

Improving Internet Security through Empirical and Qualitative Studies of Email and DNS Ecosystem

Mohammad Ishtiaq Ashiq Khan

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Applications

Taejoong T. Chung, Chair
Bimal Viswanath
Bo Ji
Daphne Yao
Tobias Fiebig

Nov 14, 2025
Blacksburg, Virginia

Keywords: Email Security, Email Sender Authentication, Email Transport Security,
Measurement, Vulnerability Discovery, DNS Security etc.

Copyright 2025, Mohammad Ishtiaq Ashiq Khan

Improving Internet Security through Empirical and Qualitative Studies of Email and DNS Ecosystem

Mohammad Ishtiaq Ashiq Khan

(ABSTRACT)

Email and the Domain Name System (DNS) remain foundational pillars of Internet communication, yet their security mechanisms continue to suffer from subtle design limitations, operational misconfigurations, and systemic fragility. This dissertation presents an empirical, measurement-driven exploration of the global email and DNS security landscape, identifying recurring patterns of misconfiguration, evaluating the real-world efficacy of recent protocol defenses, and proposing practical tools to enhance their resilience.

First, we examine email sender authentication, focusing on the large-scale deployment and operational correctness of Sender Policy Framework (SPF) and Domain-based Message Authentication, Reporting, and Conformance (DMARC). Drawing from over 12 months of longitudinal data encompassing 176 million domains, our study exposes widespread evaluation inconsistencies and misconfigurations that undermine authentication integrity and email deliverability. We further uncover novel attack vectors, including exploitable DNS amplification pathways within major email providers and open-source SPF validators, emphasizing the systemic risk of these seemingly mature defenses.

Next, we turn to email transport security, analyzing the adoption and robustness of the SMTP Mail Transfer Agent Strict Transport Security (MTA-STS) protocol introduced by major providers such as Google and Microsoft. Despite its reliance on the well-established web PKI ecosystem, we find that 28% of MTA-STS enabled domains exhibit configuration flaws that nullify the intended transport-layer protections, underscoring the practical challenges of achieving secure email delivery even under modern standards.

Finally, we investigate DNS security through the lens of DNSSEC deployment. Leveraging over 1M diagnostic records from DNSViz, we systematically classify the most frequent DNSSEC configuration errors, explore their persistence over time, and trace their operational root causes. To address these challenges, we introduce DFixer, an automated offline repair tool that aggregates cascaded error codes into root causes and generates both high-level remediation guidance and corresponding BIND command sequences. Experimental evaluation with a purpose-built erroneous zone replicator demonstrates that DFixer can automatically repair 99.99% of observed DNSSEC errors within seconds. Together, these studies reveal the gap between the theoretical robustness of Internet security protocols and their practical deployment realities. By combining large-scale empirical measurement, vulnerability analysis, and automated remediation, this dissertation advances our understanding of Internet infrastructure security and provides actionable paths toward more reliable, verifiable, and self-healing email and DNS ecosystems.

Improving Internet Security through Empirical and Qualitative Studies of Email and DNS Ecosystem

Mohammad Ishtiaq Ashiq Khan

(GENERAL AUDIENCE ABSTRACT)

Email and the Domain Name System (DNS) are essential to how the Internet works, but the security systems that protect them often fall short in practice. Even when standards exist, real-world mistakes, outdated setups, and overlooked weaknesses can leave users and organizations exposed to attacks. This dissertation takes a data-driven look at how well these security mechanisms actually work across the Internet. By studying hundreds of millions of domains over more than a year, it reveals that many email authentication systems that are meant to prevent spoofing and scams, are set up incorrectly and suffer from ambiguity in their standards. These problems not only weaken their security but also exposes new vulnerabilities that attackers could take advantage of.

The work then examines a relatively newer protocol designed to secure email transmission between servers. Although this technology relies on the same security foundations used by modern websites, more than a quarter of the domains that enable it configure it incorrectly, leaving the protections ineffective. Finally, the dissertation studies DNSSEC, a protocol intended to prevent attackers from tampering with DNS responses. By analyzing over a million diagnostic records, it identifies the most common configuration errors and explains why they occur. To help fix these problems, the research introduces an automated tool that can diagnose the root causes of DNSSEC errors and generate step-by-step instructions to repair them. In testing, the tool successfully fixed nearly every observed error in just seconds.

Overall, this work shows that the biggest threats to Internet security often come not from the protocols themselves, but from how they are deployed in practice. By combining large-scale measurement, vulnerability analysis, and automated repair techniques, this dissertation offers practical steps toward making Internet more reliable, secure, and resilient for everyone.

Dedication

To my parents

Acknowledgments

I am incredibly indebted to my advisor Dr. Tijay Chung for his unwavering support and guidance throughout this research. His insightful feedback, expertise and constant encouragement were instrumental in shaping this thesis. I would also like to extend my sincere thanks to Dr. Tobias Fiebig for generously sharing his expertise and knowledge. I am very grateful to the other committee members Dr. Bimal Viswanath, Dr. Bo Ji, and Dr. Daphne Yao for their guidance, support, and constructive comments. I would like to thank all my previous and current colleagues at NetSecLab as well as all my collaborators, with special appreciation to Dr. Casey Deccio, Dr. Olivier Hureau, Protick Bhowmick, and Weitong Li for their valuable discussions, contributions, and feedback that greatly impacted my research. I am also grateful to the anonymous reviewers and shepherds for their helpful comments during the peer-review process, as well as to the funding agencies whose support made this dissertation possible. Finally, I am thankful to my parents, family, and friends for their unwavering encouragement in my academic journey.

Contents

List of Figures	xii
List of Tables	xvii
1 Introduction	1
1.1 Motivation	1
1.2 Problem Rationale and Research Questions	2
1.2.1 Rationale for Problem Selection	2
1.2.2 Thesis-Wide Methodological Framework	2
1.2.3 High-level Research Questions	2
1.3 Research Contributions	3
1.4 Dissertation Organization	4
2 Background	5
2.1 SMTP	5
2.2 Sender Authentication	6
2.2.1 SPF	6
2.2.2 DKIM	8
2.2.3 DMARC	9
2.2.4 Reporting mechanisms	10
2.3 Transport Security	12
2.3.1 STARTTLS	12
2.3.2 DANE	12
2.3.3 MTA-STS	14
2.3.4 MTA-STS Policies	16
2.3.5 Policy Validation	16

2.3.6	Policy Delegation	18
2.3.7	Removing MTA-STS	18
2.4	DNS and DNSSEC	18
2.4.1	DNSSEC States	19
2.4.2	DNSViz	20
3	Related Work	22
3.1	Email Sender Authentication	22
3.1.1	SPF Ecosystem	22
3.1.2	DMARC Ecosystem	24
3.2	Email Transport Security	24
3.2.1	SMTP encryption	25
3.2.2	Ecosystem of DANE	25
3.2.3	MTA-STS	25
3.3	DNSSEC	26
4	Management and Operational Challenges of SPF	27
4.1	SPF Deployment for Outbound Emails	28
4.1.1	Datasets	28
4.1.2	SPF Record Prevalence	29
4.1.3	SPF Records Management	30
4.2	Superfluous SPF Records	32
4.2.1	Methodology	34
4.2.2	Domains with Superfluous SPF Records	35
4.2.3	Why Superfluous Records Exist	36
4.2.4	Case study: SPF Flattening	36
4.3	SPF for Inbound Emails	37
4.3.1	Methodology	38
4.3.2	Results	40

4.3.3	Threat Model	41
4.3.4	DoS Attack for Inbound Emails	41
4.3.5	Responsible Disclosure	43
4.4	Discussion	43
4.4.1	Recommendations	44
4.4.2	Limitations	44
5	Measurement and Security Implications of DMARC Reporting	46
5.1	DMARC Deployment for Outbound Emails	47
5.1.1	Methodology	47
5.1.2	Prevalence	48
5.1.3	DMARC Reporting Compliance	50
5.2	DMARC Reporting Deployment for Inbound Emails	51
5.2.1	Methodology	52
5.2.2	Characterization	53
5.2.3	Experiment Results	54
5.3	Report Reflection (RR) Attack	55
5.3.1	Attack 1: Inbox flooding	57
5.3.2	Attack 2: SMTP connection flooding	60
5.3.3	Summary	61
6	Unraveling the Challenges of MTA-STS Deployment	63
6.1	Measuring MTA-STS Deployment	64
6.1.1	Datasets	64
6.1.2	MTA-STS Deployment	65
6.2	MTA-STS Management	66
6.2.1	Datasets	66
6.2.2	MTA-STS Validation	67
6.2.3	Misconfigurations: Individual Components	68

6.2.4	Misconfigurations: Inconsistency Errors	74
6.2.5	Inconsistency From Multiple Third Parties	77
6.2.6	Key Takeaways	78
6.2.7	Responsible Disclosure	79
6.3	MTA-STS Policy Delegation	79
6.4	Sender-Side MTA-STS Validation	80
6.4.1	Sender Side Dataset	81
6.4.2	MTA-STS Validation	81
7	Email Security in Practice	82
7.1	Operational Practice for SPF and DMARC	83
7.1.1	Survey Methodology	83
7.1.2	Survey Findings	83
7.2	MTA-STS in Practice	85
7.2.1	Survey Methodology	86
7.2.2	Survey Findings	86
8	Understanding DNSSEC Debugging Pattern	89
8.1	Dataset	90
8.2	Validation and Category Definitions	91
8.3	How Useful Is DNSViz?	93
8.4	Exploring Negative Transitions	95
8.5	Error Prevalence in DNSSEC	97
8.6	Understanding Error Resolution Patterns	98
9	DFixer: A Comprehensive DNSSEC Fixing Framework	102
9.1	Rationale and Key Observations	102
9.2	Why Existing Tools and Naive LLMs Fall Short	102
9.3	Design and Workflow of DFixer	104
9.4	System Model and Assumptions	105

9.5	ZReplicator: Replicating Real Errors in a Local Environment	105
9.6	DFixer: Evaluation	106
9.6.1	Evaluation Pipeline	106
9.6.2	Evaluation Metrics	108
9.6.3	Implementation and Environment:	109
9.6.4	Experiment Results	109
9.6.5	Limitations of ZReplicator and DFixer	111
9.6.6	Testing Extensibility	112
10	Concluding Discussion	114
	Bibliography	116
	Appendix A Copyright Statement	126
	Appendix B Ethical Considerations	127
B.1	Data Collection	127
B.2	Survey with Email Operators	127
B.3	SMTP Server Scanning Experiment	127
B.4	Disclosure Emails for MTA-STS Misconfigurations	128
B.5	ZReplicator and DNSViz	128
	Appendix C SMTP TLS Reporting	129
	Appendix D Email Sender Authentication Survey	131
D.1	Questionnaire	131
	Appendix E MTA-STS Survey	134
E.1	Questionnaire	134
	Appendix F Sample DFixer Output	139

List of Figures

2.1	An overview of SPF validation flow at various stages of the SMTP transaction. Upon receipt of the HELO/EHLO command, either the MTA natively or through Milters can initiate a reverse DNS lookup for the incoming IP address. The standard [76] requires that SPF validation must be performed on the MAIL FROM identity, but also suggests doing it on the HELO/EHLO as well. So, after the MAIL command, sender authentication can be executed at any point during the SMTP connection by milters or policy servers. Notably, SPF validation can even occur post-queue in the content filter stage (e.g., SpamAssassin [21]).	8
2.2	Overview of how DMARC reports can be delivered through SMTP along with DNS; r.com fetches DMARC records from a.com and sends report to the email addresses listed in the rua tag. When it finds an external domain in the rua tag (c.com), it should fetch a DMARC authorization record to confirm the report delegation.	9
2.3	DANE management cases are classified depending on who manages the SMTP server and the name server. (a) SMTP server is outsourced (SO), (b) SMTP server is self-managed but name server is outsourced (SSDO), and (c) both SMTP server and name server are self-managed (SSDS). Note that the name server that serves the TLSA record is also outsourced if the SMTP server is outsourced.	13
2.4	Flowchart of MTA-STS validation from an SMTP client: The client checks for MTA-STS support by looking up the MTA-STS DNS record and resolves the policy server’s IP address ①; The client fetches the policy file through HTTPS, and matches the MX host against the mx patterns specified in the policy ②; The client connects to the MX host using STARTTLS, and validates the recipient MTA’s certificate ③.	17
4.1	The percentage of domains with SPF records is increasing; the spike on March 15, 2023 for .se domains is because around 100K domains using loopia.se as their email hosting provider added a SPF record to their domain.	29
4.2	More than 9.2% of the SPF records with include requires more than 10 DNS look ups. Note that the x axis extends to 175!	31
4.3	As of March 27th, 2023, only six SPF records appear in the include mechanisms for 50% of all SPF records. Remarkably, just two SPF records managed by two specific hosting providers account for 83.7% of the SPF records that require more than 10 DNS lookups for evaluation.	33

4.4	Removing superfluous SPF records can effectively reduce the number of required DNS lookups.	35
4.5	The percentage of domains with superfluous SPF records from our latest snapshot that match with historical MX records.	37
4.6	CDF of the number of DNS requests that our DNS authoritative server receives; 40% servers violate the total lookup limit recommendation in practice. Note that x axis extends to 25,117! ¹	38
4.7	Certain SPF validators (e.g., iRedAPD or libspf2), do not support parallel SPF validation. In such cases, attackers can introduce delays up to either T_{lim} or $t_{lim} \times N_{lim}$ seconds; Each MTA has its own timeout limit for SPF validation, denoted as M_{lim} . Should the delay exceed this threshold, subsequent incoming emails may be rejected.	42
5.1	The deployment rate of DMARC records and their reporting feature from the dataset. More than 63% of DMARC records with reporting features forwards the report to external domains across the four TLDs.	48
5.2	Only 7 domains receive the 43% of DMARC reports on behalf of domain owners.	49
5.3	The percentage of domains with DMARC records, and their percentages with reporting feature as a function of domain popularity.	50
5.4	Less popular domains tend to be misconfigured when their DMARC records are configured to forward DMARC reports to external domains.	51
5.5	Attack exploiting SMTP servers that do not perform EDV correctly.	56
5.6	Attack exploiting SMTP servers that support both DMARC and TLS-RPT reporting. The attacker intentionally provides an <i>invalid</i> certificate to reflectors when they send a report to the attacker. This triggers them to send a report to the victim as specified in the rua tag of the TLS-RPT record.	58
5.7	Attack exploiting the rua tag override; the reflector finds a rua tag, which contains a list of external domains. For each domain, it performs EDV and fetches the DMARC authorization record, which also contains a list of email addresses in the rua tag; a TLS-RPT report will be sent to each address.	59
5.8	Attack exploiting the rua tag override. Since the TLS-RPT is not supported by the reflector, the attacker can host a wildcard MX record that maps to the victim's MX record, which makes the reflector initiate n^2 SMTP connections.	61
6.1	The deployment of MTA-STS records; 461 new domains related to the same organization under .org adopted MTA-STS in Jan 2, 2024 causing the notable spike.	66

6.2	The percentages of domains publishing both MX and MTA-STS records as a function of website popularity based on Tranco 1M [99]; more popular websites are more inclined to deploy MTA-STS for their email services.	67
6.3	Percentage of MTA-STS enabled domains with errors in (1) MTA-STS records, (2) policy retrieval over HTTPS, (3) certificate of MX hosts, or (4) policy text where mx patterns of the policy file and MX records are inconsistent. Since August 2024, newly registered domains under Porkbun LLC [100] with invalid policy host certificates have increased policy server errors, affecting 7,237 domains in our latest snapshot.	68
6.4	Percentage of MTA-STS enabled domains with misconfigured policy servers by error type and managing entity. In our latest snapshot, 9,588 (37.8%) self-managed and 1,393 (4.9%) third-party policy servers had misconfigurations. A spike on June 8, 2024, for third-party servers was due to a leading provider issuing self-signed certificates for 1,385 domains. Late spikes for self-managed domains were caused by Porkbun domains, as detailed in Figure 6.3.	71
6.5	Percentage of MTA-STS enabled domains with PKIX invalid certificate for MX hosts; in our latest snapshot, 1,046 (4.4%) domains that self-manage their MX hosts and 397 (1%) domains that use email hosting services present PKIX-invalid certificates. 270 domains with self-hosted MX host fixed their Common Name mismatch error in our latest snapshot.	73
6.6	Percentage of MTA-STS enabled domains with all invalid, and partially invalid MX hosts. "enforce" mode presents domains with stricter policy and at least 1 invalid MX host; these are subject to potential email delivery failure from MTA-STS compliant senders and in our latest snapshot, we found 269 such domains.	74
6.7	Percentage of MTA-STS enabled domains with mismatches between mx patterns listed in the policy file and actual MX records due to mismatches in TLD, SLD, or extra subdomains, or typos. "enforce" mode presents mismatched domains with stricter policy; these are subject to email delivery failure from MTA-STS compliant senders and in our latest snapshot, we found 406 such domains.	75
6.8	The percentage of domains with mismatched mx patterns in their MTA-STS policy file that can be correctly matched with historical MX records.	76
6.9	Inconsistency in domains outsourcing both policy servers and email hosting, based on whether the same provider manages both.	77
7.1	The figure shows the distribution of the number of email accounts managed by each of the 62 respondents who answered both questions regarding SPF deployment and SPF validation support; note that all respondents serving as SMTP administrators confirmed the deployment of SPF records.	82

7.2	The distribution of the number of email accounts managed by each of the 92 respondents who answered the question regarding MTA-STS deployment.	87
8.1	Percentage of domains from our DNSViz dataset appearing in each Tranco 1M bin [99], including the share of signed and misconfigured domains.	92
8.2	For domains in the CD category, comparing DNSSEC status between the first and last snapshots recorded by DNSViz. Note the moderate but significant fraction of domains that either enable or disable DNSSEC over time.	94
8.3	Percentage of DNSSEC error categories out of 747,455 snapshots for second-level and lower domains. NSEC(3)-related errors, including nonzero iteration counts and missing negative proofs, account for a significant fraction. We also see many delegation failures and signature (RRSIG) issues.	97
8.4	Resolution times for critical (red boxes) vs. non-critical (orange boxes) DNSSEC errors. The numbers on the x-axis (19) are indices that map directly to the numbered error categories detailed in Table 8.3. Note that the median time to deploy DNSSEC (black box) is more than a day; this is likely attributed to the operational challenges in DNSSEC identified by prior works [34].	100
8.5	65% of the domains have a median time difference of less than a day between their consecutive two snapshots	101
9.1	Overview of the DFixer pipeline. Each iteration collects DNSSEC data with probe and grok filters relevant error codes, resolves root causes via DResolver, and produces fix commands. The process repeats until no blocking errors remain.	103
9.2	High-level illustration of our evaluation pipeline. Each JSON file from the DNSViz dataset is parsed, replicated locally via ZReplicator, and then processed by DFixer.	107
C.1	The deployment of TLS-RPT records; top graph shows the % of domains with MX records having TLS-RPT records; bottom graph shows the % of domains with MTA-STS records having TLS-RPT records. At December 21, 82 .se domains revoked their TLS-RPT records causing the downspike in the top graph. Additionally, from Jun' 24 to Aug '24, 1,411 .net domains added TLS-RPT records; only 198 of them had MTA-STS records which explains the downward spike around that time in the bottom graph.	130
F.1	Sample remediation workflow produced by APM for a domain with its only KSK revoked and linked to a DS record	140
G.1	Prompt format used for root cause identification	142

G.2	Prompt formats used for devising remediation plan with commands	143
G.3	Response from GPT-4o when passed the prompt in Figure G.1 for our invalid setup	144
G.4	Response from GPT-4o when passed the prompt in Figure G.2 for our invalid setup	145
G.5	A hallucination example from GPT-4o when passed the DNSViz visualization for our invalid setup	146

List of Tables

4.1	Overview of the datasets. The number and percentage of the domains that have SPF records are as-of March 27, 2023.	28
4.2	Percentage change (%) in the usage of each mechanism between our initial snapshot on October 13, 2021, and our final snapshot on March 27, 2023; only include and all mechanisms have increased across the TLDs, as indicated by a green background.	30
4.3	The most popular reasons and their examples of SPF record errors are shown. . . .	30
4.4	The table lists the top 10 popular MX records, inferred SPF records, and their occurrence probabilities. Additionally, it indicates (1) if the IP addresses in the SPF records have been updated, (2) the number of domains that have flattened their SPF records, and (3) the domains with stale SPF records despite updates from their hosting provider.	33
4.5	The lists 10 popular SPF libraries (Lib), milsters (M), policy servers (PS), and content filters (CF). Here, N_{lim} and V_{lim} represent the limits on DNS and void lookups, respectively; for example, Mail::SPF::Query and iRedAPD have no N_{lim} , resulting in endless DNS requests (highlighted in red). t_{lim} and T_{lim} indicate DNS timeout values for individual and total SPF evaluations, respectively. When parallel SPF validation is not supported, the SMTP server is vulnerable to DoS attack for inbound emails (highlighted in yellow).	39
5.1	Overview of the datasets for this study. The number and percentage of the domains that have DMARC records are as-of January 8, 2023. 49% of DMARC records list email addresses to receive the DMARC reports. 70% of them use external domains to forward the reports across the four TLDs.	47
5.2	The configurations of DMARC records for experiments to characterize DMARC report senders	52
5.3	Table showing the 7 popular email hosting providers and two softwares that support DMARC reporting. If email providers do not implement EDV, we do not test whether they override EDV or not (hence the –). In case of Gmail, it only sends one report to the first recipient listed in the rua tag.	56
5.4	Table showing the two email hosting providers (EHPs) support TLS-RPT reporting; the other providers (Amazon, FastMail, Yahoo, 163, QQ, and Gmail) do not support it. We could not find any opensource software for TLS-RPT.	56

6.1	Overview of our datasets; the number of the domains with MTA–STS records are as-of September 29th, 2024.	63
6.2	The list includes the top eight policy hosting providers, along with their hosting policies for customers who have opted out. The number of domains is based on our most recent data snapshot from September 29, 2024.	78
8.1	Overview of the DNSViz dataset. It contains 1.1M total snapshots spanning from 2020-03-11 to 2024-09-25. Second-level and lower (SLD+) domains are determined using the Public Suffix List (PSL). CD and SD domains are defined later in section 8.2.	90
8.2	Causes of negative transitions from a valid (sv) DNSSEC state to either bogus (sb) or insecure (is). Key rollovers and algorithm changes together account for roughly two-thirds of sv → sb transitions, while a smaller fraction stems from nameserver (NS) updates. We observe a similar pattern for sv → is transitions as well.	95
8.3	Prevalence of various DNSSEC error types in our DNSViz dataset, covering 319,277 second-level and their lower-level domains (total 747,455 snapshots); for example, the “Nonzero Iteration Count” in NSEC3 appears in 215,036 snapshots (28.8%) spanning 62,870 domains (19.7%). The markers (①–⑨) highlight notable or especially frequent misconfigurations in each category, which are also presented in Figure 8.4. Last row represents the # of snapshots (and domains) with at least 1 error from above categories.	96
8.4	Adjacency matrix of state transitions. Each cell shows # of transitions (top) and median time in hours (bottom). A dash (–) means no observed transition. We can observe administrators reacting promptly when their domain becomes bogus.	99
8.5	For each DNSSEC state of interest, we list (i) the total number of domains that appear in that state at least once, and (ii) how many never rectified the issue or re-enabled DNSSEC according to their latest snapshot in our dataset.	100
9.1	Performance of ZReplicator and DFixer among the snapshots with IE ≠ ∅.	106
9.2	Instructions issued by DFixer during its iterative remediation process in the S2 subset (i.e., zones with more complex DNSSEC errors). This highlights how DFixer repeatedly removes incorrect DS records, re-signs zones, or updates keys until all misconfigurations are resolved.	110

Chapter 1

Introduction

1.1 Motivation

Email, often referred to as electronic correspondence, has been a dominant mode of global communication since its advent in the 1970s. Although numerous alternative communication platforms have emerged, the ubiquity and interoperability of email continue to make it indispensable for personal, organizational, and inter-domain communication. However, despite its widespread use, email remains susceptible to significant security shortcomings. This is largely because its foundational protocol, the Simple Mail Transfer Protocol (SMTP), was designed in an era with different security norms and lacks inherent mechanisms for verifying sender authenticity or ensuring transport confidentiality. Consequently, email remains vulnerable to a variety of threats, such as spoofing attacks and STARTTLS downgrade attacks [30, 38, 43, 64].

