in the Tranco1M list returned insecure responses (i.e., not well-established), respectively.

For Autodiscover, 80.26% (38,456/47,917) of the domains in the Tranco1M list provided configuration information via HTTP requests, of which 75.16% (28,902/38,456) had insecure responses (i.e., returned configuration files directly in plaintext). Additionally, 12.80% (4,924/38,456) could only retrieve configuration information via HTTP requests. Of these, 668 domains were redirected to the *1and1.info* provider via HTTP GET requests. Fortunately, they were all redirected with HTTPS URLs. From the HTTPS requests perspective, 38 domains redirected HTTPS to HTTP. For Autoconfig, 20.49% (11,222/54,776) of the domains in the Tranco1M list could retrieve configuration information through MX records. Among them, 36.37% (4,082/11,222) and 20.88% (2,343/11,222) had their MX hostname set to *emx.mail.ru* and *mx.yandex.ru*, respectively. Overall, 78.94% (43,238/54,776) had their configuration information retrieved through HTTP requests. However, 84.22% (36,417/43,238) did not redirect to HTTPS URLs. In addition, 19.39% (8,383/43,238) could only retrieve configuration information via HTTP requests. Of these, 604 domains were redirected to *privateemail.com* using HTTPS URLs. We also found 380 domains with HTTP URLs in their HTTPS redirection chains, and 181 of these were redirected to *myshoptet.com*.

Plain-only connection (A3.1) + Plain or STARTTLS connection (A3.2). Analysis of the connection types in configuration files revealed that 570 Autodiscover-enabled domains and 503 Autoconfig-enabled domains provided only plaintext connection settings. Among Autoconfig-enabled domains, we identified 3 providers (*zonnet.nl*, *bigpond.com*, and *bigpond.net.au*) from the Top100Provider list. For DNS SRV records, unexpectedly, we found that 92.25% (6,805/7,377) of the Tranco1M domains did not add SUBMISSIONS. This makes the connection type a client can establish with the server dependent on the client's implementation (as discussed in Section IV-B). If the client defaults to plaintext, a plaintext connection will be established; otherwise, an encrypted connection (via STARTTLS) may be established. Fortunately, none of the clients we tested established a plaintext connection in this case.

Incorrect connection type (A4.1). For Autodiscover, we found 925 domains in the Tranco1M list with incorrect parameter values. Among these, 156 domains had `Encryption` set to *starttls*, which is not a valid use of STARTTLS and can not be recognized by clients. Further analysis showed that 112 of these domains were destined to the hosting provider *cyber-folks.pl* located in Poland. We suggested that the incorrect configuration may be due to a lack of clear guidance on setting STARTTLS in Autodiscover, leading providers to configure it improperly. Additionally, we found 668 domains on the Tranco1M list with `SSL` incorrectly set to *yes* instead of *on*. Of these, 198 hostnames point to the provider of *wpx.net* and 113 to *wpxhosting.com*. For Autoconfig, we identified 300 domains with incorrect `socketType` values in the Tranco1M list. Specifically, 237 domains set values such as `%server/imap/socket/%` or `%server/smtp/socket/%`, which are meaningless. Further analysis revealed that the mail server hostnames for these domains pointed to the service provider *kinghost.net*.

Insecure SRV priority (A9.1) + Insecure connection priority (A9.2). For SRV records, we identified 368 domains in the Tranco1M with improper priority settings: 333 IMAP, 88 POP3, and 37 SUBMISSION domains had records prioritized higher than IMAPS, POP3S, and SUBMISSIONS, respectively. For Autodiscover, 584 domains in the Tranco1M list had improper priority settings, with 288 domains relying on email hosting provided by *home.pl* and 109 by *cyber-folks.pl*. *home.pl* prioritized the plaintext connection type of POP3 (port 110) over IMAPS (port 993), while *cyber-folks.pl* supported POP3S (port 995) but favored IMAP (port 143). For Autoconfig, we found 1,367 misconfigured domains. Two service providers, *one.com* and *jino.ru*, were particularly problematic, serving emails for 404 and 63 domains, respectively. Specifically, *one.com*'s SMTP server and *jino.ru*'s IMAP server prioritized STARTTLS over implicit TLS.

Inconsistent connection types (A10.1). Configuration inconsistencies can occur in two ways: (1) within different paths of the same mechanism (as shown in Table X), and (2) between different mechanisms (e.g., *autodiscover.xml* vs. *config-v1.1.xml*). We conducted a detailed comparison of the configurations of Autodiscover, Autoconfig, and SRV records (details are provided in Appendix E for space reasons). First, we analyzed inconsistent settings within the same mechanism. From the Tranco1M list, we found 5,365 Autodiscover-enabled domains and 328 Autoconfig-enabled domains with discrepancies. Particularly, we further analyzed the inconsistencies in connection-type parameters (e.g., `SSL`, `Encryption`, `socketType`), as these could lead to downgrading of connection security. Among 146,046 *autodiscover.xml* files from the Tranco1M list, 58 domains had inconsistencies, with 11 vulnerable to downgrades from STARTTLS or implicit TLS to plaintext. For Autoconfig, among 90,850 *config-v1.1.xml*, 272 domains had inconsistent settings. Second, we compared domains with discrepancies between different configuration mechanisms. Note that if a domain had internal inconsistencies within a mechanism, that mechanism was excluded from comparison. Finally, we found 2,902 domains in the Tranco1M list with inconsistent connection types across different mechanisms, including 7 in the Tranco Top 1K list (e.g., *yandex.com* and *onet.pl*). Of these, 625 domains could be downgraded to plaintext connections due to these inconsistencies.

VII. Analysis Result of Client Defects

Auto-configuration support. As shown in Table VII, all 29 tested clients supported at least one auto-configuration mechanism. Of these, 13 supported Autoconfig, 12 supported Autodiscover, and only 5 supported service discovery via SRV records. Except for Claws Mail, 28 clients had built-in lists, and 14 of them queried the centralized ISPDB database. Additionally, 19 clients implemented heuristic guessing as an auto-configuration mechanism.

Summary. Overall, 22/29 clients were vulnerable to at least one of the attack scenarios in Table V. Specifically, 13 clients could lead to the victim connecting to an attacker-controlled server (i.e., A1 and A2), and 19 clients were susceptible to downgrades to STARTTLS or plaintext, risking credential exposure (i.e., A4-A8). Furthermore, 21/29 clients did not prompt users to confirm server configuration information (i.e., UC), meaning these attacks can be executed silently. Notably, clients were not affected by these attacks mainly because they did not support the relevant auto-configuration mechanisms. Thus, our results are sufficient to conclude that client auto-configuration implementations have widespread security defects.

Plain request (A1.1/A1.2). 13/20 clients that supported Autoconfig or Autodiscover sent plaintext HTTP requests. Surprisingly, 6/13 clients, including Postbox, Kmail, and Nine, initiated only plaintext requests. 5 clients fell back to plaintext if encrypted requests failed. Notably, 9/13 clients used plaintext requests in Autoconfig, which aligns with our server scanning results, showing that transmitting configuration information in plaintext is still prevalent in Autoconfig.