To mitigate spoofing threats, a number of sender authentication mechanisms have been introduced, such as SPF [76], DKIM [27], DMARC [74], BIMI [26]; these allow domain owners to specify authorized senders and dictate receiver actions upon authentication failure. Similarly, to secure the communication channel, protocols such as DANE [41] and MTA-STS [85] have been proposed to enforce encrypted SMTP connections. Notably, all of these mechanisms rely on the Domain Name System (DNS) as a trust and policy distribution layer. Consequently, their effectiveness is directly tied to the security and reliability of DNS itself.

However, DNS, while foundational to the modern Internet, is not immune to its own set of security and operational challenges. The introduction of DNS Security Extensions (DNSSEC) [11, 12, 13] aimed to provide authentication and integrity to DNS responses, yet its adoption has remained limited and misconfigurations are widespread. These DNS-level weaknesses can propagate upward, undermining the security guarantees of higher-layer protocols that depend on DNS-based policies and authentication records.

Given this interdependence, it is critical to holistically examine both the email security protocols built atop DNS and the security posture of DNS itself. The goal of this dissertation is therefore to thoroughly understand the operational complexities and vulnerabilities present across this ecosystem, identify systemic weaknesses that threaten the reliability of global email and DNS infrastructure, and propose data-driven, automated avenues for improvement.

1.2 Problem Rationale and Research Questions

In this section, we define the rationale behind our problem selection, outline the shared methodological framework, and explain the questions we seek to answer and the approach we use to address them.

1.2.1 Rationale for Problem Selection

We structure this dissertation around four closely related problems, chosen to reflect both their practical impact and gaps in current research. We begin with two email sender authentication protocols, SPF and DMARC, selected because they are widely deployed, underexplored at Internet scale, and known to be vulnerable to subtle but serious misconfigurations and attacks. We then turn to the email transport security protocol MTA-STS, a relatively recent standard for which no prior large-scale empirical study existed at the time of our work. Finally, we analyze the security of DNSSEC, the foundational protocol on which these and many other mechanisms rely. We focus on DNSSEC because one of the main barriers to its wider adoption is its operational complexity and tendency toward misconfiguration, motivating our design of an automated repair tool to provide fast, practical remediation.

1.2.2 Thesis-Wide Methodological Framework

Across all components in this thesis, we follow a common five-step methodology. We begin with large-scale data collection, followed by systematic data analysis. We then conduct empirical vulnerability analysis, complemented by qualitative studies to understand the real-world operational landscape. Finally, where feasible, we design and evaluate automated tools that operationalize our findings into practical remediation.

1.2.3 High-level Research Questions

At a very high-level, we want to answer the following research questions.

- RQ1.** How do Internet-scale deployment and operational practices affect the security, reliability, and effectiveness of email and DNS security protocols in practice?
- RQ2.** What recurring patterns of misconfiguration, ambiguity, and implementation inconsistency arise in email security stack, and how do these patterns undermine the theoretical guarantees of the protocols?

RQ3. How can our findings make the email security configuration standards more robust and to what extent can automated repair tools help bridge the gap between protocol design and real-world deployment?

While these questions are not addressed in a strictly one-question-per-chapter manner, this dissertation consists of four interrelated components: analysis of SPF, DMARC, MTA-STS, and DNSSEC protocols. Each component makes specific contributions and collectively they provide answers to the research questions introduced before. For clarity, each contribution listed in this dissertation is directly linked to the research question(s) it resolves.

1.3 Research Contributions

Following are our high-level research contributions:

Longitudinal Measurement of the Email and DNS Ecosystem (RQ1, RQ2) We provide a comprehensive view of the state of email and DNS security through large-scale analysis.

SPF: Conducted a longitudinal analysis revealing the persistence of misconfigurations over time.

DMARC Reporting: Conducted the first comprehensive study of DMARC Reporting landscape.

MTA-STS: Developed the first comprehensive taxonomy of misconfigurations, explaining the root causes of deployment failures.

DNSSEC: Characterized the debugging behaviors of administrators to understand why chain-of-trust errors persist.

Identification of Novel Attack Vectors (RQ1, RQ2) We move beyond simple measurement to demonstrate how operational flaws can be weaponized.

SPF: Identified the presence of exploitable SMTP servers in the wild and outlined specific attack schemes exploiting implementation weaknesses.

DMARC: Exposed critical security gaps in reporting mechanisms and demonstrated attacks leveraging ambiguities and misconfigurations in the protocol standard.

Human-Centric Analysis of Operational Reality (RQ1) Recognizing that protocols are managed by people, we integrated qualitative research to contextualize technical findings.

Surveys: Conducted surveys with practitioners regarding SPF and DMARC to map the “practical landscape” and understand the rationale behind common misconfigurations.

Operational Study: Analyzed the constraints of email operators and policy providers to reveal the friction points in adopting MTA-STS.

Remediation, Tooling, and Recommendations (RQ3) We conclude by transitioning from problem identification to concrete solutions.

Protocol Hardening: Offered specific recommendations to update and harden SPF and DMARC standards.

Automated Defense: Proposed and evaluated a semi-automated tool for fixing DNSSEC errors, alongside an automated framework for validating these fixes, directly addressing the root-cause infrastructure issues.

1.4 Dissertation Organization

Chapter 2 establishes the background knowledge that underpin the research. We then explore the existing research landscape in Chapter 3. Chapters 4 delve into a comprehensive quantitative analysis of the SPF protocol, examining its effectiveness for both inbound and outbound emails. Chapter 5 investigates the DMARC protocol using similar quantitative methods, introduces a novel attack scheme involving DMARC reporting to highlight potential vulnerabilities. Chapter 6 shifts gears, focusing on a deep exploration of MTA-STS management complexities. Chapter 7 presents a qualitative analysis of email security practices. Finally, from Chapter 8 onwards, focus turns to understanding DNSSEC errors and introduction of an automated remediation mechanism. The concluding chapter brings everything together, summarizing the key findings, discussing broader implications, and proposing potential avenues for future research.

Chapter 2

Background

2.1 SMTP

Simple Mail Transfer Protocol (SMTP) is a foundational protocol employed for email communication over the Internet. The email process commences with the user composing an email in their Mail User Agent (MUA). This email is then transmitted to the sender's Mail Transfer Agent (MTA) either via SMTP or HTTP. The sending MTA subsequently queries the recipient MTA's address through the Domain Name System (DNS), establishes a Transmission Control Protocol (TCP) connection with the target host, and conveys the email using SMTP. The process involves the execution of several SMTP commands to the recipient MTA, which include:

- HELO/EHLO commences communication with the server. They serve as greeting messages, wherein the sending server usually declares its domain. The EHLO command is an extended version that enables the client to request specific features that are supported by the server.
- MAIL FROM specifies the sender's email address, thereby informing the server about the origin of the email. It also sets the return path for delivery status notifications.
- RCPT TO indicates the recipient's email address. The server acknowledges each valid recipient with a positive response.
- DATA describes the actual email content. This command is followed by the email message, including headers and body. In the email message header, the sender can also specify one or more *From* address(es), which is generally shown as the from address in the display of the recipient's MUA [105]. We refer to this *From* as Header From.¹

Subsequently, the Mail Delivery Agent (MDA) delivers the email to the recipient using HTTP, IMAP, or POP3 protocols [75].

¹In case more than one *From* address is specified (multiple authors), another field, *Sender*, must be there in the message which designates the actual sender.

2.2 Sender Authentication

SMTP has no built-in security mechanisms; theoretically, an attacker can manipulate the address in the MAIL command and in the Header From field to spoof the sender domain [74, 115]. To mitigate these attacks, various security extensions have been proposed.

2.2.1 SPF

With SPF, domain owners can publish a policy with which a receiving mail server can assess whether a sending IPv4 or IPv6 address may deliver a message for a specific user or domain. This is done via a DNS TXT record at the sending domain. Policies are described in a TXT record (the previously used SPF DNS resource record (RR) has been discontinued [76]) using mechanisms with qualifiers and modifiers and SPF verifiers check policy of HELO identity (recommended) and MAIL FROM identity (must) against the connecting IP address. Furthermore, SPF supports the use of macros expanded based on characteristics of the message.

Mechanisms Mechanisms serve as the main method of defining policy; In total, eight mechanisms are defined: A, ip4, ip6, MX, PTR (discouraged as per RFC [76]), exists, include, and all. ip4 and ip6 allow specifying permitted sender addresses or networks for a domain. For example, the following configuration authorizes emails only from the IP address 10.0.0.1, while rejecting all others.

```
example.com    TXT    "v=spf1 ip4:10.0.0.1 -all"
```

In addition, MX allows a domains' MXes, PTR allows all clients whose reverse DNS name is in the domain (easily forged and therefore discouraged), A if a given FQDN resolves to a connecting IP and exists if a given domain exists, which is mostly useful with macros. Finally, the all record sets a default policy for all addresses not matched by an earlier mechanism.

In our SPF research, we primarily focus on the include mechanism. This mechanism allows referring to SPF records in the TXT RR of another FQDN. This is especially useful when domain name owners outsource their email service to a third-party provider, e.g., Google. A domain owner can then simply include `_spf.google.com` managed by Google to ensure SPF is appropriately configured.

```
example.com    MX      aspmx.l.google.com
example.com    TXT      "v=spf1 include:_spf.google.com"
```

This mechanism is also useful if domain owners may require other third-party services for specialized tasks; for instance, a domain owner using QQ.com for email hosting may also wish to utilize a bulk email service like Mailchimp or SendGrid. Hence, email hosting providers like Google

Workspace [18] and QQ.com [70] publish their SPF records to be included by domain owners through the `include` mechanism, and often explicitly document that users should do this.

Qualifiers Qualifiers specify the disposition if a mechanism matches. Absence of a qualifier defaults to `+` (pass).

- `+` (pass): Positive disposition.
- `-` (fail): Negative disposition. Commonly used with the `all` mechanism to reject all senders not explicitly listed.
- `?` (neutral): No disposition.
- `~` (softfail): a negative disposition but suggest to not enforce it (i.e., reject a message).

Modifiers Currently, the `exp=` and the `redirect=` modifier are specified in RFC4408 [124]. `exp=` should point to an FQDN whose TXT record explains the reason for a negative disposition. `redirect=` works similar to `include`, even though subtly different. As `redirect=` is rarely used in practice (apart from `gmail.com`), we focus in `include` in this work.

SPF Errors and Security Considerations SPF relies on DNS records for its operation. Evaluation can fail if the SPF records are inaccessible or malformed. Generally, two types of errors can arise during the evaluation of an SPF record, often resulting in email rejection:

- `temperrors` occur when the SPF validator encounters a transient issue, such as a DNS timeout. The validator may retry the operation later.
- `permerrors` occur when the SPF validator successfully retrieves a DNS record but fails to interpret it correctly.

A classic example that triggers a *permerror* is when the number of required DNS queries to evaluate an SPF record exceeds the standard limit of 10. This usually happens when multiple mechanisms refer to other DNS records, such as through `include` mechanisms. The standard imposes this limit to alleviate undue stress on DNS infrastructure.² In situations where the referenced SPF record does not resolve due to an empty DNS response or errors such as `NXDOMAIN` or `SERVFAIL`, this condition is termed a *void lookup*. The RFC additionally limits the number of void lookups to two.

²It is worth highlighting that this is distinct from `CNAME` records, which map one domain name to another in DNS. `CNAME` expansion is usually the responsibility of DNS resolvers, while `include` mechanisms in SPF are expanded by the SPF validator.

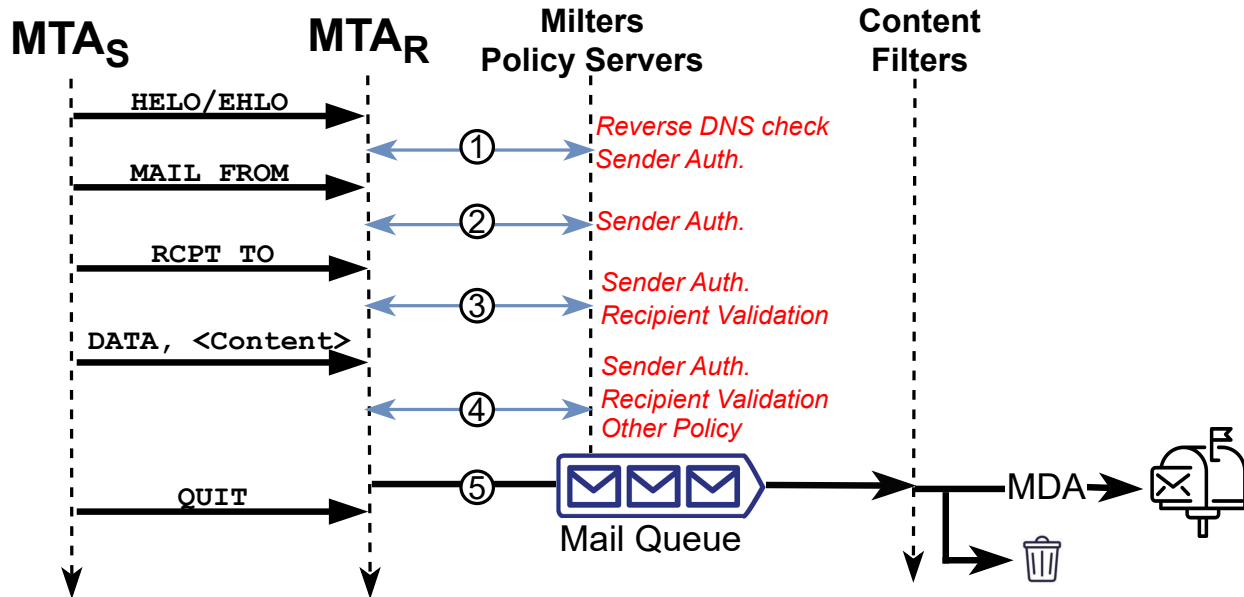


Figure 2.1: An overview of SPF validation flow at various stages of the SMTP transaction. Upon receipt of the HELO/EHLO command, either the MTA natively or through Milers can initiate a reverse DNS lookup for the incoming IP address. The standard [76] requires that SPF validation must be performed on the MAIL FROM identity, but also suggests doing it on the HELO/EHLO as well. So, after the MAIL command, sender authentication can be executed at any point during the SMTP connection by milers or policy servers. Notably, SPF validation can even occur post-queue in the content filter stage (e.g., SpamAssassin [21]).

SPF Validator Ecosystem SPF validation occurs either natively within the recipient MTA using SPF libraries (e.g., libspf2 [136]) or via external extensions, which can fall into three categories based on the timing of the SPF check.

- Before Queue Milter or Milter, in short, is a flexible framework for pre-queue email filtering and modification. Milers (e.g., milter-greylist [137]) can reject emails *during the SMTP transaction*.
- Policy Server (e.g., iRedAPD [135]) makes routing decisions based on predefined policies, such as rejecting an email if SPF fails.
- Post-Queue Content Filter (e.g., SpamAssassin [21]) operates after the MTA queues the mail, focusing on content-based attributes like message body and attachments.

2.2.2 DKIM

DomainKeys Identified Mail (DKIM) allows a email receiver to verify the integrity of the message; an email sender can include a digital signature in the email header signed by its private key [27]. The email receiver can verify the signature by fetching the sender's corresponding public key

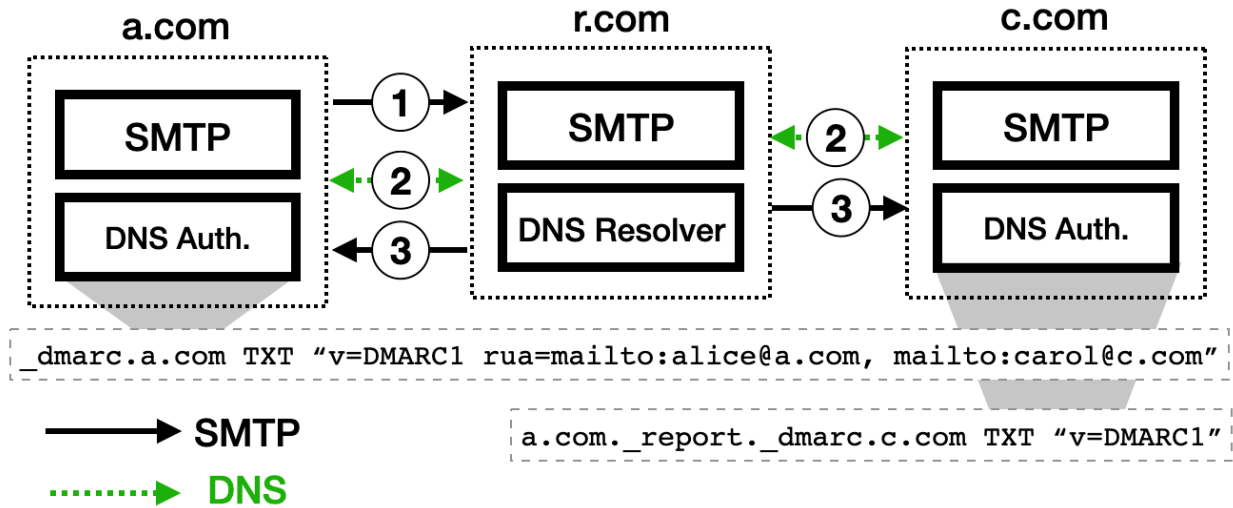


Figure 2.2: Overview of how DMARC reports can be delivered through SMTP along with DNS; `r.com` fetches DMARC records from `a.com` and sends report to the email addresses listed in the `rua` tag. When it finds an external domain in the `rua` tag (`c.com`), it should fetch a DMARC authorization record to confirm the report delegation.

(TXT records) from DNS. Goodrich et al. [60] proposed Accredited DomainKeys framework which extends DKIM by introducing a DNS-based accreditation service that attaches trust seals to verified domains, thereby strengthening domain-level email authentication.

2.2.3 DMARC

Even though SPF and DKIM provide an additional security layer to provide the authenticity of the email sender, it does not tell what actions the receiver has to take when validation fails. Thus, Domain-based Message Authentication Reporting and Conformance (DMARC) [74] was proposed to allow domain owners to publish a policy that tells the receiver to follow a certain action when SPF or DKIM validation fails. The domain name owner can publish its DMARC record as a TXT record in a subdomain named “`_dmarc`”; as shown in Figure 2.2, the DMARC record of `a.com` has to be published at `_dmarc.a.com`. The domain owner can customize the DMARC policy with a combination of tags and values in the DMARC record. The two required tags are ‘`v`’, which identifies the version (currently only ‘`DMARC1`’ exists) and ‘`p`’, which defines how to treat the email when validation fails; for example, the domain owner wishes to treat emails that fail in validation as spam by adding ‘`p=quarantine`’ to its DMARC record.

Receiver Authentication and Confidentiality

SMTP also has no built-in security feature for encrypting messages in transit and authenticating recipient. An SMTP extension called STARTTLS was proposed in 2002 to encrypt the message using a TLS session; to do so, the receiver has to send a plain-text command “STARTTLS” to the sender at the initial stage of the SMTP connection. Unfortunately, a man-in-the-middle can strip out the STARTTLS command to force the client communicates over an unencrypted connection.

To mitigate this attack, MTA-STS and DANE were proposed to let the receiver explicitly tell the clients that (1) it supports TLS for secure email transmissions and (2) validate the receiver’s identity using a policy file served through HTTPS (i.e., MTA-STS [85]) or TLSA records served through DNS (i.e., DANE [41]).

2.2.4 Reporting mechanisms

SMTP servers can send automated machine-generated reports via two mechanisms: (1) email receivers can send a DMARC report to the sender so that it can monitor authentication and judge threats. (2) Email senders also can send a TLS-RPT report to the receivers when it encounters TLS validation failure during the TLS handshake to help them debug their TLS configurations. They will also send a TLS-RPT report if any problems related to MTA-STS or DANE is encountered such as, no MTA-STS policy is found.

DMARC Report

DMARC aggregate feedback report (or DMARC Report) can contain useful information about authentication results; a domain owner who wishes to receive such feedback from email receivers can specify where and when to receive the report in their DMARC record using the three tags:

1. `rua`, which is a list of email addresses³ to receive an aggregated report about all emails sent from the domain, which are typically sent daily.
2. `ruf`, which is a list of email addresses to receive a message-specific forensic report when the DMARC validation fails; since this report is (1) a more detailed containing forensics of why DMARC validation fails, thus usually bigger, and (2) generated per email, the report sender may decide not to support this tag.
3. `ri`, which is the number of seconds elapsed between sending aggregate reports to the sender.

An email receiver can fetch DMARC records from the sender and generate reports during the DMARC validation process of the email sender. The domain name owner may publish a time interval to a

³The RFC 7489 standard [74] permits the use of various URIs including `http`, `file`, and `ftp`. However, since our focus is on email, we will concentrate solely on the `mailto` URI.

DMARC record with a `ri` tag [74] to let the email receiver send the report after the interval, but the email receiver can override it by generating reports on their own schedule.

External Destination Verification: A DMARC report contain a lot of useful information such as SPF and DMARC validation failure, and even, corrective action that needs to be taken by the domain owner. The report format is Extensible Markup Language (XML) to hold a various types of information and compressed, thus not reader-friendly. Thus, there are many third-party services that parse and analyze the report on behalf of the domain owner such as Dmarcian [134].

In such case, an email sender `a.com`, can put an email address of which domain name is outside the domain of the email sender `report@r.com` in either `rua` or `ruf` tag. Thus, it is possible that a bad actor to intentionally specify a `rua` tag to redirect reports to an external email address (i.e., victim) to make it flooded with unwanted reports. To prevent from this attack, a mail receiver (`b.com`) *should*⁴ check whether the external domain has agreed to receive the report by looking up a certain DMARC record on the external domain, which is called a DMARC authorization record. More specifically, the external domain `c.com` needs to publish a *DMARC authorization record* at `a.com._report._dmarc.c.com` with value `'v=DMARC1'` to tell the world that it is okay to send `a.com`'s report to `c.com`.

In the DMARC authorization record, `c.com` may also specify a `rua` tag to redirect a report to another email address, which is called *rua tag overriding*; this is useful when `c.com` wants to update the email address without asking each of their customers (i.e., email receivers) to update their `rua` tags. However, in order to prevent loops or indirect abuse, the domain name in the `rua` tag must remain same [74]. In other words, it only permits the user name in the email address to change. For example, if the `rua` tag of `a.com` forwards the report to `admin@b.com`, `b.com` cannot redirect its report to other domain. Thus, attackers may exploit the report senders that do not check EDV correctly to redirect their reports to a victim.

TLS-RPT

A DMARC report is generated and sent by the email receiver. In contrast, SMTP TLS Reporting (TLS-RPT) [86] is a reporting mechanism for *senders* to provide feedback to receivers when TLS validation fails due to STARTTLS negotiation errors, policy validation errors for DANE [67] and MTA-STX [85], etc. For receivers who wish to receive TLS-RPT reports, they can publish a TLS-RPT record under a prefixed domain name: `_smtp._tls`; for example, a TLS-RPT policy for a domain `b.com` can be retrieved from `_smtp._tls.b.com`. The policy consists of many directives; similar to a DMARC record, it includes the `rua` or `ruf` tag that list a list of email addresses to receive the report. Since the report is only generated when the email cannot be delivered properly because of TLS errors, the TLS-RPT report should be delivered to the external domain. Thus, unlike the

⁴The external destination mechanism is not currently mandatory (“must”) in the RFC [74], but strongly recommended (“should”).

DMARC Reporting, TLS-RPT does NOT have destination verification mechanism [86].

2.3 Transport Security

2.3.1 STARTTLS

Unfortunately, SMTP lacks inherent security features like recipient authentication or end-to-end message encryption. To provide opportunistic transport encryption, the STARTTLS extension was introduced in 2002 [66].

Contrary to HTTP that uses a dedicated port for TLS, SMTP (at least for mail exchange between MTAs) uses TLS via STARTTLS on the same port used for plain-text communication. Regrettably, this makes STARTTLS susceptible to downgrade attacks, where a man-in-the-middle attacker can remove the STARTTLS command. Furthermore the adoption of Server Name Indication [9] has been slow for SMTP in comparison to HTTP, leading to many MTAs using non-matching or self-signed certificates [43, 58, 82].

2.3.2 DANE

TLSA records DANE uses TLSA records to provide information that can verify the certificate of an application running on the domain. There can be multiple applications running on the same domain with different port numbers, and thus a TLSA record represents a port number, a protocol (i.e., TCP or UDP), and a base domain. For example, to request a TLSA record for an SMTP server of which the MX record is `mail.foo.com`, the derived domain must be `_25._tcp.mail.foo.com`. A TLSA record consists of four fields:

- **Certificate Usage** specifies how to verify certificates (or public keys) from TLS servers (e.g., SMTP server with STARTTLS). There are 4 usages depending on whose certificate is used (TA/EE) and whether PKIX validation is required (PKIX/DANE). The first two usages allow certificates from trusted CAs. Thus, a TLS server must provide a certificate chain that passes PKIX validation using root certificate stores. (i) PKIX-TA (Certificate Usage 0) allows using a root or intermediary CA's certificate. (ii) PKIX-EE (Certificate Usage 1) allows using leaf certificates issued by trusted CAs. In contrast, the next two usages do not require PKIX validation. (iii) DANE-TA (Certificate Usage 2) allows using any certificate of a root or intermediate trust anchor (TA). Thus, a server must provide a certificate chain, including a TA's (possibly self-signed) certificate, which can verify the server's leaf certificate. (iv) DANE-EE (Certificate Usage 3) allows using leaf certificates that can be self-signed; The DANE RFC [41] recommends using DANE-TA and DANE-EE since PKIX CAs offer no additional security for DANE in SMTP.
- **Selector** specifies whether the entire certificate or only the public key will be selected as Certificate Association Data.

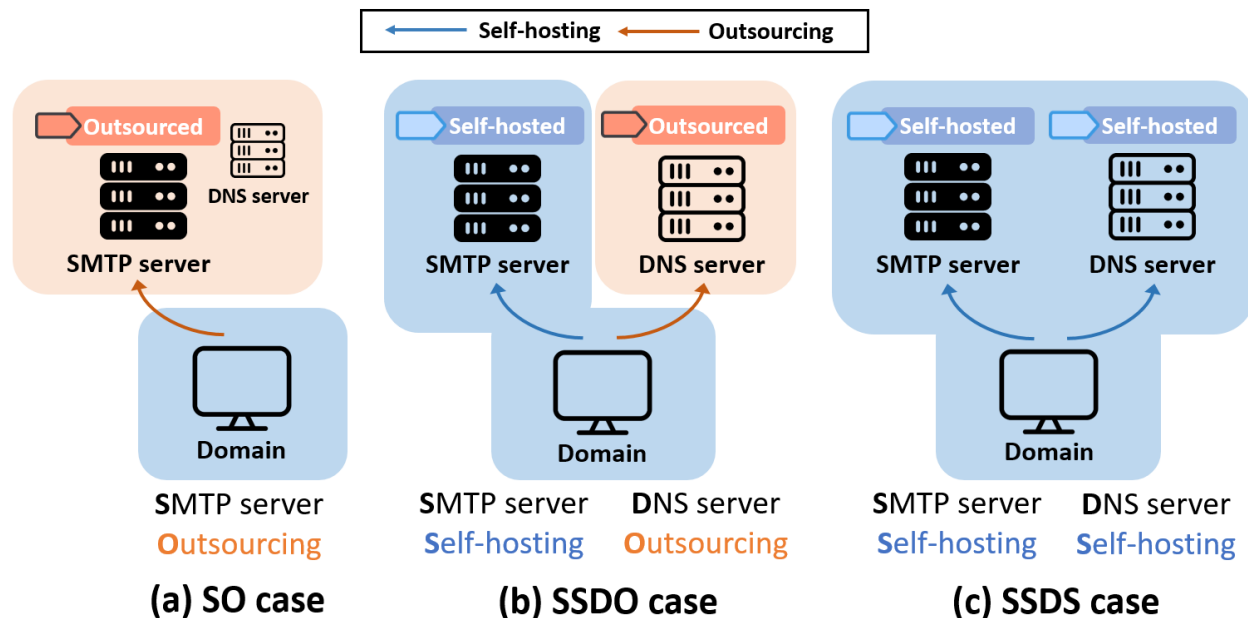


Figure 2.3: DANE management cases are classified depending on who manages the SMTP server and the name server. (a) SMTP server is outsourced (SO), (b) SMTP server is self-managed but name server is outsourced (SSDO), and (c) both SMTP server and name server are self-managed (SSDS). Note that the name server that serves the TLSA record is also outsourced if the SMTP server is outsourced.

- **Matching Type** specifies how to represent the selected certificate part. The original value, SHA-256 hash, or SHA-512 hash of the selected data can be used.
- **Certificate Association Data** contains the processed data depending on the above fields.

Where to deploy DANE SMTP At first glance, deploying DANE for an SMTP server seems straightforward because what the domain owner needs to do is to keep its certificate and TLSA record consistent.

However, it can become a bit tricky when the certificate and its corresponding TLSA record are managed by *two different entities*; domain owners may run both SMTP servers and name servers by themselves, but they can also choose to outsource their management to a popular email hosting provider (for their SMTP servers) and/or an external DNS operator (for their name servers). Thus, in practice, a domain owner has three options to deploy and manage DANE as illustrated in [Figure 2.3](#) [81].

First, a domain owner (e.g., [example.com](#)) may choose a popular email hosting provider (e.g., [one.com](#)) to outsource the SMTP server (labeled as SO). Typically, this is done by serving an MX record that delegates to an email hosting provider (e.g., [mx1.one.com](#)) such as

```
mx.example.com 600 IN MX 10 mx1.one.com.
```

Even though the MX record is served from the name server that the base domain name uses, the TLSA record will be fetched from the name server managed by the hosting provider because a TLSA record is bound to an MX record. Thus, when a domain outsources its email services to a DANE-enabled SMTP hosting provider, the provider manages both the certificate and its TLSA record. Thus, choosing a popular email hosting provider that can deploy TLSA records is an easy and effective way to support DANE. However, domain owners lose control over managing DANE because they neither manage a certificate nor TLSA records.

Second, domain owners may run and manage their SMTP servers by themselves. For their name servers, they can either (1) outsource to a popular DNS operator such as Cloudflare or their registrar's default name server (labeled as SSD0) or (2) manage it by themselves (labeled as SSDS). In the first case, *a domain owner has the responsibility to give the outsourcing DNS operator the correct TLSA record, which matches with the STARTTLS certificate that is used to encrypt the SMTP connection.* Thus, it might be problematic if the domain owners are not familiar with how to generate TLSA records (when they update their certificates) or how to give the generated TLSA records to their outsourcing DNS operators since mismatched TLSA records result in DANE validation failures. Furthermore, a recent study [34] showed that not all registrars support DNSSEC when the domain owners themselves are the DNS operator, making it impossible to deploy DANE due to missing DS records.

Why DANE validation fails In general, validating a TLSA record fails due to two reasons:

- **insecure or bogus DNSSEC:** DANE validation mandates correct DNSSEC deployment. When a DANE-validating client finds a TLSA record to be insecure, it ignores the TLSA record and concludes that the SMTP server does not support DANE, which brings all of the STARTTLS vulnerabilities back.⁵ When the TLSA record is determined to be bogus, the client is expected to abort the SMTP connection immediately.
- **Mismatched TLSA records:** this happens when the certificate and its corresponding TLSA record do not match. A previous study [82] found that about 4% of TLSA records could not be validated due to such a mismatch, but the root causes were not investigated, which motivates this paper.

2.3.3 MTA-STS

MTA-STS is a mechanism that allows a domain to require the use of Transport Layer Security (TLS) encryption with PKIX valid certificates for SMTP connections to specific MXes. It allows domain owners to specify a *policy* that lists one or more mx patterns matching the Mail Exchange (MX) hosts that support TLS for incoming email. Thus, *MTA-STS helps prevent downgrade attacks.*

⁵There is a debate whether we have to regard an insecure TLSA record as an invalid TLSA record or not [42]; as DANE mandates *full* DNSSEC deployment and a validator considers insecure TLSA records *unusable*, we regard it as an invalid TLSA record. For clarity, however, we also provide the details of the invalid reasons of a TLSA record for the rest of the paper whenever we analyze them.

To deploy MTA-STS, a domain name owner, e.g., of `example.com`, needs to implement steps across three different services:

DNS

To signal MTA-STS support, an *MTA-STS policy record* for the zone needs to be published in DNS. The policy record is a TXT record with the label `”_mta-sts”` under the domain name⁶; for example, the domain name owner may publish the following MTA-STS record;

```
_mta-sts.example.com IN TXT "v=STSV1; id=20240431;"
```

The presence of this record indicates that the domain supports MTA-STS and directs email servers to fetch the policy file from the policy server. It contains two key-value pairs: `"v"`, which specifies the version of MTA-STS (currently, only `"STSV1"` is supported), and `"id"`, which uniquely identifies the policy and should be updated whenever the policy file is modified. It also needs to fulfill the following conditions to be syntactically valid: 1) the record must begin with `"v=STSV1"`, 2) there cannot be more than one TXT records starting with `"v=STSV1"`, 3) there must be a `id` field present with an alphanumeric value, and 4) other key-value pairs can exist as extensions given that extension name and value satisfies ABNF rule in [85]. Additionally, the name for the policy server (see below), `mta-sts.example.com` must point to the right host address.

Policy Server

The policy server is a web server that hosts the MTA-STS policy file for a domain. The policy file is located at a defined `‘.well-known’` URI:

```
https://mta-sts.example.com/.well-known/mta-sts.txt
```

where `example.com` is the policy domain that opts to support MTA-STS and the domain name for the policy server is `mta-sts.example.com`. The policy server must provide the file over HTTPS, requiring TLS validation using a PKIX valid certificate. Once the validation is successful, the policy file is fetched. This file specifies the list of `mx` patterns matching the allowed MX hosts for the policy domain, the MTA-STS version, the operating mode, and the duration for which the policy is valid and should be cached by SMTP clients, as detailed in [subsection 2.3.4](#).

⁶For brevity, we refer to this as an MTA-STS record in this dissertation.

MX Hosts

Every mail-receiving domain has one or more inbound MTAs, defined either explicitly through MX records or implicitly through the A/AAAA records at the domain apex.

The MTA-STS policy file specifies the list of mx patterns matching allowed MX hosts for which TLS encryption is required. When deploying the receiving side of MTA-STS support, a domain owner must ensure that all MX they intend to list, support TLS with a PKIX valid certificate [28].

2.3.4 MTA-STS Policies

The MTA-STS policy file is a crucial component of the MTA-STS mechanism, providing SMTP servers with the necessary information to determine TLS requirements for email delivery. The policy file consists of key-value pairs separated by CRLF characters, with the main components being:

- `v` (version), which specifies the version of the MTA-STS policy format. Currently, the only supported value is "STSV1".
- `mode`, which indicates the expected behavior of a sending MTA in case of a policy validation failure (explained in [subsection 2.3.5](#)). It can have one of three values:
 - "enforce": In this mode, the sending MTA *must* enforce the MTA-STS policy. If the policy validation fails, the sending MTA *must* not deliver the email and should return an error to the sender.
 - "testing": In the testing mode, the sending MTA should perform policy validation but may still deliver the email even if the validation fails. Combined with SMTP TLS Reporting [86], this mode is useful for testing and transitioning to MTA-STS without disrupting email delivery.
 - "none": In this mode, the sending MTA should not perform any MTA-STS policy validation and should deliver the email as usual, regardless of the policy.
- `max_age`, which specifies the maximum lifetime of the policy in seconds; sending MTAs should cache the policy for up to this value and fetch the policy file at regular intervals.
- `mx`, which specifies allowed MX patterns; patterns can contain wildcard character. One or more of these patterns have to match with the selected MX host for policy validation, see [subsection 2.3.5](#).

2.3.5 Policy Validation

When an email server wants to deliver an email to a domain with a valid MTA-STS DNS record present, it first fetches the policy file from the policy server over HTTPS. The downloaded policy

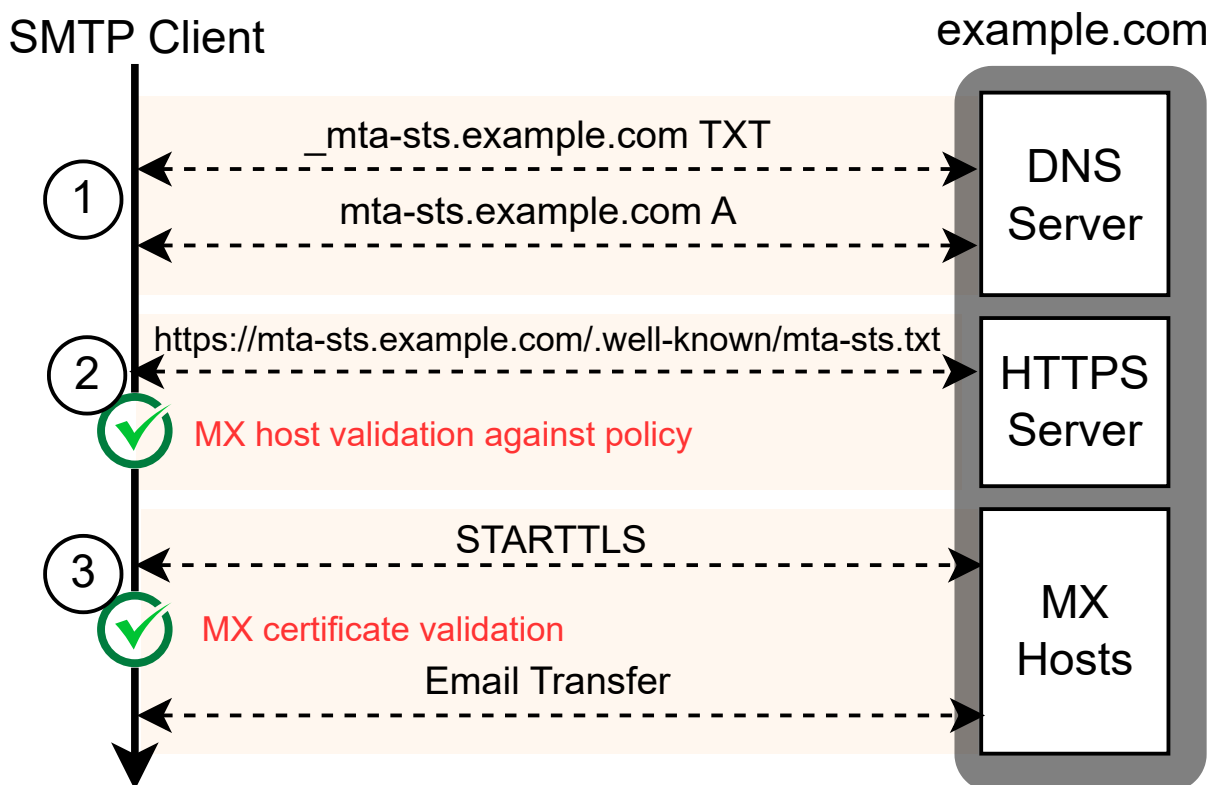


Figure 2.4: Flowchart of MTA-STS validation from an SMTP client: The client checks for MTA-STS support by looking up the MTA-STS DNS record and resolves the policy server’s IP address ①; The client fetches the policy file through HTTPS, and matches the MX host against the `mx` patterns specified in the policy file ②; The client connects to the MX host using STARTTLS, and validates the recipient MTA’s certificate ③.

is cached (“trusted”) for up to `max_age` seconds, contingent upon a successful HTTPS connection, and is referenced for subsequent transmissions until it expires from cache ⁷ Next, the sender MTA confirms that the chosen MX host aligns with at least one of the `mx` patterns presented in the policy file. If a match is found, the sending MTA initiates a TLS session with the corresponding mail server. If the MX host does not match any `mx` patterns, the MTA’s behavior depends on the mode in the policy file: if `enforce` is specified, the MTA *must* refuse to deliver; otherwise, it can proceed.

Finally, during TLS negotiation, the MX host *must* present a valid PKIX [28] certificate, and the client must verify that the certificate accurately includes the MX host in its Common Name or Subject Alternative Names. Should the MX host lack TLS support or fail certificate validation checks, the sender either proceeds or refuses delivery depending on the mode set in the policy.

Figure 2.4 illustrates how an MTA perform MTA-STS validation while delivering an email.

⁷This “trust on first use” (TOFU) approach can be vulnerable if the initial trust phase is compromised.

2.3.6 Policy Delegation

Domain name owners can delegate MTA-STS policy hosting to a third-party service. This allows a third-party to publish and maintain the MTA-STS policy file on behalf of the domain owner, reducing the complexity for the domain owner.

To delegate the MTA-STS policy, the domain owner creates a CNAME record for the policy host ("`mta-sts.example.com`") pointing to the third-party provider's policy host (e.g., "`mta-sts.provider.com`"). This enables the provider to obtain a domain validated [25] TLS certificate and serve the policy file for the domain.

However, policy delegation can introduce complexity and potential for misconfiguration, especially *when the policy hosting provider and the email service provider are different*. In such cases, the domain owner must ensure that the policy hosted by the third-party accurately reflects the TLS capabilities of the email provider, both initially and whenever changes occur. Failure to maintain consistency between the policy and the email provider's configuration can lead to issues with email delivery and security.

2.3.7 Removing MTA-STS

When removing MTA-STS, domain owners must follow a proper process to avoid email delivery failures. Since sending email servers can cache both the MTA-STS DNS record and the MTA-STS policy file, abruptly removing either can cause issues.

To address this issue, RFC8461 [85] specifies the correct procedure for removing MTA-STS; (1) Publish a new policy with "none" mode and a small `max_age` (e.g., a day); (2) Publish a new MTA-STS record with a new `id` to trigger fetching the new policy; (3) Wait for the maximum amount of time specified by the previous policy's `max_age` and the new policy's `max_age` to ensure that all senders have refreshed their cached policies; (4) Remove the MTA-STS DNS record, the MTA-STS policy subdomain, and the policy file from the HTTPS endpoint.

2.4 DNS and DNSSEC

The Domain Name System (DNS) underpins the Internet's mapping of *domain names* to specific *values*, such as the IP address of a server or the location of an email service. For example, an A record binds a domain name to an IPv4 address, while an NS record designates the authoritative name server for a domain.

However, the original DNS protocol offered no cryptographic safeguards, leaving it susceptible to attacks in which adversaries could forge responses. In response, the *DNS Security Extensions* (DNSSEC) [11, 12, 13, 56] were introduced to provide data authenticity and integrity. DNSSEC

introduces the following new record types to secure the domain hierarchy:

- **DNSKEY records:** Public keys that authenticate DNS data. Typically, each zone publishes two DNSKEY records: one for routine zone signing (known as zone-signing key or ZSK) and another for key-signing tasks (known as key-signing key or KSK), thereby compartmentalizing cryptographic responsibilities.
- **RRSIG (Resource Record Signature) records:** Cryptographic signatures covering entire sets of DNS records so-called “RRSets” for a particular domain name and record type. For example, a single RRSIG secures all A records under `example.org`, ensuring they have not been tampered with. The RRSIG is generated with a private key that corresponds to the public key declared in DNSKEY records.
- **DS (Delegation Signer) records:** Hashes of DNSKEY records stored in the *parent* zone. These hashes establish a secure link from child to parent, continuing upward until reaching the DNS root. Like other record types, DS entries are authenticated via RRSIG records.
- **NSEC/NSEC3 (Next Secure) records:** NSEC records are used to prove the non-existence of specific records of a domain. NSEC3 was introduced later [79] as a variant of NSEC to prevent zone enumeration using hashing and reduce the size of the zone file via the opt-out flag.