Inadequate eTLD check (A2.1). Except for Nextcloud Mail and Thunderbird, none of the other clients implemented the Autoconfig back-off query. However, Nextcloud Mail used a period ('.') as the delimiter when extracting the `%MXFULLDOMAIN%`, making it vulnerable in the A2 attack scenario. We particularly evaluated the impact of this attack against the Nextcloud Mail implementation in Groupware [58]. We found that 31,281 domains (involving 675 TLDs) meet the attack conditions described in Section IV-B. Using the GoDaddy API [31], we identified 224 registrable *autoconfig.tld* domains, including *autoconfig.net* and *autoconfig.co*. In total, 24,149 domains were susceptible to this attack, 54 of which were within the top 10K domains in the Tranco ranking. It is important to note that even if the server administrator does not deploy any auto-configuration mechanisms, errors in the client implementation can still put users at risk of credentials disclosure. Since we only considered registrable TLDs from GoDaddy, our evaluation only represented the lower bound of the actual impact of this attack.

Plain fallback on parser error (A4.1) + Plain default (A5.1). In scenarios where clients successfully retrieved configuration information through auto-configuration but failed to parse the parameters (A4.1), our experiments showed that 7 clients defaulted to the plaintext connection type. In another scenario (A5.1), 6 clients defaulted to plaintext when they could not obtain the configuration information through auto-configuration. Our experiments revealed variations in default connection types across clients. For example, 4 clients (including Kmail and Nextcloud Mail) defaulted to an encrypted connection type when no configuration file was available (A5.1) but defaulted to plaintext upon parsing failure (A4.1).

Ignoring the `Encryption` element (A6.1). We found that 5 clients, including Thunderbird and FairEmail, did not handle the `Encryption` element when parsing the configuration files to determine the connection type. We analyzed configuration files collected in the scanning module and found 3,942 domains in the Tranco1M list only set the `Encryption` value without specifying `SSL`. Among them, 3 domains were from the Top100Provider list.

Non-compliant SRV sorting (A7.1). We found that FairEmail requested all SRV service records and performed a uniform sort. While this did not raise security issues when following the standard [32], FairEmail sorted SRV records by both `Priority` and `Weight` in descending order, which does not conform to the specification. Fortunately, FairEmail always attempted to establish a secure connection regardless of the origin of the SRV record.

Outdated built-in lists (A8.1). We compared connection type parameters (i.e., whether encryption is used) and found that all extracted built-in lists were outdated. For example, the list from Mailspring was last updated three years ago. Taking the widely-used ISPDB as an example, it provided configurations for 873 domains. By comparing our scan results with ISPDB, we found at least 71 domains in ISPDB had outdated configurations. This was primarily due to one provider's failure to update information timely, affecting 69 domains. Fortunately, no plaintext connection types were found in these files.

UI notification. Our analysis showed that all clients, except Thunderbird and K-9 Mail, had at least one UI-related defect. Only 8 clients prompted users to confirm the results obtained from auto-configuration (UC). When configuration information contained a plaintext connection type, only Thunderbird provided a highlighted warning (W_P), and K-9 Mail required the user to enter the configuration manually. Our analysis results show that, in a real-world scenario, an attacker could attack without the victim's awareness. For instance, once a client retrieves configuration, it proceeds directly to a login attempt using the credentials entered by the user, leaving the victim unaware of a connection to an attacker-controlled server. Additionally, clients that rely on built-in lists for auto-configuration should also confirm the configuration with the

Table VII: Evaluation results of 29 email clients.

Client[1]	Auto-configuration Support[2]					Default Port [P/S][3]		Defect	UI Notification[4]			
	Autoconfig	Autodiscover	DNS SRV	Built-in list	Guess	Incoming	Outgoing		UC	W_P	W_{AD}	W_{SR}
Windows												
Postbox (7.0.60)	●$_P$	○	○	●$_I$	●	143 S	587 S	A1.1, A8.1	✓	✗		
Delta Chat (1.42.1) ∗	●	●	○	●$_I$	●	993 S	465 S	A6.1, A8.1	✗			
Outlook (16.0.10406.20006)	○	●	○	●	●	143 P	25 P	A4.1, A5.1	✗		✓	
Mailbird (3.0.6.0)	○	◐	○	●	●	–	–	A1.2	✗			
eM Client (9.2.2157)	○	●	○	●	●	143 S	587 S		✗		✗	
The bat! (11.0.3.1)	○	○	○	●	●	143 P	25 P	A5.1	✓	✗		
Linux												
Claws Mail (4.2.0git36) ∗	○	○	●	○	○	143 P	25 P	A5.1	✓	✗		✗
Thunderbird (115.6.0) ∗	●	●	○	●$_I$	●	143 P	587 P	A1.1, A4.1, A5.1, A6.1, A8.1	✓	✓	✓	
Kmail (5.24.4) ∗	●$_P$	○	○	●$_I$	○	993 S	25 S	A1.1, A4.1, A8.1	✓	✗		
Evolution (3.50.3) ∗	●	○	●	●$_I$	○	993 S	465 S	A1.1, A4.1, A8.1	✓	✗		✗
Nextcloud Mail (3.5.3) ∗	●	○	○	●$_I$	○	993 S	587 S	A1.1, A2.1, A4.1, A8.1	✗			
Geary (44.1) ∗	●	○	○	●$_I$	●	993 S	465 S	A4.1, A8.1	✗			
Android												
FairEmail (1.2149a) ∗	●	●	●	●$_I$	●	993 S	465 S	A1.1, A6.1, A7.1, A8.1	✗			✗
Nine (4.9.5e)	●$_P$	●	●	●$_I$	●	993 S	465 S	A1.1, A1.2, A6.1, A8.1	✗		✗	✗
MailTime (4.1.5.1218)	●$_P$	◐	○	●$_I$	●	993 S	465 S	A1.1, A8.1	✗		✗	
K-9 Mail (6.714) ∗	●	○	○	●$_I$	○	993 S	465 S	A1.1, A8.1	✓	N/A		
Spark Mail (3.7.2)	●	○	○	●$_I$	●	993 S	587 S	A8.1	✗			
ProfiMail Go (4.32.00)	●	○	○	●$_I$	●	143 P	25 P	A4.1, A5.1, A8.1	✗			
Maildroid (5.22)	○	○	○	●$_I$	●	143 P	25 P	A5.1, A8.1	✗			
iOS												
myMail (14.71.0)	○	○	○	●	●	993 S	465 S		✗			
iOS Mail (17.1)	○	○	○	●	○	993 S	587 S		✗			
Edison Mail (1.53.14)	○	◐	○	●	●	993 S	587 S		✗		✗	
Gmail (6.0.240225)	○	○	○	●	○	993 S	465 S		✗			
Mailbus (3.3.11)	○	◐	○	●	●	993 S	465 S	A1.2	✗			
AltaMail (8.2.5)	○	●$_P$	○	●	●	143 S	25 S	A1.2, A6.1	✗			
MacOS												
Apple Mail (13.5.2)	○	○	○	●	○	993 S	465 S		✗			
Airmail (5.7)	○	○	○	●	●	993 S	465 S		✗			
Mailspring (1.13.3) ∗	○	○	○	●	○	993 S	465 S	A8.1	✓	✗		
Spike (3.8.0)	○	◐$_P$	●	●	○	993 S	587 S	A1.2	✗	✗	✗	✗

[1] ∗ Open-source client.
[2] ○ - Not support. ● - Supported. ◐ - Support Autodiscover for Exchange only. P indicates the client only initiates plaintext requests and I indicates the client queries the ISPDB.
[3] P - Default to plaintext. S - Encrypted connection through implicit TLS or STARTTLS. Mailbird does not provide a default connection type and port, requiring the user to enter manually.
[4] UC - User confirmation. W_P - Plaintext warning. W_{AD} - Autodiscover redirect warning. W_{SR} - SRV FQDN warning. ✓ means a UI notification, and ✗ means no UI notification. K-9 Mail requires the user to enter configuration parameters manually when the configuration information retrieved contains a plaintext connection type.

user before proceeding with the login, as built-in lists may not always be up-to-date (e.g., Mailspring).