By cryptographically chaining each layer of the DNS hierarchy to the root, DNSSEC fortifies the global naming system against forgery and manipulation. The carefully designed interplay of DNSKEY, RRSIG, and DS records ensures that DNS responses can be validated from the root down to individual subdomains.

2.4.1 DNSSEC States

DNSSEC enables a resolver to classify DNS answers into distinct validation states. A security-aware resolver generally assigns one of three labels to a response:

- **Secure**, which indicates that the answer has been cryptographically verified. In this case, the resolver was able to build a complete chain of trust from a known trust anchor (e.g., the DNS root) down to the DNS record in question using all required DNSSEC records (e.g., DS and RRSIG records).
- **Insecure**, which indicates that at some delegation point, there exists a signed proof of the non-existence of a DS record. This situation typically arises when the queried domain lies in an unsigned zone (or beneath one).
- **Bogus**, which indicates a DNSSEC failure: the resolver expected the answer to be Secure (the zone is supposed to be signed), but the validation could not be successfully completed. In this case, the resolver believes there should have been a valid chain of trust, yet one or more checks failed; for example, a required signature was invalid or missing, or a DNSKEY did not match its DS record.

2.4.2 DNSViz

DNSViz [54] is a widely used DNS and DNSSEC diagnostic tool that can visualize, analyze, and help debug DNS configurations. Many administrators first encounter DNSViz through its interactive web interface [54] where they can input a domain name and view a graphical depiction of its DNS and DNSSEC status. This visualization highlights each step in the trust chain and flags any misconfigurations or validation failures.

DNSViz Command-Line Tool. Beyond the web interface, DNSViz also provides a command-line utility that underpins much of its functionality. This utility offers two primary commands that gather DNS data and then interpret it:

- *probe*: Systematically queries all the authoritative name servers of a domain to collect its DNS and DNSSEC data (e.g., DNSKEY, DS, and RRSIG records).
- *grok*: Interprets and “decodes” the raw DNS data gathered by *probe*, attempting to build a complete chain of trust from the root to the queried domain. If any link is missing or invalid, *grok* flags an error and annotates precisely where in the trust chain validation failed.

Error Codes. DNSViz assigns a rich set of error codes to describe potential DNSSEC misconfigurations. These codes capture both straightforward issues—such as missing or invalid signatures or expired RRSIG records—and more complex scenarios, including incomplete algorithm setup or delegation errors. At the time of writing, there are over 50 unique error codes, each accompanied by a human-readable message that clarifies the exact nature of the problem (e.g., “*The DS RRset for the zone included algorithm 5, but no RRSIG with algorithm 5 covering the RRset was returned in the response.*”).

Relevance to DNSSEC Management. Many operators rely on DNSViz’s web interface for quick diagnostics. By mapping out the DNSSEC trust chain in detail, DNSViz pinpoints the invalid records and delegation points. However, DNSSEC errors often have interdependencies; for example, an expired RRSIG on the apex of a zone can cascade into multiple validation failures, or a malformed NSEC record may disrupt negative responses across the trust chain.

Yet, translating DNSViz’s error messages into specific commands or configuration updates can be challenging for administrators. An error like expired signature may simply require re-signing the zone in BIND, whereas a mismatch between DNSKEY and RRSIG algorithms might necessitate a more complex key rollover process.

In this work, we first leverage DNSViz’s extensive logs to gain insights into the prevalence and persistence of DNSSEC misconfigurations, revealing recurring pitfalls and highlighting opportunities for automated or semi-automated solutions. We then leverage DNSViz’s diagnostics to

automatically map error codes to actionable fixes in BIND, streamlining the remediation process and reducing downtime.

Chapter 3

Related Work

In this chapter, we explore prior studies involving email security protocols, specifically SPF, DMARC, and MTA-STS, while also addressing related developments in DNSSEC.

3.1 Email Sender Authentication

Although prior work [98] has proposed decentralized approaches to enhance email sender authentication without relying on centralized infrastructures, traditional domain-based mechanisms such as SPF [124], DKIM [27], and DMARC [74] remain more prevalent and widely adopted. The following subsections review existing studies on their deployment and validation along with their security considerations.

3.1.1 SPF Ecosystem

Deployment for Outbound Emails There have been a few studies that have focused on the deployment of SPF *records*; for example, Mori et al. [90] investigated the top 500 Alexa domains in 2011, finding that half had deployed SPF records but also noting common syntactic errors in their SPF policies. Foster et al. [58] extended this work by examining the Alexa 1M domains in 2015, revealing that 40% of these domains had valid SPF records. Hu et al. [64] and Wang et al. [123] continued this trend by showing increasing deployment rates, 44.9% and 54.1% respectively, in subsequent years. Recently, in 2023, Czybik et al. [32] conducted a scan of 12 million domains, revealing that 56.5% have implemented SPF records and Blechschmidt et al. [22] reported a 41% SPF deployment ratio among the Top 10M domains.

Validation for Inbound Emails The evaluation of SPF validation policies for inbound emails presents unique challenges. Earlier studies have primarily resorted to one of three methods: (1) analyzing popular email server logs, such as Gmail, (2) registering with popular email service providers and testing their services, or (3) sending unsolicited emails to assess validation protocols. For example, Durumeric et al. [43] examined SMTP handshake logs for emails sent to and from Gmail between 2014 and 2015; their results showed that Gmail successfully validated 92% of incoming emails using SPF. Similarly, Foster et al. [58] evaluated popular email providers and

found that 15 out of 22 had SPF records for outgoing emails. Moreover, 20 out of these 22 providers performed SPF validation for incoming emails, although only 10 actively enforced the SPF results.

More recent research by Hu et al. [64] revealed that 31 out of 35 top email providers validated SPF records for inbound messages while Blechschmidt et al. [22] revealed 46 of 47 providers did so. This underscores the widespread adoption of SPF validation in the wild. In 2021, Casey et al. [38] sent 26K *legitimate* emails to measure SPF, DKIM, and DMARC validation rates, discovering that up to 85% of domains performed SPF validation for incoming emails. For a separate set of domains, they also measured SPF behavior by making an SMTP connection and then aborting the connection right *after* sending the DATA command but before sending any email content, which allowed them to measure SPF validation behavior of 1,574 MTAs. For this study, we intentionally disconnect *before* the DATA command since this command is intended for sending emails and might potentially activate spam extensions, leading to unnecessary computational resource consumption.

Misconfigurations Misconfiguration of SPF policies is a significant concern, as highlighted by numerous studies; for example, Durumeric et al. [43] reported that 29% of mail servers had overly permissive SPF policies that included more than 10,000 addresses. Hu et al. [64] indicated that 0.1% of domains in the Alexa 1M dataset had an SPF record that allowed all IP addresses to pass.

Tatang et al. [117] performed a graph-based analysis on the Alexa 1M dataset and observed that many domains included overly broad SPF policies from large providers like Amazon or SendGrid, making them vulnerable to spoofing attacks. Moreover, they found domains that permitted IP addresses from as many as 9,000 different ASes, marking them as potential targets for attackers. Scheffler et al. [112] discovered that several email servers did not adhere to RFC limits on the number of SPF referrals by connecting to *the entire IPv4 address space and sending emails using popular usernames*, thereby exposing a gap in compliance with standards.

Recent studies have even also pointed to potential security vulnerabilities in SPF implementations; Jeitner et al. [72] discovered possible injection and buffer overflow vulnerabilities in two popular SPF implementations, namely `policyd-spf` and `libspf2`.

While these previous works offer valuable insights into SPF misconfigurations, our study distinguishes itself by aiming for a comprehensive evaluation of the current SPF landscape. Our study extends these prior works in two ways. First, we undertake what is, to our knowledge, the most comprehensive assessment of SPF deployment, covering 176 million domains and their respective misconfigurations across a 15-month period. Second, our investigation is not solely confined to descriptive statistics; we also delve into the root causes of prevalent misconfigurations. We employ a mixture of quantitative measurement data and qualitative analysis to explore the underlying mechanisms contributing to these issues. Specifically, we examine the security implications of exceeding RFC-recommended DNS lookup limits and investigate the rationale behind why email operators may choose to deviate from these guidelines.

3.1.2 DMARC Ecosystem

DMARC Deployment and Validation As many email security protocols that aim to authenticate senders are proposed, a few studies have focused on the deployment of SPF, DKIM, or DMARC. A few studies have examined the adoption rate of email sender validation protocols. In 2015, Durumeric et al. [43] reported the early stage of DMARC adoption; the study showed that only 1.1% domains with MX records have deployed DMARC. Similarly, a recent study on BIMI [129] showed that 19% domains in the Tranco list [99] has a DMARC record and Wang et al. [123] reported of 11.9% of DMARC deployment. On the validation side, Hu et al. [64] conducted experiments on providers with email spoofing scenarios and revealed many popular providers missing DMARC validation in 2018 and tried to identify the cause behind the slow adoption of the protocol [65]. Casey et al. [38] also measured the adoption rate of SPF and DMARC validation in 2021 by sending legitimate email to user inboxes; they sent vulnerability disclosure emails to 42,924 email addresses encompassing 26K domains and found 54% of them to validate DMARC records. Holzbauer et al. [61] also evaluated multiple aspects of email delivery and found that 91.3%, 63%, and 53.5% of domains they tested adopted SPF, DKIM, and DMARC respectively.

Recently, BIMI (Brand Indicators for Message Identification) [26] and ARC (Authenticated Received Chain) [10] were proposed to enhance the spoofing detection, but their deployment is extremely rare [122, 129].

Attack Vectors Recently, several studies [24, 30, 110] examined email spoofing exploits to bypass SPF, DKIM, and DMARC; for example, Bennett et al. [24] identified a buffer overflow vulnerability in one of the SPF libraries called libSPF2. Shen et al. [110] exploited automatic email forwarding service to bypass the security validation and Chen et al. [30] applied black-box fuzzing and discovered 18 types of evasion that bypass DKIM validation, which worked in 10 popular email providers.

Our work extends prior work in two ways. First, we primarily focus on DMARC reporting where none of the prior studies have focused on. We show how SMTP servers deployed for their incoming and outgoing emails using the longitudinal datasets collected by our active measurement and comprehensive survey from popular email hosting providers. Second, we introduce DDoS attack scenarios that exploit misconfiguration and missing security considerations of DMARC reporting.

To our best knowledge, our work on DMARC reporting mechanism is the first to study focusing on the deployment, management, and their potential vulnerabilities.

3.2 Email Transport Security

In this section, we discuss related studies about security protocols for SMTP encryption and the DANE ecosystem.

3.2.1 SMTP encryption

SMTP does not encrypt its messages itself. Thus, the STARTTLS extension [66] was first introduced for email encryption. Several studies [43, 58, 68, 121] reported that STARTTLS is widely deployed. However, they also found widespread mismanagement of STARTTLS; 70% of the collected STARTTLS certificates cannot be authenticated due to misconfigurations. This is somewhat expected because STARTTLS does not specify what to do for invalid STARTTLS certificates. Recently, Poddebniak et al. [97] also revealed security vulnerabilities of STARTTLS such as command injection and credential stealing in SMTP, POP3 [87] and IMAP [31] protocols, which are largely due to additional but vague negotiation processes. To overcome these limitations, MTA-STS was also proposed to authenticate email servers and encrypt email messages [85]. Compared to DANE, it is simpler to deploy MTA-STS by leveraging TXT records. However, it is still vulnerable to MITM attacks as it does not mandate DNSSEC.

3.2.2 Ecosystem of DANE

Zhu et al. [131] measured the DANE deployment in 2015 by focusing on SLDs of .com and .net; they found that only 997 domains out of 485k signed domains have TLSA records; 13% of them were invalid. Recently, Lee et al. [82] focused on the deployment of DANE for SLDs with MX records in five TLDs and popular mail service providers in 2020. The paper showed a slow but gradually increasing DANE deployment rate; less than 1% of second-level domains of .com, .net, and .org deployed TLSA records. However, .nl and .se had deployed DANE relatively aggressively due to financial incentives from the registries. Also, they reported that 3.6% of TLSA records were not matched with the corresponding certificates, thus making them invalid. However, they could not find the root causes.

Considering DANE is still in the early stage, some efforts have been made to keep track of its deployment or to provide debugging tools for DANE administrators. For example, the NL registry (SIDN) and Dukhovni et al. publish DANE deployment statistics on websites based on their active scans and present SMTP DANE validation results on a daily basis [113, 114]. Also, there are web-based DANE validation tools [35, 45, 46] that can help administrators debug and configure TLSA records.

3.2.3 MTA-STS

As MTA-STS is a relatively new protocol, there is not much prior studies in the field. In 2021, Tatang et al [116] analyzed inbound MTA-STS misconfigurations and adoption rate. In 2023, Stock et al [22] analyzed both inbound and outbound MTA-STS deployment. According to their study, only 4 providers support MTA-STS validation. They identified lack of shared cache support for sending MTAs for a provider as the prominent reason for low outbound MTA-STS support.

3.3 DNSSEC

Despite over a decade of advocacy, DNSSEC’s global adoption has remained remarkably low. Sources like the Internet Society [3, 50] and Verisign’s DNSSEC Scoreboard [2] indicate that only around 5% of .com and .net domains are signed.

Operational Challenges. Registrar support for DNSSEC remains uneven; for example, only five registrars responsible for 50% of domain name registrations offered robust DNSSEC tooling in 2019 [34, 107]. Many organizations delegate DNS hosting to third parties (e.g., Cloudflare [36]), introducing further hurdles such as inconsistent DS record [34]. On the resolver side, Chung et al. [33] found that only 12% of security-aware resolvers actually validated DNSSEC records, undermining the protocol’s potential impact.

Inherent Complexity. Alongside operational challenges, the inherent complexity in DNSSEC’s fundamental design actively hinders its broader adoption [80]. The standard demands for meticulous precision and coordination between multiple entities which makes routine tasks like key or algorithm rollovers risky if any actor in the parent-child chain fails to follow best practices [88, 89]. For example, 8% of resolvers failed to adopt new root KSK even after a considerable period which led to the postponement of root KSK rollover in 2017 [88]. Deccio et al. [37] also showed that 24% of 2,634 measured zones exhibited configurations guaranteed to fail due to administrative complexity.

Real-World Failures. Misconfigurations have caused high-profile outages at the TLD level [51, 53] and for major platforms like Slack [119]. Recent measurements by Nosyk et al. [95] found 3.1M (1%) of 303M domains generating Extended DNS Errors (EDE) [77], suggesting a persistent gap between DNSSEC’s theoretical security and practical deployment.

Overall, these studies underscore a recurring theme: DNSSEC demands precise multi-party coordination, making misconfigurations alarmingly common. Our work further examines *which* errors are most common, and *why* they arise ([chapter 8](#)) and proposes an automated remedy ([chapter 9](#)).

Chapter 4

Management and Operational Challenges of SPF

SPF, or Sender Policy Framework, is among the earliest solutions aimed at preventing email sender spoofing and was proposed in 2003, and standardized in 2006 [124].

Its operation is straightforward: the domain owner specifies a list of authorized IP addresses via DNS TXT records¹ so that the email receivers can verify if the source IP address of an incoming email is on that authorized list.

However, the apparent simplicity is misleading; managing SPF records can be intricate and mis-configured SPF records are quite common, which expose active domains to phishing and spoofing attacks. This becomes more difficult when they point to other SPF records not under the control of the domain owner, such as those belonging to hosting providers. This adds another layer of complexity, as domain owners must also monitor these third-party SPF records for changes.

Furthermore, the age of the SPF standard has led to a disconnect between its original guidelines and current best practices, which often results in vulnerabilities. For example, the SPF records of email hosting providers may point to multiple other SPF records to utilize their infrastructure, requiring the receiver to make more than 10 DNS queries, a practice that is actually prohibited by the RFC guidelines [76]. Consequently, we observe a variety of implementation quirks in the wild. Our analysis reveals certain email software comes configured with lenient DNS query limits by default. Additionally, according to our survey and scans, some email providers lack query limitations altogether, thus making themselves vulnerable to potential reflection attacks.

In this part of the dissertation, we present a comprehensive study of the SPF ecosystem to bridge the gap between the operation practice and challenges in real-world SPF deployment. To examine the sender-side (i.e., the domain name owners) we employ 17 months of daily SPF record snapshots covering all second-level domains across .com, .org, .net, and .se top level domains. For assessing the receiver-side, we circumvent the ethical issues associated with sending unsolicited emails by leveraging ambiguities in the RFC to initiate SPF record checks.

Our contributions in this work are as follows [16]:

- We conduct a large-scale longitudinal analysis of the SPF ecosystem both from sender side and recipient side (RQ1).

¹For the sake of conciseness, we refer to these as SPF records in this dissertation.

TLD	Domains w/ MX	Domains with MX records		
		SPF	SPF	
			include	include with Ext.
.com	75.8 M	48 M (63.2%)	28.9 M (60.3%)	28.8 M (99.6%)
.net	6.5 M	3.5 M (53.8%)	2 M (57.1%)	1.98 M (99.6%)
.org	5.8 M	3.2 M (55.2%)	1.9 M (59.3%)	1.9 M (99.7%)
.se	845 K	439 K (52%)	365 K (83.1%)	365 K (99.7%)

Table 4.1: Overview of the datasets. The number and percentage of the domains that have SPF records are as-of March 27, 2023.

- We find that the vast majority of SPF records are *syntactically* correct, with fewer than 0.4% errors. However, issues arise during the evaluation phase, where many records (6.5%) incur excessive DNS lookups, exceeding the RFC-defined limit of 10 (RQ2).
- We identify two primary sources of such misconfigurations: first, two popular hosting providers that incorporate too many domains in their SPF records, and second, oversights by SMTP administrators who neglect to remove obsolete SPF records after service migration (RQ1).
- We outline an attack scenario that can temporarily prevent victims from receiving emails by causing their SPF validators to experience extended DNS resolution delays, all while the victims remain unaware due to the absence of emails in their inbox (RQ2).
- We conduct a survey among mail operators to understand the SPF landscape in practice. Our survey reveals that even large email providers managing over 1,000 accounts often deviate from required limits in RFCs [76], highlighting a disconnect between outdated standards and current operational needs (RQ1).
- Finally, we offer a set of community-focused recommendations aimed at mitigating potential abuse and bridging the gap between nearly two-decade-old standards and contemporary SPF usage (RQ3).

4.1 SPF Deployment for Outbound Emails

We first focus on domain name owners who *publish* SPF records and how they are deployed and managed.

4.1.1 Datasets

To scan a vast number of registered domains, we conduct DNS scans across four top-level domains (TLDs): the most widely used three generic TLDs (gTLDs) .com, .org, and .net gTLDs and a country-code TLD (ccTLD), .se, recognized for its aggressive deployment for email security protocols such as DANE [82] or DMARC [16]. Initially, we gather daily zone files from the respective registries for each of the four TLDs, namely Verisign for .com and .net, Public Internet

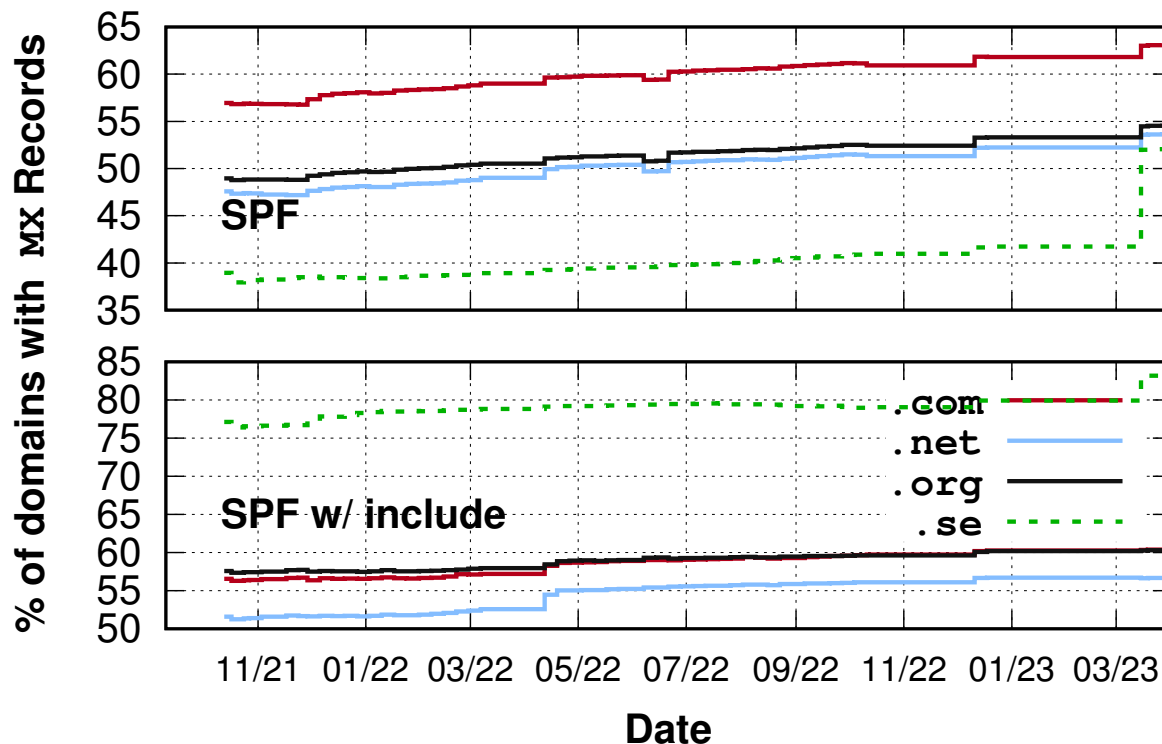


Figure 4.1: The percentage of domains with SPF records is increasing; the spike on March 15, 2023 for .se domains is because around 100K domains using loopia.se as their email hosting provider added a SPF record to their domain.

Registry for .org, and Internetstiftelsen for .se. This allows us to acquire the Name Server (NS) information for *all second-level domains* (SLDs). For each of SLDs, we fetch SPF records for each domain. In total, our snapshots span 17 months of 176 M domains from October 13th, 2021 to March 27th, 2023, which is summarized in [Table 4.1](#).

4.1.2 SPF Record Prevalence

We now first examine how SPF records are deployed by domain name owners. [Figure 4.1](#) shows the percentage of domains that have SPF records during our measurement period. First of all, we see a stable increasing trend across all four TLDs; for example, the deployment rate in .com domains increased from 56.9% to 63.2% during our measurement period. Similarly, we also observe that the prevalent usage of `include`, which indicates that the domain name owners import other SPF policy for their domains; for example, in case of .se, we find that 83% of the domains with MX records use external SPF policies for their authentication. We now examine whether the policy is from other parties; since it is not straightforward to compare whether two domains (base domain and the

TLD	Percentage Changes of Mechanism (Δ)							
	a	mx	include	ip4	ip6	ptr	exists	all
.com	-3.4	-3.8	+3.9	-2.0	-0.3	-0.8	-0.0	+0.2
.net	-2.9	-3.1	+5.1	-3.1	+0.0	-0.7	-0.0	+0.5
.org	-3.4	-3.9	+2.7	-1.8	-0.3	-1.1	-0.0	+0.3
.se	-9.1	-9.1	+6.2	-7.3	-0.9	-0.2	-0.0	+0.3

Table 4.2: Percentage change (%) in the usage of each mechanism between our initial snapshot on October 13, 2021, and our final snapshot on March 27, 2023; only `include` and `all` mechanisms have increased across the TLDs, as indicated by a green background.

Type	Reasons	# of domains	Example	
Syntax	Missing Values	ip4 and ip6 a and mx	7,827 (0.01%) 2,127 (0.003%)	"v=spf1 ip4: -all" "v=spf1 a: -all"
	Invalid Value		23,147 (0.04%)	"v=spf1 ip4:1.2.3 -all"
	Unknown Mechanism		53,466 (0.1%)	"v=spf1 mx -al"
	Recursive Value		55,907 (0.1%)	a.com TXT "v=spf1 include:a.com -all"
	Multiple SPF records		82,606 (0.15%)	a.com TXT "v=spf1 a -all" a.com TXT "v=spf1 mx -all"
	Evaluation	Missing Records	TXT record MX record A record	85,140 (0.16%) 324,524 (0.6%) 280,890 (0.5%)
Too many DNS lookups			3,548,014 (6.5%)	a.com TXT "v=spf include:a1.com -all" % <i>a1.com includes a2.com, etc.</i>

Table 4.3: The most popular reasons and their examples of SPF record errors are shown.

domain in `include`) are managed by the same entity², thus we compare whether the domains are different from the base domain or not.

Remarkably, across all TLDs, we find that 99.8% utilize SPF records distinct from their base domain. This strongly suggests that the majority are employing the `include` mechanism to import SPF policies from external sources; Such a trend is further substantiated when examining the changes in mechanism usage over our measurement period; Table 4.2 reveals the evolution in the percentage distribution of each mechanism within SPF records. Notably, we observe that only the `include` and `all` mechanisms have seen an increase in usage across the TLDs, further emphasizing the dependence on external policies and the general move towards stricter email authentication. This phenomenon can be attributed to the prevalent use of third-party mail hosting providers. These providers frequently recommend that their clients employ the `include` mechanism to reference their SPF records, which may encompass a broad array of authorized IP addresses.

4.1.3 SPF Records Management

The management of SPF records is crucial due to their role in authorizing the SMTP server of the sender domain. When checking the incoming IP address against SPF records, the receiving SMTP servers can encounter a variety of errors, which largely fall into two categories: syntax and

²One possible approach is to leverage WHOIS record, but the infrastructure is heavily rate-limited and notoriously inconsistent [83] and many domains are registered through privacy-preserving services, making it challenging to identify the actual owner [29].

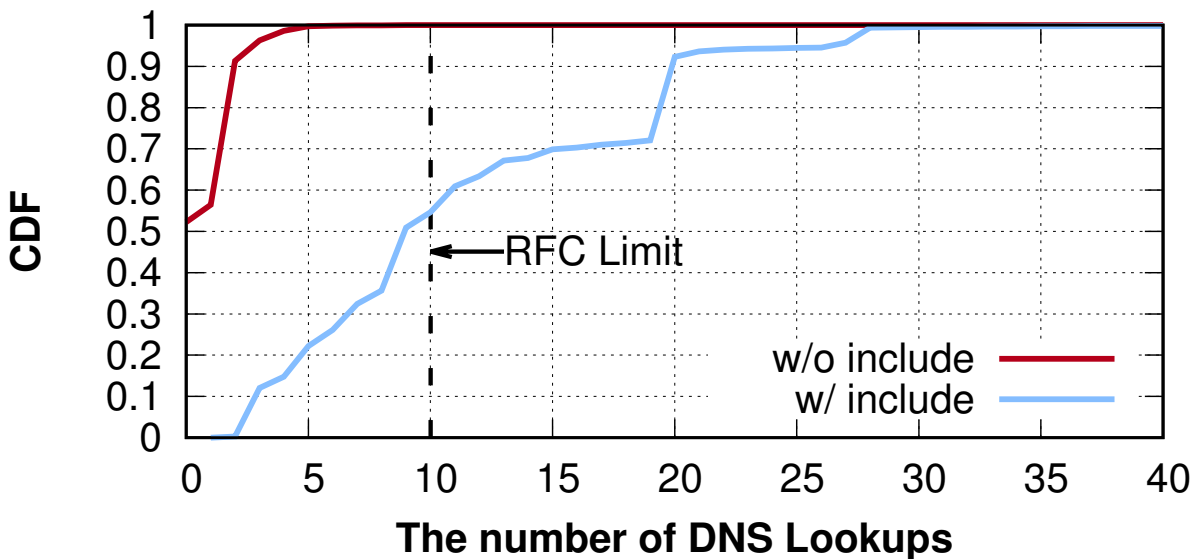


Figure 4.2: More than 9.2% of the SPF records with `include` requires more than 10 DNS look ups. Note that the x axis extends to 175!

evaluation errors.

Syntax Error: SPF records that do not follow the syntax of the standard [76] can be instantly detected during the parsing process. The syntax of the record is validated first before the evaluation, thus if there are any syntax errors anywhere in the record, it is determined as `permerror`, terminating the SPF evaluation process immediately without further evaluation.

Evaluation Error: SPF records have the ability to reference other DNS records, such as A records, through mechanisms. This means that even if an SPF record appears valid, it can be deemed invalid during the evaluation process if, for instance, the domain name in the `include` or `does not exist` fails. Furthermore, even if all mechanisms within an SPF record are successfully evaluated, the validation can still fail if it requires a large number of DNS lookups during the evaluation process since it can impose a significant performance overhead on the receiving server. To address this, the standard [76] mandates issuing a `permerror` when the total number of DNS lookups exceeds 10. One challenge is that domain owners may not promptly detect these errors, especially *when the SPF records referenced by mechanisms (e.g., `include`) are managed by other domains*; for example, if the SPF records managed by other domains incur the extra look ups causing the total number of DNS lookup exceed 10, it may cause `Permerror`.

Results

We use our latest scan and focus on 55 M domains that have SPF records to assess the accuracy of these SPF records by fully resolving all mechanisms contained within them. [Table 4.3](#) shows the results. First of all, upon examining the syntax of SPF records, we find that they *look* well-managed; for example, only 0.003% of SPF records do miss some values for specific *A* or *MX* modifiers.

However, when focusing on evaluation errors, more prevalent issues emerge. For instance, we find 690,554 domains (1.2%) with incorrectly configured mechanisms that are unresolvable. Even more concerning is the issue of excessive DNS lookups during SPF evaluation; specifically, 3,584,014 (6.5%) domains require more than 10 DNS lookups for evaluation. This is particularly significant because the RFC standard [76] mandates generating a `permerror` outcome in such cases, potentially disrupting email delivery.

Given that most domains in the `include` mechanisms originate from sources external to the base domain ([Table 4.1](#)), mainly due to hosting providers, this widespread occurrence is expected; for example, SPF records using `include` mechanisms require a considerably higher number of DNS lookups compared to those without such mechanisms. As depicted in [Figure 4.2](#), 3,564,056 (9.2%) of these SPF records, with at least one `include`, require over 10 lookups.

To investigate if this is a result of popular hosting providers misconfigurations, we evaluate each domain listed in the `include` mechanisms. We count their occurrences in SPF records; [Figure 4.3](#) reveals that 50% of SPF records reference six other SPF records from well-known hosting providers³. Next, we also do so, but focusing on those that require more than 10 DNS lookups; interestingly, 2,983,114 (83.7%) of them reference just two SPF records⁴. These are associated with Hostgator and Bluehost, each of their customers requiring 10 and 13 lookups, respectively. Yet, 616,581 (17.3%) of the domains that demand more than 10 lookups do not reference these two major hosting providers. We delve into this subset of domains in the following section.

4.2 Superfluous SPF Records

We have discovered that around 616 K domains require more than 10 DNS lookups but they do not include either Hostgator or Bluehost. This raises an immediate question: why do these domains require many lookups? In this section, our goal is to determine the essentiality of all the IP addresses or prefixes specified as approved senders in their SPF records, and whether any redundant mechanisms are in place. For instance, if an email server transitions to a different host or opts for an alternative hosting provider, the administrator must update their DNS entries accordingly; for example, in 2023, Microsoft Outlook encountered email outages due to their SPF

³outlook.com (11.6%), google.com (10.3%), secureserver.net (9%), registrar-servers.com (8.4%), websitewelcome.com (6.3%), mailchannels.net (2.7%)

⁴websitewelcome.com and bluehost.com

Rank	MX Records	Inferred SPF Records	# of Domains	Prob.	SPF Updated	# of Flatten Domains	# of Domains w/ Stale SPF
1	*.google.com	_spf.google.com	4,357,014	95.9%	✗	2,836	–
2	*.outlook.com	spf.protection.outlook.com	3,974,407	93.4%	✓	1,324	182 (13.7%)
3	*.registrar-servers.com	spf.efwd.registrar-servers.com	3,782,396	98.4%	✗	201	–
4	*.gmail.com	_spf.google.com	1,226,471	99.0%	✗	2,836	–
5	*.ovh.com	mx.ovh.com	571,484	95.2%	✓	48	10 (20.8%)
6	*.jellyfish.systems	spf.web-hosting.com	509,629	97.7%	✓	86	51 (59.3%)
7	*.hostinger.com	_spf.mail.hostinger.com	467,240	93.9%	✓	167	43 (26.3%)
8	*.qq.com	spf.mail.qq.com	465,717	97.9%	✓	151	44 (29.1%)
9	*.titan.email	spf.titan.email	427,060	99.2%	✓	38	9 (23.6%)
10	*.gandi.net	_mailcust.gandi.net	389,624	97.0%	✗	22	–

Table 4.4: The table lists the top 10 popular MX records, inferred SPF records, and their occurrence probabilities. Additionally, it indicates (1) if the IP addresses in the SPF records have been updated, (2) the number of domains that have flattened their SPF records, and (3) the domains with stale SPF records despite updates from their hosting provider.

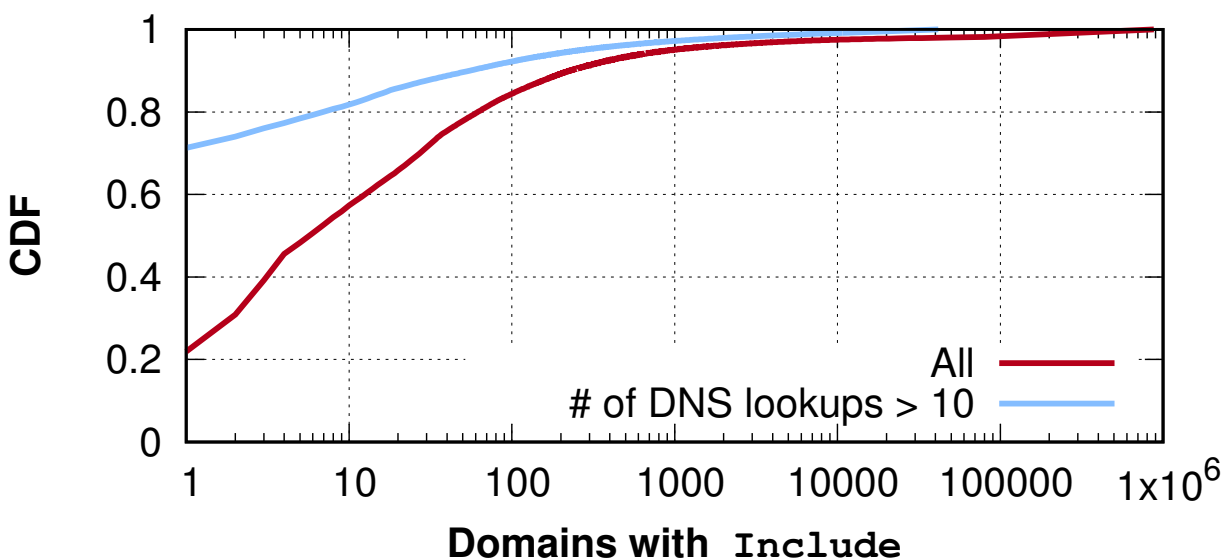


Figure 4.3: As of March 27th, 2023, only six SPF records appear in the include mechanisms for 50% of all SPF records. Remarkably, just two SPF records managed by two specific hosting providers account for 83.7% of the SPF records that require more than 10 DNS lookups for evaluation.

records not being updated despite changes in their actual email server IP addresses. Therefore, the prudent approach would involve retaining their historical include mechanism (i.e., those that might be considered superfluous) while simultaneously introducing new SPF records to ensure proper functionality.

The domains with superfluous SPF records does not seem to cause any operational problems; however, it may amplify the required DNS lookups causing delivery issue. Thus, we aim to find the domains having superfluous SPF records that persist despite alterations to the sending MTAs.

4.2.1 Methodology

To assess whether an SPF record contains superfluous entries, we first need to infer the ideal minimally covering SPF record for a domain. While, for a small set of major mail providers, these are published on their websites and can be assumed to be up-to-date, this is not the case for the average domain. Hence, while we can—and do—collect the documentation pages of large email providers to get ground-truth on their ideal SPF policy, we face a gap when it comes to smaller setups.

To bridge this gap, we work on the receiving mail servers communicated via MX records that are also responsible for sending emails for a domain. Even though this assumption is especially not true for large providers, we found it to be a common case also, e.g., signified by 350K domains in March 27, 2023 having mx as the only mechanism in their SPF policy. Nevertheless, such domains may also leverage external senders *in addition* to in-house sending, e.g., for newsletters.

Inferring SPF Records from MX Records

To infer the minimal SPF record of a domain, we first assume that domains utilizing third-party hosting services should have at least one `include` mechanism. Now, we compute the *likelihood* that a domain with a specific MX record, mx_x also includes an SPF record, spf_y in their `include` mechanism. To be precise, we follow these steps:

1. For a given domain name, we generate a list of tuples that consist of all possible combinations of its MX records and the domains in their `include` mechanisms; for example, if a domain name has a two MX records and three `include` mechanisms, we generate a total of 6 tuples.
2. We repeat this process for all domains and count the occurrence of domains for each tuple (mx_m, spf_n) , which we define as $d(mx_m, spf_n)$.
3. For each tuple, we calculate a conditional probability, $P(sp f_k | mx_m) = \frac{d(mx_m, sp f_k)}{\sum_{i=1}^n d(mx_m, sp f_i)}$ where $1 \leq k \leq n$ and n is the number of observed SPF records with mx_m .

Using this approach on our latest snapshot, we obtain 31,876,212 tuples along with their conditional probability. To evaluate our metric, we manually survey the top 10 email hosting providers to obtain the mapping of their MX records and SPF records for their customers.

As shown in [Table 4.4](#), we find that all of their probabilities are above 93%, each of which aligns with our manual survey. It is worth noting that attaining an absolute 100% probability is practically unfeasible; even in the case of a domain featuring a highly prevalent MX record (e.g., `smtp.google.com`), the inclusion of superfluous SPF records can diminish the likelihood of achieving a perfect match with `_spf.google.com` to a value lower than 100%.

Since our objective is to establish a mapping of MX records to their corresponding SPF records for popular hosting providers, we consider only those cases where the probability of SPF record given MX record, $P(sp f_k | mx_m)$ is above 90%, and the number of domains associated with such records is

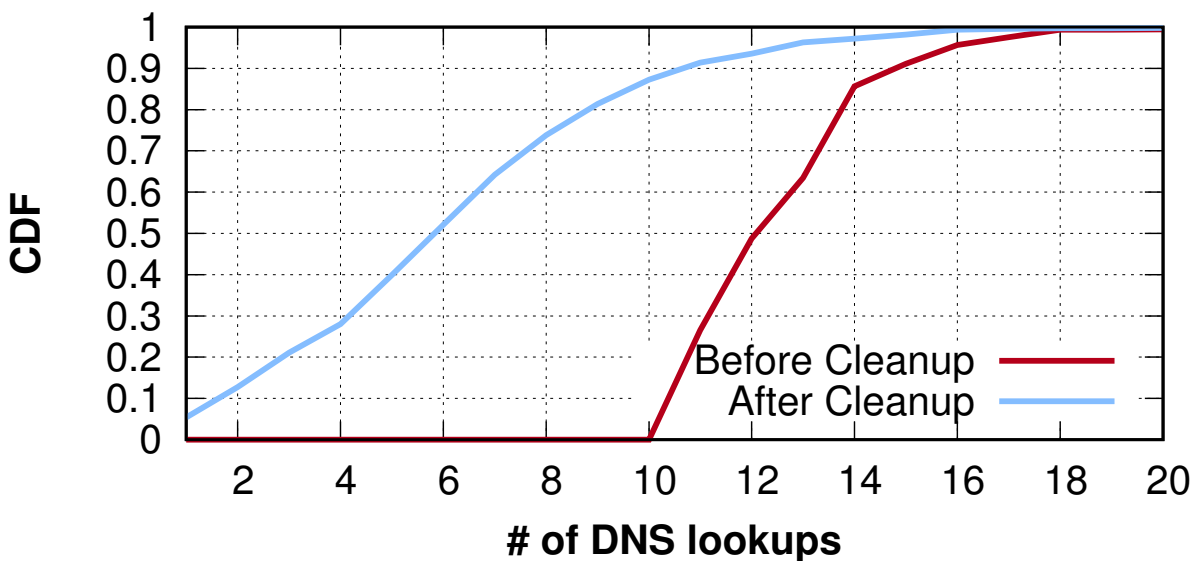


Figure 4.4: Removing superfluous SPF records can effectively reduce the number of required DNS lookups.

more than 100, which leaves us 3,141 tuples of MX records and their corresponding SPF records; clearly, this threshold introduces a trade-off between the size of the tuple list and accuracy. We delve into this topic in more detail in [subsection 9.6.5](#).

4.2.2 Domains with Superfluous SPF Records

Next, we identify the domains that have superfluous SPF records. To accurately pinpoint these redundant SPF records, we first only consider the domains for which we know all of the associated MX records for the SPF records in their `include` mechanisms using our compiled datasets, which leaves us with a total of 24,832 domains. Next, we delve into quantifying the number of domains that possess *superfluous* SPF records, ones that are not tied to any MX records; to be precise, we label an SPF record as superfluous if its corresponding MX records are not present in our dataset.

Remarkably, our analysis reveals that 20,124 (81.0%) of these domains are burdened with superfluous SPF records. Such redundant records could be problematic, raising the likelihood of mail delivery complications due to these extraneous SPF entries. The resolution of this concern lies in the removal of these superfluous SPF records; as shown in [Figure 4.4](#), we can observe that 17,554 (87.2%) of these domains can effectively streamline their lookup within 10, thereby alleviating the potential mail delivery problem.

4.2.3 Why Superfluous Records Exist

As elaborated in [subsection 4.2.1](#), a potential drawback of our methodology arises when sending MTAs are not represented in the MX records, leading to false-positives in our identification of superfluous SPF records.

We now investigate if these superfluous SPF records have any correlation with historical or outdated MX records that a domain might have previously used; if such a connection exists, it could strongly suggest that the presence of these superfluous records may be due to domain owners neglecting to update their SPF configurations in a timely manner. This scenario could arise when domains transition their email services from one hosting provider to another, inadvertently overlooking the removal of the former SPF records.

We examine if the MX records linked to these superfluous SPF records can be traced among the entire history of MX records used by the domain; in essence, we ascertain whether these superfluous SPF records can indeed be aligned with any of the domain's previous (and now outdated) MX records.

For the 20,124 domains that have superfluous SPF records in our latest snapshot, we found that 18,936 (94.1%) domains *have changed their MX records during our measurement period*. Now, for each historical snapshot, we examine if we can find any MX records that correspond to the superfluous SPF records, which is shown in [Figure 4.5](#); we find that a substantial majority of these superfluous SPF records 16,884 (83.9%) stem from obsolete SPF entries that have not been removed following the migration of mail servers. Interestingly, for 15.3% of these superfluous SPF records, we can find their corresponding MX records in our initial snapshot, suggesting these redundant entries have persisted for at least 15 months.

4.2.4 Case study: SPF Flattening

To mitigate the recursive nature of SPF records, the concept of *SPF flattening* has been proposed [127]. This technique involves fully evaluating the SPF records and explicitly listing all the assessed IP addresses or prefixes, thereby eliminating the need for `include` mechanisms. While SPF flattening is effective in reducing the recursion introduced by `include` mechanisms, it comes with its own set of challenges. Specifically, this approach becomes less reliable when IP addresses within the nested SPF records undergo changes. If these flattened addresses are not updated in a timely manner to reflect such modifications, email delivery may face disruptions.

To evaluate this possible issue, we employ a two-step approach: first, for each domain, we examine if the set of allowed IP addresses can fully cover all the permitted IP addresses specified in the SPF record tied to its MX record. This is done by comparing with our pre-established groundtruth dataset. Next, we monitor for any changes in IP addresses within the SPF records of providers throughout our measurement period.