VIII. Discussion

In this section, we first analyze the root causes of the security defects in auto-configuration mechanisms, then present mitigation recommendations, and discuss the ethical concerns and limitations of this work.

A. Root Causes

Defects and complexities in protocol design give rise to disparate deployment. Beyond administrator negligence, we believe the primary cause of defects stems from the protocols themselves. First, Autoconfig was originally designed without HTTPS [76], leaving the transmission of configuration files vulnerable to tampering. Second, Autodiscover does not define STARTTLS, which confuses administrators during setup. Our heuristic exploration also revealed that many administrators mistakenly equate the *tls* value in the `Encryption` element with *starttls*. Lastly, regarding SRV service records, RFC 6186 (published in 2011) did not define encrypted-only records for SMTP (i.e., SUBMISSIONS), which were added 7 years later in RFC 8314 (published in 2018). This transition may have led to misconfigurations among administrators who intend to provide encrypted-only services (i.e., implicit TLS). Our analysis showed that 50.35% (3,426/6,805) of the SUBMISSION records from the Tranco1M list incorrectly had port 465 set, contrary to the RFC standard [23].

Lack of guidance for client implementations. As described in Section III, several auto-configuration mechanisms are available for clients. However, these mechanisms are not all well standardized and lack of clear guidance, which makes client implementations of auto-configuration depend on the developer's knowledge of the implemented mechanism. Our client-side experiments revealed that many clients did not strictly follow specifications, leading to two problems: (1) failing to retrieve configuration files, which impacted usability, and (2) introducing security issues. For example, DeltaChat, FairEmail, and Nine only sent GET requests for Autodiscover, potentially preventing the correct retrieval of configuration files.[3] Additionally, clients supporting multiple mechanisms often introduce more security defects, making the security of the auto-configuration process dependent on its weakest link.

Challenges of balancing usability and security. Our scan

[3]This work focuses on security issues, so usability-related issues are not included in Table VII.

results showed that most servers support insecure connections to ensure clients can retrieve configuration files. Moreover, the need for compatibility across different client implementations compels administrators to deploy multiple mechanisms simultaneously, increasing the maintenance burden. Similarly, to enhance user experience, clients tend to support as many mechanisms as possible, potentially introducing more defects. Notably, our analysis revealed that most clients attempted to log in without user confirmation of the configuration information, prioritizing usability at the expense of security. As the saying goes, complexity is the enemy of security [69]. The community should aim for a simpler, more secure auto-configuration mechanism that is easy to deploy and implement, striking a balance between usability and security.

B. Mitigations

Enforcing secure connections, especially in clients. Most servers and clients support insecure connections for compatibility purposes. We believe that clients should take the lead to enforce TLS in auto-configuration, similar to how browsers enforced HTTPS [68]. Moreover, servers should migrate from plaintext or STARTTLS connection type to implicit TLS services and disable support for plaintext connections where possible. We also included such recommendations [50] in our reports to mail providers.

Checking and updating configurations regularly. Server administrators should regularly check and update their published configuration information. We have released a testing tool available at https://github.com/emailconfigtest/mailconfig, which supports querying and comparing mail domain configuration information across different mechanisms. We also included built-in lists of open-source projects in our tool to help administrators identify outdated and inconsistent configurations. For client developers, it is essential to ensure that built-in lists contain up-to-date configurations. While querying the ISPDB [3] in real-time is a good practice, maintaining secure and reliable configuration information in the ISPDB requires coordinated efforts from the community.

Implementing professional clients. A client should consider the effectiveness and security of auto-configuration mechanisms. For effectiveness, the client should support multiple mechanisms to maximize its ability to retrieve configuration information. For security, it must carefully extract domains (e.g., using tools like *PSL* [7]) to construct candidate URLs and use HTTPS for configuration retrieval. Regardless of any failures (e.g., parsing issues), the client should provide encrypted connection types by default. Adequate UI notifications should also be implemented as a defense against attacks. Additionally, when employing multiple auto-configuration mechanisms, the client should compare the results and, in the case of inconsistencies, apply the most secure connection type.

C. Ethical Concerns and Responsible Disclosure

All our experiments focused on publicly accessible services without collecting personal information. We adhered to best practices for Internet measurements as outlined in [12], [27]. We declared the purpose of our measurements on web pages and set up a PTR record. Client analysis experiments were conducted on our own platform, and no users were affected.

We contacted all affected clients using the following methods: (1) submitting reports to their SRC or HackerOne.[4] (e.g., Outlook and Nextcloud Mail) (2) raising issues on forums (e.g., GNOME,[5] Bugzilla[6]) or emailing maintainers via their provided security policy. (3) contacting security teams (e.g., security@address) or submitting forms on their web portals. In general, we have contacted all 22 clients and received responses from 10 clients so far (including Thunderbird, Nextcloud Mail, and Mailspring), confirming some or all of the defects reported. Specifically, Nextcloud Mail and KMail assigned CVE identifiers to the vulnerabilities associated with A2 and A1.1, respectively. Nextcloud Mail also rewarded us for our findings. FairEmail fixed all reported defects except for the plaintext request in Autoconfig, stating that many email providers still use insecure connections to transmit configuration files. For the centralized ISPDB database [3], we raised an issue on GitHub and attached all the domains with outdated configurations, which was confirmed by the developers and has been updated.

We also launched a notification campaign for all affected domains. Following the practices from previous research [79], [72], we sent reports to their dedicated email addresses, including security@, support@, abuse@, postmaster@ and info@. For reports that failed to deliver through the above email addresses, we have contacted those domains by visiting their web portals and extracting email addresses [77]. Overall, we have received 1,340 response emails so far, including 753 automated responses from ticketing systems. 93 domains (including a ranked top 500 domain) acknowledged us for the report and are in the process of being fixed.

D. Limitations

First, we evaluated the security of email auto-configuration based on the configuration information provided by the server. We did not establish connections to mail servers to analyze which email service ports were opened and connection types they supported. Second, for clients using HTTPS requests to download configuration files, we did not investigate whether trusted root certificates were deployed and the client performed strict certificate validation, such as hostname matching. Lastly, since Autoconfig is not yet standardized, our measurements only represent a snapshot of its adoption across servers at the time of the study. Future results may vary as the standard evolves and adoption patterns shift.

This work does not consider DNS resolve-related security threats. While we assume that Type-I attackers can tamper with TCP packets, we consider ISPs trustworthy since recursive resolution typically occurs within their network. Clients have limited defenses against attacks at recursive resolution, and even DNSSEC does not directly protect query results here. Although RFC 8314 [50] advises against connections based on unsigned SRV records, an active attacker within ISPs could modify recursive DNS responses, including DNSSEC validation flags. Such threats are not specific to SRV records and are applied to all network applications relying on domains.

[4]https://www.hackerone.com/

[5]https://gitlab.gnome.org/GNOME

[6]https://bugzilla.mozilla.org/home

IX. Conclusion

In this paper, we performed the first systematic security analysis of email auto-configuration in the wild, revealing widespread defects in server deployment and client implementation. We summarized 10 attack scenarios, including 8 newly identified defects. These attacks could result in victims connecting to attacker-controlled servers or leaking credentials. Among the 79,212 domains supporting one or more auto-configuration mechanisms, 49,013 domains were deployed with defects. Of these, 43,566 and 11,824 domains are vulnerable to the two attacks, respectively. Among the 29 analyzed clients, 22 were affected by at least one attack scenario, and 21 did not adequately prompt users to confirm the configurations.

These findings demonstrate that current server deployments and client implementations of email auto-configuration bring security weaknesses to email services. Professional practices and implementation guidelines are imperative to address the defects due to misconfiguration, mismanagement, and flawed implementation. Moreover, the community should prioritize the security concerns surrounding auto-configuration and take actions to eliminate the defects due to compatibility.

Acknowledgment

We thank all the anonymous reviewers for their insightful feedback on improving this paper. This research was partially supported by National Key RD Plan of China under Grant 2020YFB1005803, National Natural Science Foundation of China under Grant (62302258, 62472021), Beijing Natural Science Foundation under Grant 4242023, and Youth Top Talent Support Program of Beihang University under Grant YWF-22-L-1272. Yiming Zhang is in part supported by the Shuimu Tsinghua Scholar Program.

References

[1] "All banks domains and IPs," https://github.com/cloudipsp/all_banks_ips, accessed: 2024-04-16.

[2] "Free or Disposable Email Providers Domains - Collected and combined from various resources primarily built on top of lists provided by Okutbay & frankwarwick," https://gist.github.com/drakodev/e85c1fd6d9ac8634786d6139e0066fa0, accessed: 2024-04-16.

[3] "Ispdb - generic database of mail server configuration," https://github.com/thunderbird/autoconfig/tree/master/ispdb, accessed: 2024-10-05.

[4] "A list of domains for disposable and temporary email addresses," https://gist.github.com/adamloving/4401361, accessed: 2024-04-16.

[5] "Mail account autoconfiguration via DNS SRV (possibly with DNSSEC) rfc6186," https://bugzilla.mozilla.org/show_bug.cgi?id=342242, accessed: 2024-04-26.

[6] "Nodemailer," https://github.com/nodemailer/nodemailer., accessed: 2024-03-31.

[7] "Public suffix list," https://publicsuffix.org/, accessed: 2024-04-09.

[8] Amit Serper, "Autodiscovering the great leak," Sep. 2021, https://www.akamai.com/blog/security/autodiscovering-the-great-leak, accessed: 2024-10-05.

[9] Apple Inc., "Mail - Official Apple Support," https://support.apple.com/mail, accessed: 2024-10-05.

[10] M. I. Ashiq, W. Li, T. Fiebig, and T. Chung, "You've got report: Measurement and security implications of DMARC reporting," in *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, J. A. Calandrino and C. Troncoso, Eds. USENIX Association, 2023, pp. 4123–4137. [Online]. Available: https://www.usenix.org/conference/usenixsecurity23/presentation/ashiq

[11] ——, "SPF beyond the standard: Management and operational challenges in practice and practical recommendations," in *33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024*, D. Balzarotti and W. Xu, Eds. USENIX Association, 2024. [Online]. Available: https://www.usenix.org/conference/usenixsecurity24/presentation/ashiq

[12] M. D. Bailey, D. Dittrich, E. Kenneally, and D. Maughan, "The menlo report," *IEEE Secur. Priv.*, vol. 10, no. 2, pp. 71–75, 2012. [Online]. Available: https://doi.org/10.1109/MSP.2012.52

[13] B. Blechschmidt and B. Stock, "Extended hell(o): A comprehensive large-scale study on email confidentiality and integrity mechanisms in the wild," in *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, J. A. Calandrino and C. Troncoso, Eds. USENIX Association, 2023, pp. 4895–4912. [Online]. Available: https://www.usenix.org/conference/usenixsecurity23/presentation/blechschmidt

[14] B. Bucksch, "Mail Autoconfig," Internet Engineering Task Force, Internet-Draft draft-bucksch-autoconfig-00, https://datatracker.ietf.org/doc/draft-bucksch-autoconfig/00/, Work in Progress.

[15] B. Bucksch, "Proposal: Auto-configuration," 2008, https://groups.google.com/g/mozilla.dev.apps.thunderbird/c/6L2wrzGWGQg#a73bd97251b18777, accessed: 2024-10-05.

[16] L. Ceci, "Emails sent per day 2025," https://www.statista.com/statistics/456500/daily-number-of-e-mails-worldwide/, accessed: 2024-04-29.

[17] J. Chen, V. Paxson, and J. Jiang, "Composition kills: A case study of email sender authentication," in *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*, S. Capkun and F. Roesner, Eds. USENIX Association, 2020, pp. 2183–2199. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/chen-jianjun

[18] I. S. Consortium, "BIND 9," https://www.isc.org/bind/, accessed: 2024-10-05.

[19] M. R. Crispin, "Internet message access protocol - version 4," *RFC*, vol. 1730, pp. 1–77, 1994. [Online]. Available: https://doi.org/10.17487/RFC1730

[20] D. Crocker, "Standard for the format of ARPA internet messages," *RFC*, vol. 822, pp. 1–49, Aug. 1982. [Online]. Available: https://doi.org/10.17487/RFC0822

[21] ——, "Mailbox names for common services, roles and functions," *RFC*, vol. 2142, pp. 1–6, May 1997. [Online]. Available: https://doi.org/10.17487/RFC2142

[22] ——, "Internet mail architecture," *RFC*, vol. 5598, pp. 1–54, Jul. 2009. [Online]. Available: https://doi.org/10.17487/RFC5598

[23] C. Daboo, "Use of SRV records for locating email submission/access services," *RFC*, vol. 6186, pp. 1–9, 2011. [Online]. Available: https://doi.org/10.17487/RFC6186

[24] Dovecot, "The Secure IMAP server," https://www.dovecot.org/download/, accessed: 2024-10-05.