From the steps, we observe that six providers that have implemented such updates, which suggest that the domains using these hosting providers and relying on flattened SPF records could

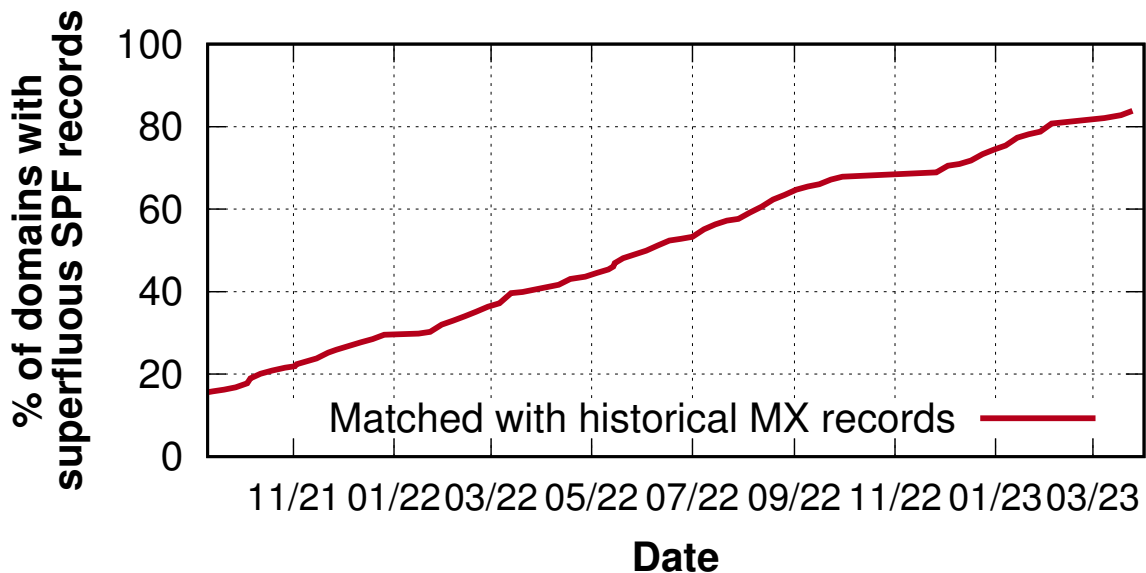


Figure 4.5: The percentage of domains with superfluous SPF records from our latest snapshot that match with historical MX records.

potentially face complications in email delivery; the results are presented in [Table 4.4](#).

Interestingly, we discover that numerous domains which have transitioned to flattened domain structures have not yet updated their SPF records, rendering them stale; for instance, among the domains using JellyFish and adopted flattened SPF records, 59.3% of them have neglected to revise their SPF records, despite the fact that the SPF records of the hosting provider itself have been updated. This discrepancy could potentially lead to email rejections due to SPF validation failures.

4.3 SPF for Inbound Emails

In the previous section, we have found that 6.5% of domains serve SPF records that require more than 10 DNS lookups, which may cause the email receivers to reject incoming emails from the domains since it raises `permerror`; however, such a high percentage might indicate that *some receiving MTAs may allow more than that*.

Now, we shift our focus to the SMTP servers that check SPF records for incoming emails. In this section, we primarily focus on the DNS lookup limit specified in the standard for: (1) total DNS lookup and (2) void lookup.

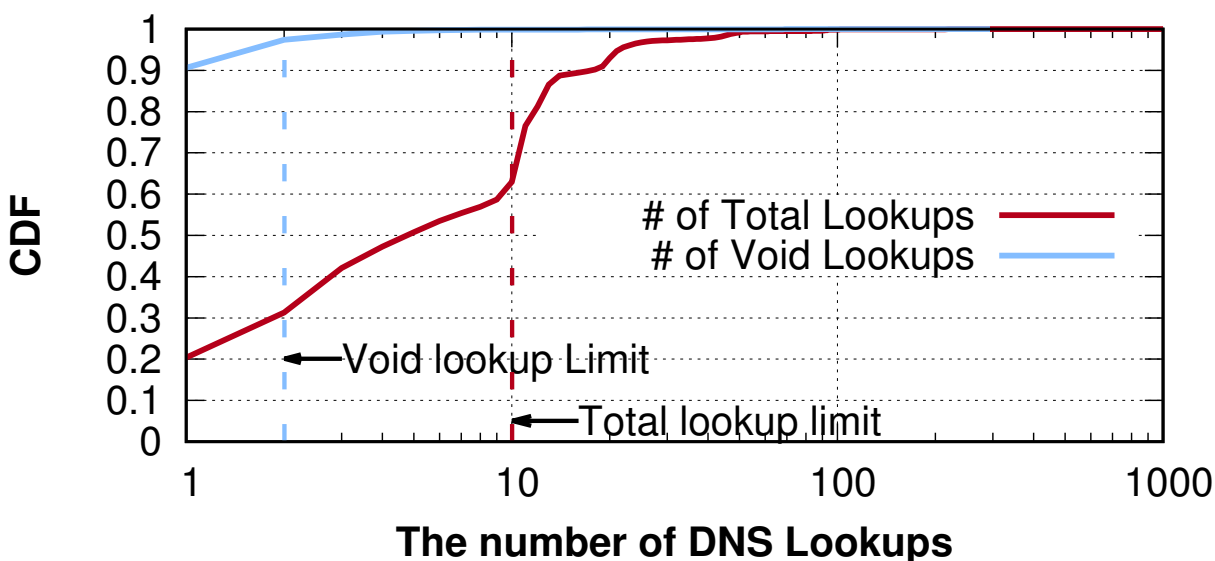


Figure 4.6: CDF of the number of DNS requests that our DNS authoritative server receives; 40% servers violate the total lookup limit recommendation in practice. Note that x axis extends to 25,117!⁵

4.3.1 Methodology

At first glance, it might seem straightforward to measure how MTAs impose limits on DNS lookups of SPF records; we could set up our own MTAs and serve an SPF record that incurs many DNS lookups by utilizing nested include mechanisms. Subsequently, we could send emails to randomly selected MTAs and tally the number of incoming DNS queries; this measurement methodology has been employed in prior studies to gauge the deployment of SPF checks for incoming emails [38, 112]. However, we argue that this approach could elicit ethical dilemmas; we have not been granted explicit permission to send emails to these MTAs, which may classify our action as unsolicited behavior, thereby potentially causing inconvenience or even harm. Instead, we leverage an RFC ambiguity to trigger DNS queries from MTAs.

Alternative Approach: We note that the RFC standard [76] does not specify the SMTP command that should trigger the receiving MTAs to perform the resolution of the SPF record; given that the SPF check is performed on the domain name in the MAIL FROM header, we may expect that some SMTP servers may send DNS queries for SPF records once it receives the MAIL FROM command from the sender; for example, we find that MTA software such as postfix with the iRedAPD extension sends DNS queries for the SPF record after RCPT TO command. This implies that it may be possible to trigger DNS queries for SPF records *without sending emails* to the recipient’s inbox if it looks up SPF records as soon as it receives the domain name; theoretically, as the MAIL FROM header contains the sender’s domain name, it may be enough for them to lookup SPF records; thus we only send the HELO, MAIL FROM, and RCPT TO commands to determine if it trig-

Name	Type	Ver.	DNS			
			# of Lookups		Timeout (s)	
			N_{lim}	V_{lim}	t_{lim}	T_{lim}
libspf2 [136]	Lib	latest	10	1	5	∞
Mail::SPF::Query [93]	Lib	< 1.8	∞	1	15	∞
Mail::SPF::Query [93]	Lib	≥ 1.8	10	1	15	∞
pyspf [140]	Lib	latest	10	2	2	20
milter-greylist [137]	M	latest	10	10	5	∞
spfmlter [141]	M	latest	10	2	5	20
mtpolicyd [138]	PS	latest	10	2	30	∞
policyd-spf [139]	PS	latest	20	1	5	20
iRedAPD [135]	PS	< 5.1	∞	∞	3	∞
iRedAPD [135]	PS	≥ 5.1	20	20	3	∞
SpamAssassin [21]	CF	latest	20	1	3	5
RSpamD [7]	CF	latest	30	30	1	∞

Table 4.5: The lists 10 popular SPF libraries (Lib), milters (M), policy servers (PS), and content filters (CF). Here, N_{lim} and V_{lim} represent the limits on DNS and void lookups, respectively; for example, Mail::SPF::Query and iRedAPD have no N_{lim} , resulting in endless DNS requests (highlighted in red). t_{lim} and T_{lim} indicate DNS timeout values for individual and total SPF evaluations, respectively. When parallel SPF validation is not supported, the SMTP server is vulnerable to DoS attack for inbound emails (highlighted in yellow).

gers DNS queries, and then terminate our SMTP connection *without* executing the DATA command to avoid transferring email data.

However, we acknowledge that our methodology could potentially identify fewer vulnerable receiving MTAs because most email spam filters or SPF milters perform an SPF record check once the mail data is transferred. Nevertheless, we believe that this approach is appropriate for understanding the vulnerability empirically without adversely affecting the MTAs in the wild. Furthermore, we obtain a set of unique MX records by scanning zone files, and then resolve their IP addresses while removing those that share the same IP addresses to limit the number of MX servers.

Experiment Process:

We purchase a domain name, *a.com*, and configure our customized DNS authoritative server and SMTP server using Postfix [101]. From the latest snapshot of four zone files, we examine the MX records and their corresponding IP addresses to create a unique set of MTAs, yielding 1,886,825 MX records in total and then proceed as follows:

1. For each MX record, mx_a , we generate two unique subdomains: $mx_a.t.a.com$ and $mx_a.v.a.com$ to serve two types of SPF records to evaluate their DNS lookup and void lookup limits. To assess

⁵Along with this SMTP server, we found that two additional SMTP servers sent duplicate queries over 1,000 times; we attempted to contact them using common administrator email addresses (e.g., *postmaster@*) and the admin name in the *rname* field of the SOA record, but we were unable to reach them. This could be due to SPF validators that store SPF records in their database and periodically update the SPF records for security purposes (e.g., iRedAPD converts SPF records of specified domains to IP addresses on an hourly basis).

the DNS lookup limit, we serve SPF records featuring 50 include mechanisms, which incurs 51 DNS lookups. For the void lookup limit, the SPF record includes 50 include mechanisms, 36 of which lead to void lookups; we provide an explanation of the choice of parameters in [subsection 9.6.5](#).

2. We establish two SMTP connections to the target MX server: one for gauging the total lookup limit (Exp. #1) and another for probing the void lookup limit (Exp. #2). In each connection, we issue a HELO command featuring the unique subdomain.
3. Subsequently, we transmit our MAIL FROM address with noreply as the username.
4. Next, we send a RCPT TO command with postmaster as the username.
5. We pause for 3 seconds to allow sufficient time for the SPF queries to reach our DNS authoritative servers, after which we terminate the SMTP connection.
6. All incoming DNS queries are logged for analysis.

We conducted these experiments from February 28, 2024 to March 16, 2024. Ethical implications of this experiment are discussed in [section B.3](#). Along with this, we investigate 10 popular SPF libraries, milters, policy servers, and content filters, as detailed in [Table 4.5](#), through source code analysis and testing.

4.3.2 Results

With our methodology, we successfully connect to 1.2M (64%) SMTP servers out of 1.89M ones in total and find that 81,843 (6.8% of 1.2M) initiate SPF queries prior to issuing the DATA command; this suggests that most SMTP servers opt for SPF validation after the DATA command, likely to exploit the policy flexibility available at this stage.

Now, we focus on the number of DNS queries we receive. [Figure 4.6](#) shows the results. First, we find that 903 servers (1.1%) send more than 51 queries, but they do so by *issuing duplicated SPF queries*. This could be due to either (1) SPF validation taking place from multiple extensions, (2) the involvement of multiple DNS resolvers executing identical queries, or (3) the use of SPF validators that lack DNS lookup limits as elaborated in [subsection 2.2.1](#). We will further explore this in [section 7.1](#). Among them, we find that 195 SMTP servers queried for all 50 include in our SPF record; this strongly suggests that some SMTP servers use SPF validators that *do not have any total lookup limit*. Out of these 195 servers, 165 of them maintain a void lookup limit as they aborted SPF resolution right after querying the set number of domains that incur NXDOMAIN responses in Exp. #2; the rest (30) requested all 36 domains that result in void lookups, which indicates that these servers could potentially serve as reflectors for launching DNS queries against a targeted victim's authoritative server. Interestingly, even for those who made < 51 queries in Exp. #1, we find that 39 SMTP servers made all 36 void lookups, which indicates that they do not have a void lookup limit even though they have a total lookup limit.

These findings corroborate the hypothesis that a substantial number of SMTP servers operate with lax or even unlimited lookup limits. As validated in [Table 4.5](#), both Mail::SPF::Query and

iRedAPD lack a DNS lookup cap. Given that the iRedAPD's patch was in August 2022, this signals potential vulnerabilities in numerous mail servers.

4.3.3 Threat Model

Each MTA establishes its own timeout limits for SPF validation. If the corresponding SPF validator does not support parallel processing of incoming SMTP connections, attackers can exploit this timeout leading to interruptions in processing valid incoming emails for a victim recipient MTA.

Preliminaries: To execute this attack, attackers should have (1) a domain name (2) a custom authoritative server for the domain to serve DNS records and manipulate DNS response time. More details on the execution is introduced in [subsection 4.3.4](#).

1. **DoS attack:** the attackers can disrupt the processing of benign incoming emails by deliberately slowing down DNS responses during SPF validation, causing timeouts and thus either delaying or outright denying email processing.
2. **Low economic barrier:** the attacker may (1) register a domain name using the TLDs that offer domain names at no cost (e.g., .ml [106]) (2) utilize a free-tier hosting service (e.g., AWS micro instance) to operate a customized DNS server designed to return DNS responses with extra delays.

4.3.4 DoS Attack for Inbound Emails

Each MTA sets individual timeout limits for SPF validation, leading to variations in the waiting time for process completion. For example, when a Policy Server is responsible for validation, the MTA may wait until a *policy service timeout* is triggered. Likewise, a *Milter command timeout* comes into play when Milter handles the validation. In the case of Postfix, the default settings specify a 100-second limit for policy service timeout and a 30-second limit for Milter timeout during SPF validation for incoming emails [102]. Should SPF validation exceed these predefined timeframes, the SMTP connection handling the email will be terminated.

This feature becomes a vulnerability when the SPF validators in use *do not support parallel SPF validation and dns lookups continue even after MTA aborts the connection*; as a result, if the validation process exceeds the *policy service timeout*, it can cause a cascade of timeouts, leading to the rejection of multiple incoming emails. This makes the emails undeliverable to the recipient MTA, effectively serving as a Denial-of-Service (DoS) attack vector.

For example, iRedAPD imposes a limit of 3 seconds (t_{lim}) for each DNS query and a maximum of 20 DNS lookups (N_{lim}); an attacker can exploit these constraints to induce delays. Specifically, by carefully crafting DNS responses that take exactly 3 seconds to resolve, the attacker can cause

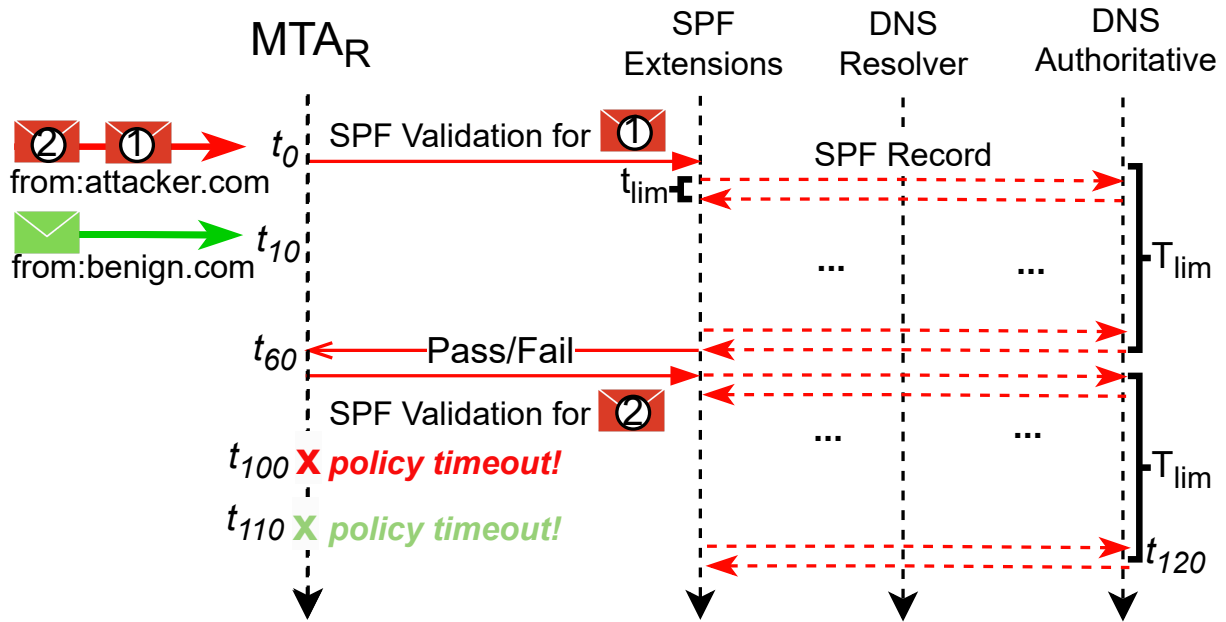


Figure 4.7: Certain SPF validators (e.g., iRedAPD or libspf2), do not support parallel SPF validation. In such cases, attackers can introduce delays up to either T_{lim} or $t_{lim} \times N_{lim}$ seconds; Each MTA has its own timeout limit for SPF validation, denoted as M_{lim} . Should the delay exceed this threshold, subsequent incoming emails may be rejected.

the SPF validation process to last for up to 60 seconds (3 seconds \times 20). This can lead to a SPF validation exceeding the system's configured timeouts, effectively becoming a bottleneck and causing incoming emails to be rejected.

However, in practice, t_{lim} is also influenced by the DNS resolver's own timeout settings. Should the DNS resolver that the SPF validator employs have a shorter timeout value, the resolver could trigger its own timeout, potentially reducing the window for attack but also complicating the validation process. In such a scenario, attackers can take advantage of the fact that *CNAME record expansion is managed by the DNS resolvers, thus not contributing to N_{lim}* ; this allows the attacker to induce the resolver into initiating a fresh DNS request, effectively resetting the resolver's timeout. For example, consider the attacker sets up the following SPF and DNS records:

```
a.com      TXT      "v=spf1 include:r1.a.com"
r1.a.com   CNAME    r2.a.com
r2.a.com   TXT      "v=spf1 include:r3.a.com"
r3.a.com   CNAME    r4.a.com
```

In this example, the attacker has purposely created a chain of CNAME records to prolong the DNS lookup process; when the DNS resolver reaches the CNAME record at `r1.a.com`, it is compelled to initiate a new DNS request for `r2.a.com`. Subsequently, the request for `r2.a.com` leads to

another CNAME record (`r3.a.com`), triggering yet another DNS request and resetting the timeout. This can be extended further, thereby increasing the time needed for SPF validation but decreasing the chance to trigger t_{lim} .

By sending two emails simultaneously, the attacker can cause a cumulative delay of 120 seconds; given that Postfix's default policy service timeout is 100 seconds [102], all incoming emails in the 20-second window following the initial delay from t_0 to t_{20} would be rejected. This situation is illustrated in Figure 4.7. Specifically, the second email initiates its SPF validation at t_{60} and will hit the policy timeout, leading to termination of the SMTP connection at t_{100} , but blocking further SPF validation until t_{120} .

It is worth noting that SPF validation often occurs after the `RCPT TO` command; this allows attackers to potentially initiate a DoS attack without actually dispatching emails, making detection and mitigation more challenging.

4.3.5 Responsible Disclosure

We have reported a vulnerability concerning our proposed attack to the developer's forum [19] and GitHub's issue tracker [20]. The previously identified issue with `Mail::SPF::Query` versions that were susceptible has been resolved. Furthermore, to assist users in assessing the security of their domains, we have incorporated a search feature on our website to let them inspect the status of their SMTP servers. Importantly, this feature is designed to prevent the scraping of entire lists.

4.4 Discussion

We presented a multi-faceted, deep-dive investigation into SPF records and their management, encompassing both a quantitative analysis based on comprehensive scans and qualitative insights obtained through operator surveys. We found that a significant majority (60.2%) of SPF records use the `include` mechanisms, largely depending on external domains. Most SPF records are syntactically correct, however, they falter in the evaluation phase primarily due to excessive DNS lookups that surpass RFC-imposed limits; our study points to two major culprits behind these misconfigurations: popular hosting providers with large number of `include` in their SPF records and SMTP administrators failing to update obsolete records post-migration. We demonstrate a novel attack vector that exploits these misconfigurations, causing disruptions in email receipt without alerting the victim. Our operator surveys reveal that many large-scale email providers deviate from RFC best practices, underscoring a significant gap between existing standards and current operational requirements.

In summary, our work serves as a comprehensive resource for understanding the state of SPF management, its security implications, and avenues for improvement. It is our hope that this dissertation will act as a catalyst for more secure and effective email systems moving forward.

4.4.1 Recommendations

Altogether, our findings illuminate the present landscape of SPF deployment and management. We offer insights on revisions needed for existing recommendations and RFCs.

- Given the increase in (nested) SPF policies over time, we suggest increasing the current lookup limit of 10 to, e.g., 20 as it is already handled in several implementations.
- Domain owners and email operators should make use of subdomains for (e.g., newsletter) sending or other tasks delegated to an external email service provider to avoid the ‘too many lookups’ issue. These subdomains can then have a smaller SPF policy, only including, e.g., the service used for newsletters (`news.example.com`) or the CRM software (`support.example.com`).
- Email service providers on-boarding a new client should not only validate that client’s SPF policy for the correct inclusion of *the provider’s records*, but should also check whether their customers have superfluous records.
- Non-parallel email receipt/milter execution should be avoided, and milter operators should more clearly document it if their implementation or certain configurations for their software leads to non-parallel email receipt.
- Depending on the preferences of an email operator, they might choose to perform SPF queries after the remote server indicates the end-of-data (‘. \n\n’), but before accepting a message; this approach can be advantageous for protecting against the attack we proposed. However, this strategy might result in missing the opportunity to reject connections from unauthorized sources early, leading to memory consumption (to load the message in the DATA command). Therefore, email operators must carefully configure their systems according to their needs.

Furthermore, we have to acknowledge that especially milter libraries for established protocols like SPF are a part of the often forgotten and regularly maintained by but a few essential building blocks of modern technology such as NTP [108]. They hardly change, thus often hardly require maintenance—unless vulnerabilities occur [24]—and hence it becomes easy for such milter projects to slide into obscurity and abandonment.

Finally, the common truth of operating digital infrastructure also remains true; operators should regularly check for updates to components of their setups.

4.4.2 Limitations

Our study has several limitations. First, our analysis of superfluous SPF records based on MX records is susceptible to false positives. This inaccuracy can be attributed to two factors: (1) Domains not listed in MX records might still be legitimate email senders, such as newsletter services. (2) Our 90% threshold heuristic may not universally hold true. To overcome these limitations, we confined our analysis to domains with more than 100 associated senders and corroborated our results with ground-truth data from the top 10 email providers.