[25] V. Dukhovni, "Opportunistic security: Some protection most of the time," *RFC*, vol. 7435, pp. 1–11, Dec. 2014. [Online]. Available: https://doi.org/10.17487/RFC7435

[26] Z. Durumeric, D. Adrian, A. Mirian, J. Kasten, E. Bursztein, N. Lidzborski, K. Thomas, V. Eranti, M. D. Bailey, and J. A. Halderman, "Neither snow nor rain nor MITM...: an empirical analysis of email delivery security," in *Proceedings of the 2015 ACM Internet Measurement Conference, IMC 2015, Tokyo, Japan, October 28-30, 2015*, K. Cho, K. Fukuda, V. S. Pai, and N. Spring, Eds. ACM, 2015, pp. 27–39. [Online]. Available: https://doi.org/10.1145/2815675.2815695

[27] Z. Durumeric, E. Wustrow, and J. A. Halderman, "Zmap: Fast internet-wide scanning and its security applications," in *Proceedings of the 22th USENIX Security Symposium, Washington, DC, USA, August 14-16, 2013*, S. T. King, Ed. USENIX Association, 2013, pp. 605–620. [Online]. Available: https://www.usenix.org/conference/usenixsecurity13/technical-sessions/paper/durumeric

[28] I. D. Foster, J. Larson, M. Masich, A. C. Snoeren, S. Savage, and K. Levchenko, "Security by any other name: On the effectiveness of provider based email security," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, I. Ray, N. Li, and

C. Kruegel, Eds. ACM, 2015, pp. 450–464. [Online]. Available: https://doi.org/10.1145/2810103.2813607

[29] R. Gellens and J. C. Klensin, "Message submission," *RFC*, vol. 2476, pp. 1–15, Dec. 1998. [Online]. Available: https://doi.org/10.17487/RFC2476

[30] ——, "Message submission for mail," *RFC*, vol. 6409, pp. 1–20, Nov. 2011. [Online]. Available: https://doi.org/10.17487/RFC6409

[31] Godaddy, "Domains API," https://developer.godaddy.com/doc/endpoint/domains, accessed: 2024-04-26.

[32] A. Gulbrandsen, P. Vixie, and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)," *RFC*, vol. 2782, pp. 1–12, Feb. 2000. [Online]. Available: https://doi.org/10.17487/RFC2782

[33] P. E. Hoffman, "SMTP service extension for secure SMTP over transport layer security," *RFC*, vol. 3207, pp. 1–9, 2002. [Online]. Available: https://doi.org/10.17487/RFC3207

[34] R. Holz, J. Amann, O. Mehani, M. A. Kâafar, and M. Wachs, "TLS in the wild: An internet-wide analysis of tls-based protocols for electronic communication," in *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*. The Internet Society, 2016.

[35] H. Hu and G. Wang, "End-to-end measurements of email spoofing attacks," in *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, W. Enck and A. P. Felt, Eds. USENIX Association, 2018, pp. 1095–1112. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/hu

[36] K. Huang, "Email Client Market Share in August 2021: Email Clients Hold Steady," Sep. 2021, https://www.litmus.com/blog/email-client-market-share-august-2021, accessed: 2024-04-29.

[37] Hubspot, "Internet message access from form submissions," https://knowledge.hubspot.com/forms/what-domains-are-blocked-when-using-the-forms-email-domains-to-block-feature, accessed: 2024-04-16.

[38] F. Ising, D. Poddebniak, T. Kappert, C. Saatjohann, and S. Schinzel, "Content-type: multipart/oracle - tapping into format oracles in email end-to-end encryption," in *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, J. A. Calandrino and C. Troncoso, Eds. USENIX Association, 2023, pp. 4175–4192. [Online]. Available: https://www.usenix.org/conference/usenixsecurity23/presentation/ising

[39] J. C. Klensin, "Simple mail transfer protocol," *RFC*, vol. 5321, pp. 1–95, Oct. 2008. [Online]. Available: https://doi.org/10.17487/RFC5321

[40] J. Kundrát., "Trojita 0.4.1, a security update for CVE-2014-2567," http://jkt.flaska.net/blog/Trojita_0_4_1__a_security_update_for_CVE_2014_2567.html, accessed: 2024-07-06.

[41] N. Labs, "Unbound DNS Resolver," https://nlnetlabs.nl/projects/unbound/about/, accessed: 2024-04-26.

[42] H. Lee, M. I. Ashiq, M. Müller, R. van Rijswijk-Deij, T. T. Kwon, and T. Chung, "Under the hood of DANE mismanagement in SMTP," in *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, K. R. B. Butler and K. Thomas, Eds. USENIX Association, 2022, pp. 1–16. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/lee

[43] H. Lee, A. Gireesh, R. van Rijswijk-Deij, T. Kwon, and T. Chung, "A longitudinal and comprehensive study of the DANE ecosystem in email," in *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*, S. Capkun and F. Roesner, Eds. USENIX Association, 2020, pp. 613–630. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/lee-hyeonmin

[44] J. Ma, L. Chen, K. Xue, B. Luo, X. Huang, M. Ai, H. Zhang, D. S. L. Wei, and Y. Zhuang, "Fakebehalf: Imperceptible email spoofing attacks against the delegation mechanism in email systems," in *33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024*, D. Balzarotti and W. Xu, Eds. USENIX Association, 2024. [Online]. Available: https://www.usenix.org/conference/usenixsecurity24/presentation/ma-jinrui

[45] W. Mayer, A. Zauner, M. Schmiedecker, and M. Huber, "No need for black chambers: Testing TLS in the e-mail ecosystem at large," in *11th International Conference on Availability, Reliability and Security, ARES 2016, Salzburg, Austria, August 31 - September 2, 2016*. IEEE Computer Society, 2016, pp. 10–20. [Online]. Available: https://doi.org/10.1109/ARES.2016.11

[46] Microsoft, "Exchange 2007 Autodiscover and certificates," https://techcommunity.microsoft.com/t5/exchange-team-blog/exchange-2007-autodiscover-and-certificates/ba-p/593753?WT.mc_id=M365-MVP-9501, accessed: 2024-10-05.

[47] ——, "[MS-OXDISCO]: Autodiscover HTTP Service Protocol," Aug. 2021, https://learn.microsoft.com/en-us/openspecs/exchange_server_protocols/ms-oxdisco/d912502b-c0e2-41a1-8b0e-f714ba523e08, accessed: 2024-03-06.

[48] ——, "[MS-OXDSCLI]: Autodiscover Publishing and Lookup Protocol," Aug. 2021, https://learn.microsoft.com/en-us/openspecs/exchange_server_protocols/ms-oxdscli/78530279-d042-4eb0-a1f4-03b18143cd19, accessed: 2024-03-06.

[49] ——, "Autodiscover for Exchange," Sep. 2022, https://learn.microsoft.com/en-us/exchange/client-developer/exchange-web-services/autodiscover-for-exchange, accessed: 2024-04-26.

[50] K. Moore and C. Newman, "Cleartext considered obsolete: Use of transport layer security (TLS) for email submission and access," *RFC*, vol. 8314, pp. 1–26, Jan. 2018. [Online]. Available: https://doi.org/10.17487/RFC8314

[51] MozillaWiki, "Thunderbird:Autoconfiguration - MozillaWiki," 2021, https://wiki.mozilla.org/Thunderbird:Autoconfiguration, accessed: 2024-10-05.