Second, in [section 4.3](#), we have classified SMTP servers that perform more than 50 record lookups as vulnerable, which may introduce false positives. However, as elaborated in [Table 4.5](#), our analysis indicates the maximum lookup limit identified across all evaluated software is 30, suggesting a lower likelihood of false positives.

Third, SPF queries were received from 81,843 (6.8%) recipient MTAs we successfully connected to. Despite appearing modest, we believe that this number represents a well-balanced trade-off compared to the numbers obtained by sending emails (e.g. 26K SPF validating MTAs in [\[38\]](#)). Furthermore, we evaluated 10 popular open-source SPF validators; although there is a chance that some less widely used software may have been missed, we have still covered validators of 49 (91%) administrators who participated in our survey.

Fourth, we used `postmaster` as the recipient username. Out of 1.2M servers we were able to successfully connect to, we got a `negative` response to the `RCPT TO` command for (679K) 56% of them, which indicates the absence of this user; MTAs are also often set up to whitelist `postmaster` address. Thus, even if `RCPT TO` command succeeds, sender validation might not take place.

Fifth, it is important to note that our study intentionally excludes the consideration of additional lookup limits resulting from MX or PTR records within an SPF record. This decision was made to avoid the ethical implications associated with initiating four connections to the same SMTP server.

Sixth, when identifying superfluous SPF records, we seek out the SPF records and their corresponding MX records to create a *pseudo-ground truth dataset* by focusing on the popular SPF records associated with more than 100 domains. Aiming for a high level of confidence in mapping to pinpoint superfluous records, we adopt a conservative threshold of 100 domains; for example, reducing this threshold to 20 domains notably expands the coverage, from 20,124 to 39,632 domains with superfluous records. However, we believe this adjustment leads to a high number of false positives.

Lastly, our analysis covers all second-level domains within four TLDs, thus not accounting for other domains that operate under subdomains.

Chapter 5

Measurement and Security Implications of DMARC Reporting

Given the rising deployment status of SPF and DKIM [24, 58, 123], however, there had been a lack of standard, domain-specific policies for receivers to decide how to handle messages when either SPF or DKIM validation *fails*.

Domain-based Message Authentication Reporting and Conformance (DMARC) [74] solves this problem by allowing domain name owners to publish a policy as a DNS record (i.e., DMARC record) specifying how email receivers should handle emails that fail either SPF or DKIM validation. Additionally, DMARC also provides a *reporting* mechanism so email senders can learn how receivers applied the selected policy to received mails via XML reports sent back to the addresses specified in the DMARC record.

Unfortunately, DMARC, also, is complex, creating many opportunities for mismanagement, often opening doors for attackers to exploit; for example, DMARC reports are usually big and compressed, and generated automatically from SMTP servers. Hence, for example, attackers can put arbitrary addresses as report recipients in their DMARC record and trigger the SMTP servers to send reports, consuming resources. Surprisingly, the DMARC reporting ecosystem has not yet been empirically analyzed. While there have been several studies on DMARC record deployment, we find that no prior efforts investigated how email senders and receivers handle DMARC reporting.

Therefore, we present a comprehensive study of the DMARC reporting ecosystem. To study the sender-side (i.e., report receiver), our work uses 13 months of daily DMARC record snapshots for all second-level domains of .com, .net, .org, and .se. To study the receiver-side (i.e., report sender), we perform a controlled experiment on the top 7 email hosting providers as well as two open-source software.

Overall, following are our key contributions [17]:

- We observed 49% of domains having DMARC records use reporting features, 70% of which are configured to forward the report to external domains. However, 26% of them are misconfigured, thus not able to receive the report, which also happens in the most popular domains (e.g., 10% of the top 10K most popular domains) (RQ1, RQ2).
- Out of 7 popular email hosting providers that support DMARC reporting, we identify 6 of them

TLD	Domains MX Records	Domains with MX records		
		DMARC	DMARC Report	
				Report from Ext.
.com	75.6 M	5.0 M (6.6%)	2.4 M (49.4%)	1.7 M (68.8%)
.net	6.5 M	453 K (6.9%)	245 K (54.1%)	172 K (70.2%)
.org	5.8 M	390 K (6.7%)	213 K (54.5%)	152 K (71.4%)
.se	848 K	81 K (9.6%)	30 K (37.4%)	24 K(80.1%)

Table 5.1: Overview of the datasets for this study. The number and percentage of the domains that have DMARC records are as-of January 8, 2023. 49% of DMARC records list email addresses to receive the DMARC reports. 70% of them use external domains to forward the reports across the four TLDs.

do not follow the security recommendation from RFC7489 [74], which allows attackers to turn them into *reflectors* to send reports to arbitrary addresses (RQ2).

- We introduce multiple attack schemes leveraging ambiguity in the standard that make victims experience a large influx of unwanted traffic (i.e., reports) (RQ2).
- We conducted a survey of email administrators that shows misconfigurations in practice, even for those that manage more than 1,000 accounts (RQ1).
- Finally, we provide guidelines to the community that can help protect from abusive scenarios and enable the improvement of existing standards (RQ3).

5.1 DMARC Deployment for Outbound Emails

The domain owners can list email address(es) in their rua or ruf tags to receive DMARC reports from the SMTP servers that generate report in their DMARC validation process. We begin our analysis by focusing on the deployment and management of DMARC by domains.

5.1.1 Methodology

To cover a large number of registered domains, we use DNS scans from four TLDs: the .com, .org, and .net gTLDs, which are the most popular gTLDs and .se ccTLD, which is well known for deploying email security protocols such as DANE [82]. For each of the four TLDs, we first obtain daily zone files from their registries (.com and .net from Verisign, .org from Public Internet Registry, .se from Internetstiftelsen) to obtain Name Server (NS) for all second-level domains (SLDs). For each of these SLDs, we construct and fetch DMARC records for each domain.

In total, our snapshots span 12 months from January 3, 2022 to January 8, 2023, which is summarized in [Table 5.1](#).

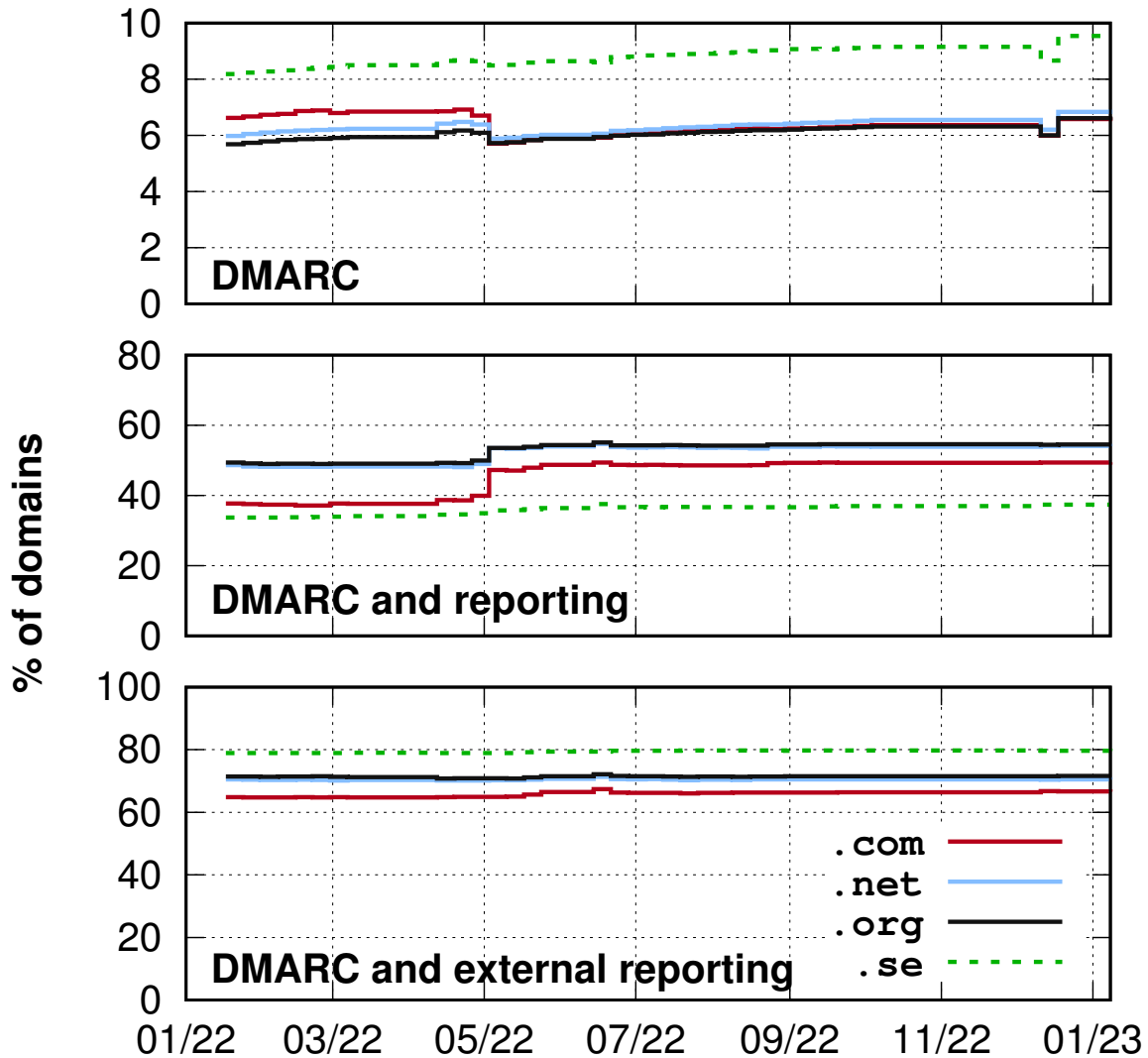


Figure 5.1: The deployment rate of DMARC records and their reporting feature from the dataset. More than 63% of DMARC records with reporting features forwards the report to external domains across the four TLDS.

5.1.2 Prevalence

We now examine how DMARC reporting has been deployed by domain owners by focusing on the number of second-level domains with MX records that (1) serve a DMARC record and (2) also have either a rua or ruf tag.

Figure 5.1 (top) plots the percentage of domains with MX records in .com, .org, .net, and .se second-level domains that have a DMARC record; We observe that the DMARC deployment rate for domains under .com is over 6% and 8% for .se domains. Domains under .se ccTLD have a higher adoption rate compared to the other three TLDS. Domains under .se ccTLD have a higher

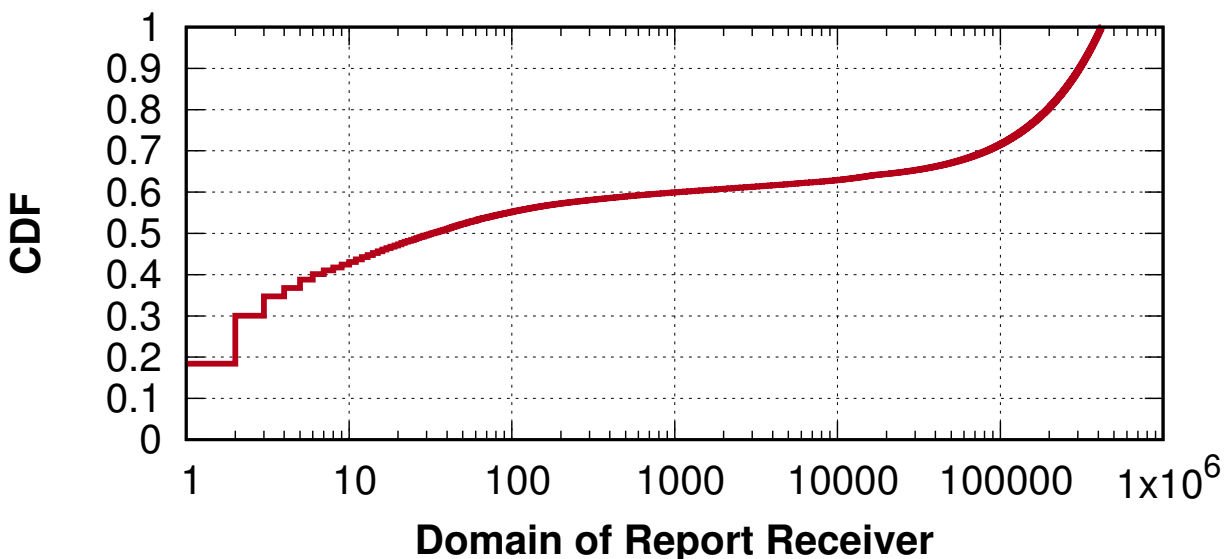


Figure 5.2: Only 7 domains receive the 43% of DMARC reports on behalf of domain owners.

adoption rate compared to other three TLDs; previous studies [34, 82] reported similar findings that `.se` (and `.nl`) domains show higher deployment rates of security protocols like DNSSEC and DANE. During our measurement period, we observe two incidents that impacted on the overall DMARC record deployment status; on May 3rd, 2022, 1.3 M domains managed by a DNS provider, `dan.com` retracted the DMARC records. These records did not have any reporting addresses and therefore, we see a spike in the corresponding graph for reporting (middle).

When focusing on the domains that also have a `rua` or `ruf` tag, we see that its deployment rate is over 35% across all four TLDs; for example, 55% of DMARC records in `.org` and `.net` listed email addresses to receive the report. Interestingly, we also find that the vast majority of them point to external domains, implying that domain name owners outsource third party organizations to receive and analyze the report; for example, a domain, `dmarcadvisor.com` accounts for 14% of the DMARC records in `.se` domains.

To deep investigate the skewness of the external domains that receive the reports, Figure 5.2 shows the number of DMARC records on which each domain appears; we find a highly skewed distribution, with only 7 domains receive the 43% of DMARC reports on behalf of domain owners while most of them (402K (96%)) domains only receive one domain.

We next look at how DMARC reporting is deployed as a function of domain popularity. Figure 5.3 shows the percentage of the domains with MX records in the Alexa top 1M domains that have DMARC records (top). Among them, we also show the percentage of the ones with `rua` or `ruf` tags (bottom). We first observe that popular domains tend to have DMARC records; these are more likely to have a `rua` or `ruf` tag, most of which are configured to use external domains. For example, 4,530 (45.3%) domains have DMARC records where 87% of them have `rua` or `ruf` tag among the 10,000 most popular domains while only 1,397 (14%) domains have DMARC records where 63%

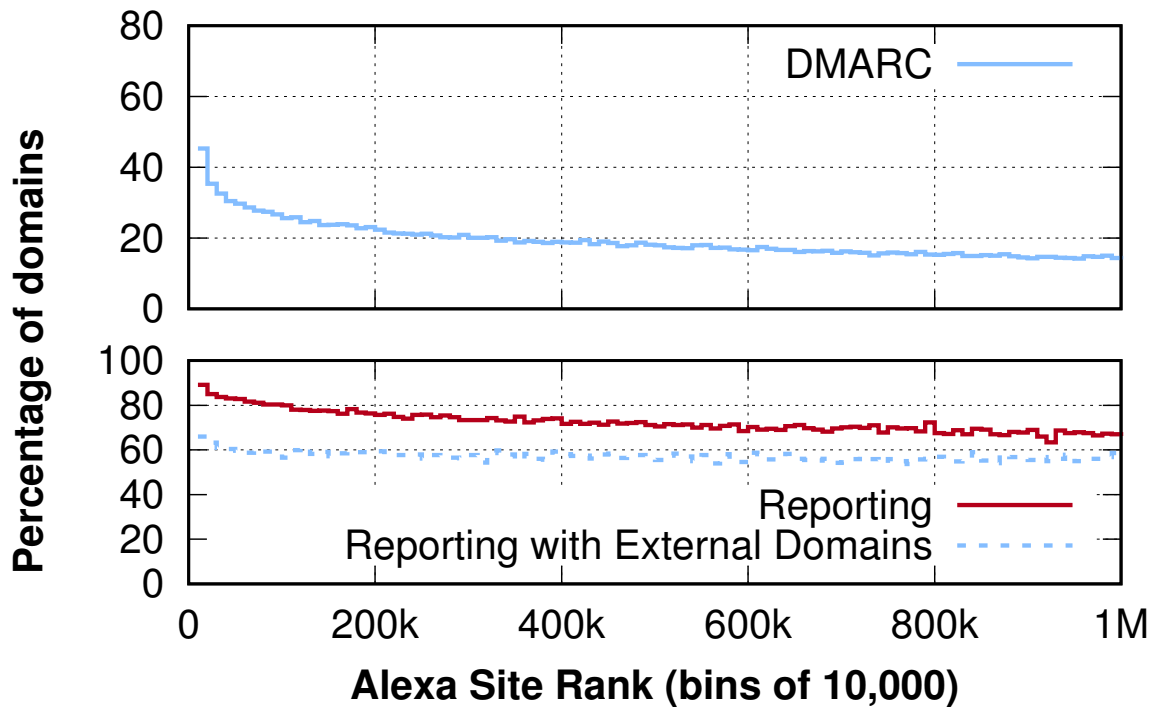


Figure 5.3: The percentage of domains with DMARC records, and their percentages with reporting feature as a function of domain popularity.

of them have the reporting tags among the 10,000 least popular domains. Interestingly, we find that most of DMARC records with a reporting feature use external domains to forward their reports *regardless of popularity* except the 30,000 most popular domains; around 59% of DMARC records with a reporting feature use external domains.

5.1.3 DMARC Reporting Compliance

We have observed that most of the DMARC records (around 60% across the rank) forward their reports to external domains. However, for external domains to receive the reports on behalf of the domain owners, they have to publish the corresponding DMARC authorization record as described in [subsection 2.2.3](#). For those DMARC records of which rua tags have external domains, we send additional DNS queries to their authoritative name servers to fetch the DMARC authorization records from our latest snapshot. We also calculate the percentage of the domains with external DMARC reporting that do not have the DMARC authorization record as a function of domain popularity in [Figure 5.4](#). We make a number of observations.

First of all, out of 2M DMARC records that have external domains, we find that 520K (26%) of them do not have the authorization DMARC records by getting NXDOMAIN or the responses that do not have the tag "v=DMARC1". When we look at the relationship between such misconfiguration

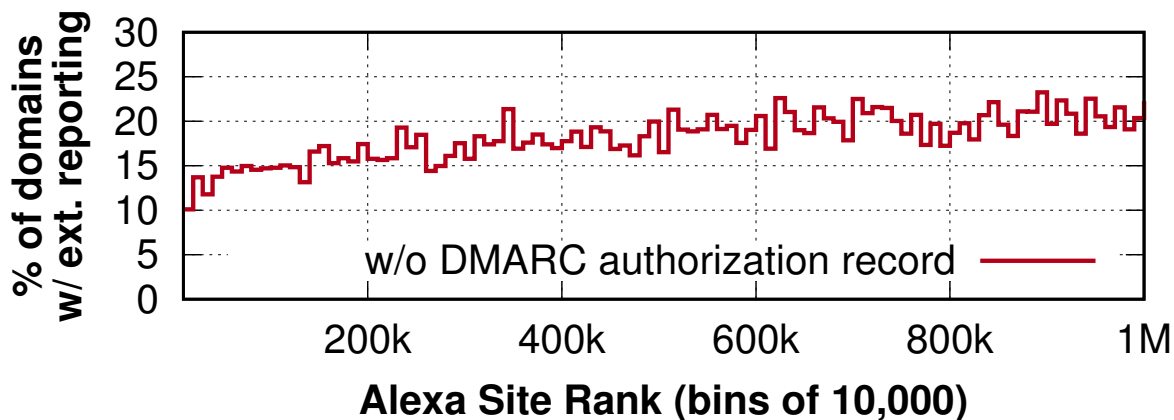


Figure 5.4: Less popular domains tend to be misconfigured when their DMARC records are configured to forward DMARC reports to external domains.

and the domain popularity, we find prevalent misconfiguration on even most popular domains; for example, 10% of domains that forward their DMARC reports to external domains do not have the matched authorization DMARC records.

Interestingly, we also observe an increasing trend of such misconfiguration as the ranking increases; for example, almost 20% of the least 10K popular Alexa top-1M domains with DMARC that use external domains are misconfigured compared to about 10% for the most 10K popular domains. These domains would not be able to receive any DMARC reports *from the report senders that perform EDV*; in the next section, we examine how popular DMARC report senders (i.e., email receivers) perform EDV.

5.2 DMARC Reporting Deployment for Inbound Emails

In the previous section, we analyzed the deployment of DMARC reporting and TLS-RPT for outbound emails. We found that 70% of SMTP servers with a DMARC record have rua tags pointing to external domains, and 26% of these domains are not configured correctly.

Now, we shift our focus to the SMTP servers that send DMARC reports for incoming emails. It is crucial for these report senders to be correctly configured from a security standpoint; for example, if the DMARC report sender fails to perform EDV, attackers can redirect all reports to a victim by changing their rua tags to the victim’s domain.

Characterizing the report senders without sending unsolicited emails is challenging. Therefore, we focus on the top email hosting providers that send DMARC reports and popular open-source software measured from Dmarcian’s DMARC Data Reporters [47], which lists popular email hosting providers that support DMARC reporting. Dmarcian receives and analyzes reports on behalf of domain name owners, and the top report senders are made publicly available.

No.	Name	Type	Rdata
Exp. 1	_dmarc.a.com	TXT	v=DMARC1; p=none; rua=mailto:admin1@a.com, ... , mailto:admin10@a.com
Exp. 2	_dmarc.a.com	TXT	v=DMARC1; p=none; rua=mailto:admin@a.com, ... , mailto:admin@a.com
Exp. 3	_dmarc.a.com	TXT	v=DMARC1; p=none; rua=mailto:admin@s1.a.com, ... , mailto:admin@s10.a.com
Exp. 4	_dmarc.a.com	TXT	v=DMARC1; p=none; rua=mailto:admin@a.com, mailto:admin@b.com
Exp. 5	_dmarc.a.com	TXT	v=DMARC1; p=none; rua=mailto:admin@a.com, mailto:admin@b.com
	a.com._report._dmarc.b.com	TXT	v=DMARC1; rua=mailto:admin1@b.com, mailto:admin2@b.com
Exp. 6	_dmarc.a.com	TXT	v=DMARC1; p=none; rua=mailto:admin@a.com
	_smtp._tls.a.com	TXT	v=TLSRPTv1; mailto:admin@b.com
	_dmarc.s1.a.com	TXT	v=DMARC1; p=none; rua=mailto:admin@s1.a.com ri=86400
Exp. 7	_dmarc.s2.a.com	TXT	v=DMARC1; p=none; rua=mailto:admin@s2.a.com ri=86400
	_dmarc.s3.a.com	TXT	v=DMARC1; p=none; rua=mailto:admin@s3.a.com ri=86400

Table 5.2: The configurations of DMARC records for experiments to characterize DMARC report senders

We selected the top 7 email hosting providers and included recent popular providers that support DMARC reporting: Google Workspace, Amazon, Fastmail, etc. The list of the email hosting providers is shown in [Table 5.3](#). It is worth mentioning that this list is distinct from the list of popular email *service* providers, as some providers only send DMARC reports when domain owners purchase a specific plan to outsource their email services. Additionally, we also tested the two most popular open-source DMARC software, OpenDMARC, and Rspamd, as per our survey, which will be described in detail in [section 7.1](#).