[52] J. Müller, M. Brinkmann, D. Poddebniak, H. Böck, S. Schinzel, J. Somorovsky, and J. Schwenk, ""johnny, you are fired!" - spoofing openpgp and S/MIME signatures in emails," in *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, N. Heninger and P. Traynor, Eds. USENIX Association, 2019, pp. 1011–1028. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/muller

[53] J. Müller, M. Brinkmann, D. Poddebniak, S. Schinzel, and J. Schwenk, "Re: What's up johnny? - covert content attacks on email end-to-end encryption," in *Applied Cryptography and Network Security - 17th International Conference, ACNS 2019, Bogota, Colombia, June 5-7, 2019, Proceedings*, ser. Lecture Notes in Computer Science, R. H. Deng, V. Gauthier-Umaña, M. Ochoa, and M. Yung, Eds., vol. 11464. Springer, 2019, pp. 24–42. [Online]. Available: https://doi.org/10.1007/978-3-030-21568-2_2

[54] My-Addr, "List of most popular email domains (by number of live emails)," 2016, https://email-verify.my-addr.com/list-of-most-popular-email-domains.php, accessed: 2024-10-05.

[55] J. G. Myers and M. T. Rose, "Post office protocol - version 3," *RFC*, vol. 1939, pp. 1–23, May 1996. [Online]. Available: https://doi.org/10.17487/RFC1939

[56] I. Nesterov and M. Goncharov, "All your emails belong to us: exploiting vulnerable email clients via domain name collision," *Black Hat Asia*, 2017.

[57] C. Newman, "Using TLS with imap, POP3 and ACAP," *RFC*, vol. 2595, pp. 1–15, Jun. 1999. [Online]. Available: https://doi.org/10.17487/RFC2595

[58] Nextcloud, "Nextcloud Groupware," https://nextcloud.com/groupware/, accessed: 2024-10-05.

[59] Nginx, "Nginx Release Version," https://nginx.org/en/download.html, accessed: 2024-10-05.

[60] V. L. Pochat, T. van Goethem, S. Tajalizadehkhoob, M. Korczynski, and W. Joosen, "Tranco: A research-oriented top sites ranking hardened against manipulation," in *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society, 2019. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/tranco-a-research-oriented-top-sites-ranking-hardened-against-manipulation/

[61] D. Poddebniak, C. Dresen, J. Müller, F. Ising, S. Schinzel, S. Friedberger, J. Somorovsky, and J. Schwenk, "Efail: Breaking S/MIME and openpgp email encryption using exfiltration channels," in *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, W. Enck and A. P. Felt, Eds. USENIX Association, 2018, pp. 549–566. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/poddebniak

[62] D. Poddebniak, F. Ising, H. Böck, and S. Schinzel, "Why TLS is better without STARTTLS: A security analysis of STARTTLS in the email context," in *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, M. D. Bailey and R. Greenstadt, Eds.

USENIX Association, 2021, pp. 4365–4382. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/poddebniak

[63] J. Postel, “Simple mail transfer protocol,” *RFC*, vol. 821, pp. 1–72, Aug. 1982. [Online]. Available: https://doi.org/10.17487/RFC0821

[64] Postfix, “Postfix Announcements,” https://www.postfix.org/announcements.html, accessed: 2024-10-05.

[65] E. Rescorla, “HTTP over TLS,” *RFC*, vol. 2818, pp. 1–7, May 2000. [Online]. Available: https://doi.org/10.17487/RFC2818

[66] ——, “The transport layer security (TLS) protocol version 1.3,” *RFC*, vol. 8446, pp. 1–160, Aug. 2018. [Online]. Available: https://doi.org/10.17487/RFC8446

[67] P. Rice, “How to Test Autodiscover Functionality in Microsoft Outlook,” Nov. 2018, https://www.prrcomputers.com/blog/how-to-test-autodiscover-functionality-in-microsoft-outlook/, accessed: 2024-07-01.

[68] E. Schechter, “A milestone for Chrome security: marking HTTP as “not secure”,” Jul. 2018, https://blog.google/products/chrome/milestone-chrome-security-marking-http-not-secure/, accessed: 2024-08-27.

[69] B. Schneier, “Essays: A Plea for Simplicity - Schneier on Security,” https://www.schneier.com/essays/archives/1999/11/a_plea_for_simplicit.html, accessed: 2024-10-05.

[70] K. Shen, C. Wang, M. Guo, X. Zheng, C. Lu, 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 *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, M. D. Bailey and R. Greenstadt, Eds. USENIX Association, 2021, pp. 3201–3217. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/shen-kaiwen

[71] S. Singanamalla, E. H. B. Jang, R. J. Anderson, T. Kohno, and K. Heimerl, “Accept the risk and continue: Measuring the long tail of government https adoption,” in *IMC ’20: ACM Internet Measurement Conference, Virtual Event, USA, October 27-29, 2020*. ACM, 2020, pp. 577–597. [Online]. Available: https://doi.org/10.1145/3419394.3423645

[72] B. Stock, G. Pellegrino, F. Li, M. Backes, and C. Rossow, “Didn’t you hear me? - towards more successful web vulnerability notifications,” in *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018.

[73] C. Stransky, O. Wiese, V. Roth, Y. Acar, and S. Fahl, “27 years and 81 million opportunities later: Investigating the use of email encryption for an entire university,” in *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*. IEEE, 2022, pp. 860–875. [Online]. Available: https://doi.org/10.1109/SP46214.2022.9833755

[74] D. Tatang, R. Flume, and T. Holz, “Extended abstract: A first large-scale analysis on usage of MTA-STS,” in *Detection of Intrusions and Malware, and Vulnerability Assessment - 18th International Conference, DIMVA 2021, Virtual Event, July 14-16, 2021, Proceedings*, ser. Lecture Notes in Computer Science, L. Bilge, L. Cavallaro, G. Pellegrino, and N. Neves, Eds., vol. 12756. Springer, 2021, pp. 361–370. [Online]. Available: https://doi.org/10.1007/978-3-030-80825-9_18

[75] D. Tatang, F. Zettl, and T. Holz, “The evolution of dns-based email authentication: Measuring adoption and finding flaws,” in *RAID ’21: 24th International Symposium on Research in Attacks, Intrusions and Defenses, San Sebastian, Spain, October 6-8, 2021*, L. Bilge and T. Dumitras, Eds. ACM, 2021, pp. 354–369. [Online]. Available: https://doi.org/10.1145/3471621.3471842

[76] Thunderbird, “Thunderbird Autoconfiguration,” https://www.bucksch.org/1/projects/thunderbird/autoconfiguration/#ISPDB, accessed: 2024-10-05.

[77] C. Utz, M. Michels, M. Degeling, N. Marnau, and B. Stock, “Comparing large-scale privacy and security notifications,” *Proc. Priv. Enhancing Technol.*, vol. 2023, no. 3, pp. 173–193, 2023. [Online]. Available: https://doi.org/10.56553/popets-2023-0076

[78] W. Venema, “Plaintext command injection in multiple implementations of STARTTLS (CVE-2011-0411),” https://www.postfix.org/CVE-2011-0411.html, accessed: 2024-04-09.

[79] C. Wang, Y. Kuranaga, Y. Wang, M. Zhang, L. Zheng, X. Li, J. Chen, H. Duan, Y. Lin, and Q. Pan, “Breakspf: How shared infrastructures magnify SPF vulnerabilities across the internet,” in *31st Annual Network and Distributed System Security Symposium, NDSS 2024, San Diego, California, USA, February 26 - March 1, 2024*. The Internet Society, 2024. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/breakspf-how-shared-infrastructures-magnify-spf-vulnerabilities-across-the-internet/

[80] 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 *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, K. R. B. Butler and K. Thomas, Eds. USENIX Association, 2022, pp. 1185–1201. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/wang-chuhan

[81] G. Workspace, “The Most Common Email Aliases Backed by Data,” Aug. 2022, https://blog.101domain.com/google-workspace/most-common-email-aliases, accessed: 2024-04-16.

APPENDIX

A. Autodiscover Request Body and Parameter Definitions

The specification [48] provides an example of the Autodiscover request body (as shown in List 1), which is a formatted XML containing the `EmailAddress` element that identifies the email address (e.g., *user@example.com*) for which the configuration information will be retrieved. The same specification [48] also defines the elements included in the Autodiscover response (i.e., *autodiscover.xml*, as shown in List 2). For Autoconfig, since there is currently no formal standard, we refer to the relevant RFC drafts [14] for the configuration file definitions (i.e., *config-v1.1.xml*). Tables VIII and IX show the definitions of some of the elements in these configuration files, respectively.

```
<?xml version='1.0' encoding='utf-8' ?>
<Autodiscover xmlns="http://schemas.microsoft.com/exchange/autodiscover/outlook/requestschema/2006">
    <Request>
        <EMailAddress>user@example.com</EMailAddress>
        <AcceptableResponseSchema>http://schemas.microsoft.com/exchange/autodiscover/outlook/responseschema/2006a</AcceptableResponseSchema>
    </Request>
</Autodiscover>
```

Listing 1: An example of the body of POST request in Autodiscover.

```
<AccountType>email</AccountType>
<Action>settings</Action>
<Protocol>
    <Type>IMAP</Type>
    <Server>imap.example.com</Server>
    <Port>993</Port>
    <SPA>off</SPA>
    <SSL>on</SSL>
    <TTL>0</TTL>
    <Encryption>SSL</Encryption>
</Protocol>
<Protocol>
    <Type>SMTP</Type>
    <Server>smtp.example.com</Server>
    <Port>465</Port>
    <SPA>off</SPA>
    <SSL>on</SSL>
    <TTL>0</TTL>
    <Encryption>SSL</Encryption>
</Protocol>
```

Listing 2: An example of *autodiscover.xml*. The bold text indicates configuration information for IMAP and SMTP servers.

Table VIII: Definitions of key element values of *autodiscover.xml*.

Element	Value	Meaning
Server	*any*	The hostname of the mail server.
Port	*any*	Typically is a well-known port, e.g., 993, 995, 465.
SSL	*on*, *off*	Whether to establish an encrypted connection, default is "on".
SPA	*on*, *off*	Whether secure password authentication is required, default is "on".
Encryption	*none*, *ssl*, *tls*, *auto*	If present, overrides the SSL element. "none" represents no encryption is used. "ssl" and "tls" stand for Secure Sockets Layer (SSL) or Transport Layer Security (TLS) is used, respectively, where SSL is superseded by TLS. "auto" represents using the most secure encryption that both client and server support.

[1] Note that Autodiscover specification [47] does not define STARTTLS. In our experiments, we determine whether the connection type is STARTTLS based on the well-known port.

Table IX: Definitions of key element values of *config-v1.1.xml*.

Element	Value	Meaning
hostname	*any*	The hostname of the mail server.
port	*any*	Typically is a well-known port, e.g., 993, 995, 465.
socketType	*plain*, *starttls*, *ssl*	Plaintext, or encrypted connection via STARTTLS or SSL.
authentication	*password-cleartext*, *password-encrypted*, *GSSAPI*, *NTLM*, *client-IP-address*, *TLS-client-cert*, *OAuth2*, *none*	Authentication methods.

B. Request Paths in Server Scanning

As shown in Table X, our scanning module includes 10 candidate URLs for Autodiscover and 6 for Autoconfig.

C. Deployment of Auto-configuration

We retrieved a total of 152,646 *autodiscover.xml* files from Autodiscover, 95,070 *config-v1.1.xml* files from Autoconfig, and 31,499 SRV records. We filtered out files that could not be parsed or had no configuration information. The results of our scan are shown in Table XI. Specifically, among the 1,053,469 domains scanned, 79,212 (7.52%) domains support at least one auto-configuration mechanism (Autodiscover, Autoconfig, or SRV records). Autodiscover and Autoconfig were supported by 49,538 and 57,331 domains, respectively, while only 11,281 domains supported SRV. From the perspective of different domain lists, as expected, the Top100Provider list showed the highest deployment rate, with 30% of domains supporting auto-configuration.

We further analyzed the distribution for auto-configuration support. Our results showed that very few domains support all three mechanisms at the same time. For example, only 2.23% (1,695/76,104) domains in the Tranco1M list support all three mechanisms. The Top100Provider and FreeDisposableProvier lists, which are more closely tied to email services, have 26.67% (8/30) and 9.94% (190/1,911) of domains that support all three mechanisms simultaneously.

D. Evaluation Results of Autodiscover and Autoconfig for A1.

Table XII presents the evaluation results of Autodisocver and Autoconfig for A1 (refer to Section IV-B and Section VI).

E. Comparison Method for Configuration Information

Considering the prioritization of services in the configuration files, we extracted only the first parameter setting of the incoming or outgoing server in each mechanism's results for comparison. Specifically, we focused on the `sockettype`, `SSL`, and `Encryption` elements related to connection types. We aligned the configuration information obtained from Autodiscover and SRV records with the value of `socketType` defined in Autoconfig as a reference. For Autodiscover, the `Encryption` element, if present, overrides the SSL element as defined in Table VIII.

When `Encryption` was set to *ssl* or *tls*, we further determined the connection type to be *ssl* or *starttls* based on the port. By default, we set the connection type to *starttls* for ports 587, 143, and 110, and *ssl* for ports 465, 993, and 995. If `Encryption` was set to *auto*, we determined the connection type based on whether the port was well-known (see Table I), defaulting to *starttls* for unfamiliar ports. We also applied this default setting if there were incorrect values in the `sockettype`, `SSL`, or `Encryption` elements. For SRV service records, we prioritized encrypted-only service records (IMAPS, POP3S, and SUBMISSIONS). If the server lacked such records, we determined the connection type based on the non-encrypted-only service records and whether their ports were familiar. Note that the analysis results presented in this paper are based on the processing methods defined here. Actual results may vary depending on specific client implementations.

F. Test Platform

Our platform was set up with a mail server running Postfix [64] and Dovecot [24], and a web server running Nginx [59] on CentOS 7. We applied certificates from Let's Encrypt for both servers. For the mail server, we enabled both implicit TLS and STARTTLS services on the well-known ports for IMAP and SMTP protocols. We did not test the POP3 protocol here since we were primarily concerned with the auto-configuration mechanism, and the exact protocol used is trivial. We hosted both *autodiscover.xml* and *config-v1.1.xml* files in the appropriate directories on the web server and logged request paths and errors in *access.log*. To track DNS queries from clients, we set up an authoritative DNS server running Bind9 [18] on Ubuntu 22.04.1. We added SRV records for all services listed in Table IV. In addition, we also added an SRV record for *_autodiscover._tcp* to support Autodiscover queries.

Table X: The request paths in Autodiscover and Autoconfig scanning module.

Path	URL	Request Method
Autodiscover		
1	http://autodiscover.example.com/autodiscover/autodiscover.xml	HTTP POST
2	https://example.com/autodiscover/autodiscover.xml	HTTP POST
3	_autodiscover._tcp.example.com. IN SRV 0 0 443 target.com.	DNS SRV request for Autodiscover server
	https://target.com/autodiscover/autodiscover.xml	HTTP POST
4	http://autodiscover.example.com/autodiscover/autodiscover.xml	HTTP GET for initial request, POST for redirection
5	https://autodiscover.example.com/autodiscover/autodiscover.xml	HTTP POST
6	http://example.com/autodiscover/autodiscover.xml	HTTP POST
7	https://autodiscover.example.com/autodiscover/autodiscover.xml	HTTP GET
8	https://example.com/autodiscover/autodiscover.xml	HTTP GET
9	http://autodiscover.example.com/autodiscover/autodiscover.xml	HTTP GET
10	http://example.com/autodiscover/autodiscover.xml	HTTP GET
Autoconfig		
1	https://autoconfig.example.com/mail/config-v1.1.xml?emailaddress=info@example.com	HTTP GET
2	https://example.com/.well-known/autoconfig/mail/config-v1.1.xml?emailaddress=info@example.com	HTTP GET
3	http://autoconfig.example.com/mail/config-v1.1.xml?emailaddress=info@example.com	HTTP GET
4	http://example.com/.well-known/autoconfig/mail/config-v1.1.xml?emailaddress=info@example.com	HTTP GET
5	example.com. IN MX 0 mx.mail.provider.com.	DNS MX request for mail provider
	https://autoconfig.mail.provider.com/mail/config-v1.1.xml?emailaddress=info@example.com	HTTP GET
	or https://autoconfig.provider.com/mail/config-v1.1.xml?emailaddress=info@example.com	HTTP GET

Table XI: The deployment rate of auto-configuration among different domain lists.

Domain list	Domains # (100%)	Intersection[1]	MX Records	Autodiscover	Autoconfig	SRV Records	Support[2]
Tranco1M	1,000,000	1,000,000 (100.00%)	646,895 (64.69%)	47,917 (4.79%)	54,776 (5.48%)	11,060 (1.11%)	76,104 (7.61%)
Top100Provider	100	100 (100.00%)	100 (100.00%)	15 (15.00%)	18 (18.00%)	22 (22.00%)	30 (30.00%)
GovDomain	41,865	8,831 (21.10%)	19,882 (47.49%)	1,104 (2.64%)	973 (2.32%)	165 (0.40%)	1,446 (2.74%)
BankDomain	7,021	1,943 (27.70%)	5,488 (78.17%)	112 (1.60%)	90 (1.28%)	11 (0.16%)	160 (2.28%)
FreeDisposableProvider	16,752	1,380 (8.24%)	8,293 (49.50%)	612 (3.65%)	1,754 (10.47%)	296 (1.77%)	1,911 (11.41%)
Total	1,053,469	1,000,000 (94.92%)	670,709 (63.67%)	49,538 (4.70%)	57,331 (5.44%)	11,281 (1.07%)	79,212 (7.52%)

[1] Intersection refers to the size of the intersection of the domain list with the Tranco1M list.
[2] Support refers to how many domains support at least one auto-configuration mechanism.

Table XII: Evaluation results of Autodiscover and Autoconfig in A1.

(a) Number of Autodiscover-enabled domains returning configuration information per request method and the number of not well-established servers.

Domain List	Support	HTTP Return[1]	w/ HTTP Only[1]	HTTPS Return[2]	w/ HTTPS Only[2]	Not well-established[3]	w/o to-HTTPS redirection[3]	w/ to-HTTP redirection[3]
Tranco1M	47,917	38,456 (80.26%)	4,924 (10.28%)	42,993 (89.72%)	9,461 (19.74%)	28,905 (60.32%)	28,902 (60.32%)	38 (0.08%)
Top100Provider	15	14 (93.33%)	0 (0.00%)	15 (100.00%)	1 (6.67%)	7 (46.67%)	7 (46.67%)	0 (0.00%)
GovDomain	1,104	919 (83.24%)	37 (3.35%)	1,067 (96.65%)	185 (16.76%)	773 (70.02%)	773 (70.02%)	1 (0.09%)
BankDomain	112	89 (79.46%)	7 (6.25%)	105 (93.75%)	23 (20.54%)	69 (61.61%)	69 (61.61%)	0 (0.00%)
FreeDisposableProvider	612	552 (90.20%)	30 (4.90%)	582 (95.10%)	60 (9.80%)	185 (30.23%)	185 (30.23%)	0 (0.00%)

(b) Number of Autoconfig-enabled domains returning configuration information per request method and the number of not well-established servers.

Domain List	Support (From MX)	HTTP Return[1]	w/ HTTP Only[1]	HTTPS Return[2]	w/ HTTPS Only[2]	Not well-established[3]	w/o to-HTTPS redirection[3]	w/ to-HTTP redirection[3]
Tranco1M	54,776 (11,222)	43,238 (78.94%)	8,383 (15.30%)	46,393 (84.70%)	11,538 (21.06%)	36,464 (66.57%)	36,417 (66.48%)	380 (0.69%)
Top100Provider	18 (1)	17 (94.44%)	2 (11.11%)	16 (88.89%)	1 (5.56%)	7 (38.89%)	7 (38.89%)	0 (0.00%)
GovDomain	974 (121)	834 (85.63%)	58 (5.95%)	915 (93.94%)	139 (14.27%)	756 (77.62%)	756 (77.62%)	1 (0.10%)
BankDomain	90 (16)	73 (81.11%)	2 (2.22%)	88 (97.78%)	17 (18.89%)	64 (71.11%)	64 (71.11%)	1 (1.11%)
FreeDisposableProvider	1,754 (1,061)	681 (38.83%)	209 (11.92%)	1,545 (88.08%)	1,073 (61.17%)	555 (31.64%)	553 (31.53%)	10 (0.57%)

[1] HTTP return indicates that configuration information can be retrieved via an HTTP request. HTTP only indicates that configurations can only be retrieved through an HTTP request and not via HTTPS.
[2] HTTPS return indicates that configuration information can be retrieved via an HTTPS request. HTTPS only indicates that configurations can only be retrieved through an HTTPS request and not via HTTP.
[3] Not well-established indicates that servers do not redirect HTTP to HTTPS or include HTTP URLs in the redirection chain for HTTPS requests.