5.2.1 Methodology

The purpose of these experiments is to identify the characteristics of report senders. To achieve this, we purchase a domain name, `a.com`, for our DNS authoritative server and SMTP server, which will serve as the email sender. We use BIND [1] for the authoritative server and Postfix [101] for the SMTP server. Additionally, we purchase another domain name, `r.com`, and delegate its SMTP server to a third-party email hosting provider, which will receive emails from `a.com` and examine its DMARC record to send reports.

We then proceed as follows:

1. We select an email hosting provider and subscribe to their hosting plan. The DNS settings for `r.com` are configured to delegate its MX record to the email hosting provider. When testing email software, we run a Postfix SMTP server at `a.com` and DMARC reporting software.
2. The DMARC record for `a.com` is configured according to one of the experiment scenarios outlined in [Table 5.2](#), which will be described in more detail later.
3. Emails are sent to `r.com` from `a.com`.
4. All incoming DNS queries are recorded and all reports are stored.

5.2.2 Characterization

To characterize the behavior of report senders, we configure the DNS authoritative servers to serve a different DMARC record for each experiment. A DMARC record can define a variety of rules using multiple tags [74], but we specifically focus on `rua` and `ri` tags, which can be exploited by attackers; more specifically, we focus on whether attackers can use them as *reflectors* to generate a massive number of reports to a victim within a very short time window.

We concentrate on four distinct perspectives of implementation vulnerabilities that can be taken advantage of by attackers: (1) *RFC Ambiguity* where the standard lacks specific implementation details, (2) *RFC Exploit* where the implementation is accurate but can still be exploited (3) *Misconfiguration* where the software improperly implements the RFC, and (4) *others*. To this end, we consider the DMARC record configurations as shown in [Table 5.2](#):

RFC Ambiguity

Multiple destinations (i.e., email addresses) can be specified in the `rua` or `ruf` tag; the attackers can easily increase the attack amplification factor by putting multiple email addresses in the tags.

Exp. 1: Multiple report recipients Since the RFC does not limit the number of allowed email addresses in the `rua` tag, theoretically, the attackers can put many addresses as they wish to increase the amplification factor as long as it fits to a DNS response. The attackers can first estimate the limit by specifying multiple addresses belonging to them.

Exp. 2: Duplicated email addresses Extending the first experiment, we intentionally put the same email address multiple times; if the report sender does not merge them together, attacker can easily put a single well-known email address such as `postmaster`.

Exp. 3: Subdomain de-aggregation This scenario examines whether a report sender merges subdomains into one. If the report senders send a report to a unique domain, this can be used to amplify the number of reports.

Misconfiguration

Exp 4. External destination verification We check whether the report sender performs External Domain Verification (EDV) as suggested in the RFC document [74]. In order to do so, we purchase another domain, `b.com`, which also run an SMTP server. We configure our DMARC records to forward the report to both `a.com` and `b.com`. After sending an email from `a.com`, we check

whether the external domain also receives the report; if so, the report sender can be exploited to make the victim receive unwanted reports.

RFC Exploit

Exp. 5: DMARC EDV overriding DMARC authorization records can also include a `rua` tag, which allows reports to be redirected to multiple email addresses within the same domain. However, this tag can also be used to amplify the number of reports by using multiple email addresses in the `rua` tag. For example, `a.com` in [Figure 2.2](#) can have multiple email addresses (e.g., n), each of which points to different external domains. These external domains can put multiple email addresses in their `rua` tag in their DMARC authorization records; this results in an exponential increase in the number of reports received, as the EDV process is performed for each external domain, causing the victim to receive n^2 reports.

Exp 6. TLS-RPT external destination verification Unlike a DMARC report, a TLS-RPT report is produced by an *email sender* and recommended to another party that does not share the same TLS configuration with the email receiver. Thus, TLS-RPT does not have any external domain verification mechanism, which attackers may exploit; attackers can use an SMTP server that supports both TLS-RPT and DMARC reports as reflector. For example, it can make the SMTP server to send DMARC reports to the attackers, but redirecting them to the victim by deliberately causing TLS errors when they make STARTTLS connections to send DMARC reports. This will be detailed in [5.3.1](#). With Exp. 6, we perform whether the SMTP servers that send DMARC report also support TLS-RPT.

Others

Exp. 7: Synchronous report generation If the attackers can leverage multiple report senders to make them send multiple reports *simultaneously*. The report sender may respect the `ri` value on the DMARC record and send the report exactly after the value. Alternatively, the report sender may ignore the `ri` value, but generate and transmit reports to all report receivers on its schedule. To detect this, we send one email each from different subdomain one hour apart; if all the reports arrive simultaneously (or within a very short time window), it could be a strong signal that they do not respect `ri` values and manage their own clock. We also run this experiment five times to predict the next report generation time.

5.2.3 Experiment Results

From the experiments, we mainly focus on the implementations that attackers can potentially exploit. [Table 5.3](#) shows the results.

Limit of the email addresses in the rua tag: Surprisingly, we find that only Fastmail imposes a limit of 10 email addresses; note that we only list 50 email addresses, thus the other providers may have a higher limit, which attackers can exploit to increase their amplification factor. We also find that OpenDMARC has a hard limit of 255 bytes for the DMARC records, thus the records that exceed the limit raise errors. rspamd does not have any limit, thus any number of email addresses that the DNS response size permits is allowed.¹

Report aggregation: We find that 6 out of 7 email hosting providers do not attempt to aggregate the email addresses in the rua tags; for example, Google, Yahoo, and QQ send n duplicated emails to the same sender. Even worse, all these three providers do not implement EDV mechanisms, thus attackers can exploit these providers to send n emails to the same address. 5 providers do not attempt to aggregate the reports by domains and subdomains, which can be potentially exploited by attackers. OpenDMARC also does not attempt to aggregate the reports while rspamd does so.

EDV implementation: We find that Google, Yahoo, QQ, and Gmail, and OpenDMARC do not implement EDV, which raise a serious concern; attackers can put arbitrary email addresses on their domains and simply email these hosting providers so that they can send more than 50 emails to the victim. Amazon, Fastmail, 163, and rspamd implements EDV correctly while only Fastmail implements the EDV overriding.

Report sending time: We find that *all* email hosting providers do not respect the r_i value; each provider manages their own clock to send reports simultaneously; for example, we observe that all reports we receive from Google come at midnight (ET). For both software, we also confirm that they do not respect r_i , but their default configuration is to send all DMARC reports at midnight by using crontab.

5.3 Report Reflection (RR) Attack

Here, we introduce how email receivers that send DMARC report can be misused to mount new attacks. As shown in the previous section, some of them do not perform EDV correctly (due to their misconfiguration) and do not limit the number of email addresses in the rua tag (due to the ambiguity of the RFC standard). These two vulnerabilities are the core of our attack; attackers can use them to make the victim (1) receive a massive volume of reports or (2) establish a large number of TCP connections within a very short time window. Our attacks both exploits the misconfigured SMTP servers and the ambiguity of the RFC standards.

¹It depends on the resolver that rspamd uses. If the resolver supports EDNS0 and retry over TCP, more than 2,000 email addresses can be listed (assuming that each email address takes up 20 bytes).

EHP	Report Size (B)	# of addr.	EDV		Email Address Aggr.			Respect	RI Predictable	RUF Support
			Check	Overriding	Addr.	Domain	Subdomain			
Google	3,962	50	X	-	X	X	X	X	✓	X
Amazon	4,324	50	✓	X	✓	X	X	X	X	X
FastMail	4,839	10	✓	✓	✓	X	X	X	✓	X
Yahoo	4,626	50	X	-	X	X	X	X	✓	X
163	4,034	50	✓	X	✓	X	X	X	✓	X
QQ	3,628	50	X	-	X	X	X	X	✓	X
Gmail	3,962	1	X	-	✓	✓	✓	X	✓	X
Comcast	3,337	-	✓	-	-	-	-	X	X	-
Microsoft	8,926	50	✓	X	X	X	X	X	X	X
OpenDMARC	2,238	8-12 ⁴	X	-	X	X	X	X	✓	✓
Rspamd	2,320	50	✓	X	✓	X	X	X	✓	X

Table 5.3: Table showing the 7 popular email hosting providers and two softwares that support DMARC reporting. If email providers do not implement EDV, we do not test whether they override EDV or not (hence the -). In case of Gmail, it only sends one report to the first recipient listed in the rua tag.

EHP	Report Size (B)	Email Address Aggr.			Predict.
		Addr.	Domain	Sub.	
Google	5,839	✓	✓	X	✓
Comcast	5,094	X	✓	✓	✓

Table 5.4: Table showing the two email hosting providers (EHPs) support TLS-RPT reporting; the other providers (Amazon, FastMail, Yahoo, 163, QQ, and Gmail) do not support it. We could not find any opensource software for TLS-RPT.

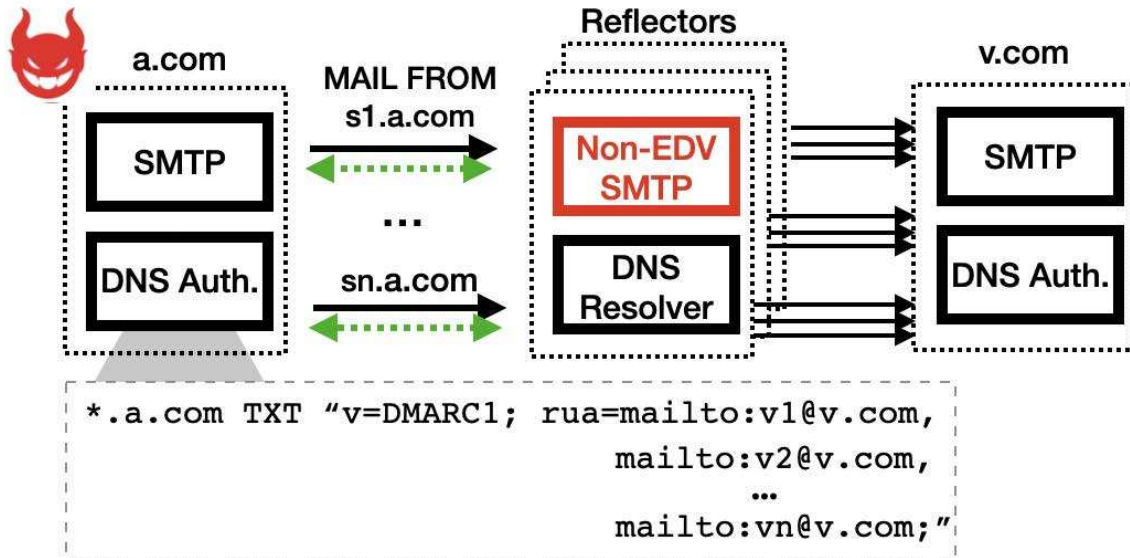


Figure 5.5: Attack exploiting SMTP servers that do not perform EDV correctly.

5.3.1 Attack 1: Inbox flooding

When email receivers do not implement EDV mechanism, attackers can exploit them as reflectors and make the victim receive multiple reports. Even if email receivers have EDV mechanisms in place, the attackers can still launch an attack by using TLS-RPT rua tags with the victim's email address.

Email receivers without EDV

Email receivers that do not implement EDV are vulnerable to exploitation as illustrated in [Figure 5.5](#). The attackers can add multiple email addresses hosted in the victim's SMTP server in their rua tag and send an email to the reflectors. As shown in [Table 5.3](#), the report size (R) is different across the email hosting providers, all of which are bigger than what is required to send an email to a reflector (approximately 200 bytes to transmit considering SMTP transactions of HELO, MAIL FROM, RCPT TO, DATA commands along with a short email content). If the email receiver does not aggregate the reports or limit the number of email addresses in the rua tag (M), the attackers can easily increase the number of reflected reports. As shown in [Table 5.3](#), the report size (R) is different across the email hosting providers, all of which are bigger than 200 bytes. Thus, the amplification factor (F) for this attack would be $F = \frac{R}{200} \times M$; Google, Yahoo, and QQ are susceptible to this attack and their amplification factors are $950\times$, $1150\times$, and $900\times$ respectively with $M = 50$. This attack can be further amplified if the email receiver also supports TLS-RPT, which is explained in detail in [section 5.3.1](#).

Email receivers using OpenDMARC are also vulnerable ($F = 110\times$), as OpenDMARC does not implement EDV nor aggregate the reports. Additionally, attackers can easily identify email servers using OpenDMARC as it puts the string, 'opendmarc-reports' and its version number in the 'X-Mailer' header of every report generated.

Email receivers with EDV

Even if email receivers implement the EDV mechanism, they can still be exploited as reflectors if they support TLS-RPT. When a TLS error occurs during the STARTTLS negotiation due to validation errors with DANE [\[67\]](#) or MTA-STS [\[85\]](#), the email sender will send a TLS-RPT report. Unlike DMARC reports, the EDV mechanism is not present in TLS-RPT, allowing attackers to redirect the report to any email address.

In this attack scenario, the attacker puts email addresses under the same domain in the rua tag so that the reflector sends a DMARC report to the attacker. The attacker then causes a TLS error during the STARTTLS negotiation by sending a malformed certificate. The reflector, if supporting TLS-RPT, will fetch the attacker's TLS-RPT records, which point to the victim's email address.

⁶OpenDMARC restricts DNS records to a maximum of 255 characters.

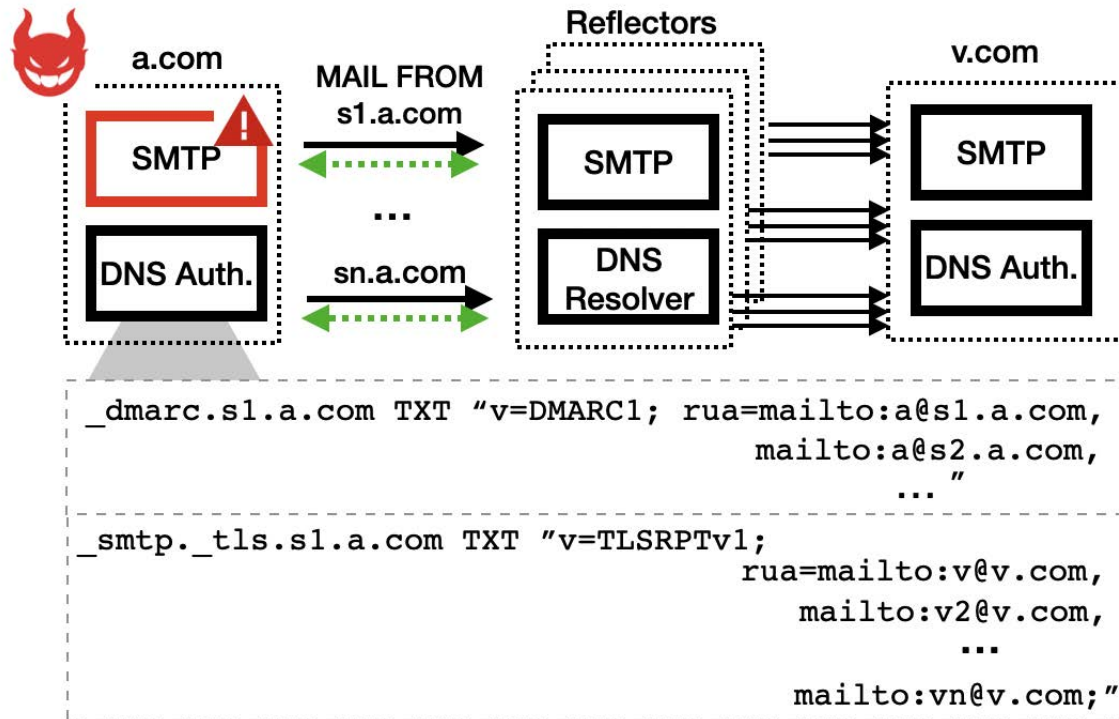


Figure 5.6: Attack exploiting SMTP servers that support both DMARC and TLS-RPT reporting. The attacker intentionally provides an *invalid* certificate to reflectors when they send a report to the attacker. This triggers them to send a report to the victim as specified in the rua tag of the TLS-RPT record.

Since TLS-RPT allows for any email addresses to be listed, even those outside the domain's authority, the reflector will send reports to the victim. The attack scenario is illustrated in Figure 5.6. Assuming the number of permitted email addresses in both the DMARC rua tag and TLS-RPT rua tag is n , the attacker can make the reflector send n^2 emails to the victim. The amplification factor becomes $F = \frac{R}{200} \times m^2$ where m is the max no. of email addresses in both DMARC and TLS-RPT rua tag.³ We find that Google is susceptible to this attack; however, it aggregates TLS-RPT reports that share the same recipient address. Thus, their amplification factor is $1,460 \times (= \frac{5,839}{200} \times 50)$.

DMARC EDV overriding: Attackers can amplify the number of reports further by using DMARC rua tag overriding. An email receiver retrieves the DMARC authorization record when it encounters an external domain in the rua tag. This authorization record can also have a rua tag, which over-

³Astute readers may notice that the formula does not take into account the cost associated with the TLS handshake; attackers can trigger TLS-RPT reports by intentionally not supporting STARTTLS while serving TLSA records [41] or MTA-STS records [85];

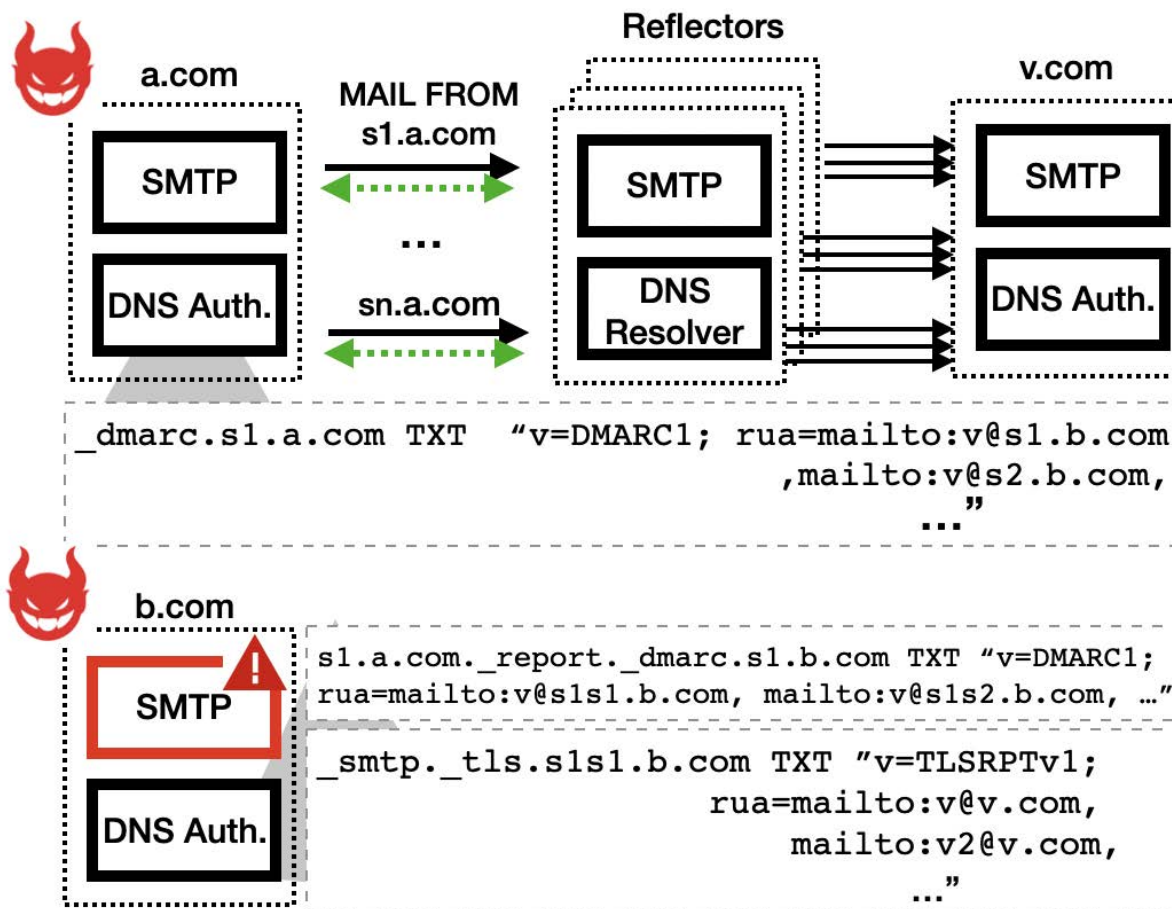


Figure 5.7: Attack exploiting the rua tag override; the reflector finds a rua tag, which contains a list of external domains. For each domain, it performs EDV and fetches the DMARC authorization record, which also contains a list of email addresses in the rua tag; a TLS-RPT report will be sent to each address.

rides the email address, which replaces a single email address in the original rua tag with all email addresses listed in the rua tag of the DMARC authorization record.

In this attack, the attacker, **a.com**, puts n email addresses under the domain **b.com**, which is managed by the attacker. Since **b.com** is an external domain, the reflector performs EDV and retrieves its DMARC authorization records from **b.com**'s name server. The attacker can override each email address in the rua tag with n additional email addresses, all of which are subdomains of **b.com** (e.g. **s1s1.b.com**). As in the previous attack, the attacker's SMTP server intentionally breaks the TLS, causing the reflector to send TLS-RPT reports to the victim. The attacker can then include additional n email addresses in the rua tag, generating n^3 reports; in our experiment, however, we could not find any email hosting providers vulnerable to this attack.

5.3.2 Attack 2: SMTP connection flooding

In a previous attack, attackers could exploit email receivers that either do not support EDV or support both EDV and TLS-RPT to cause the victim to receive reports. However, as shown in [Table 5.4](#), there are only 2 popular email hosting providers and no open-source software that support TLS-RPT at this moment. Therefore, our focus now shifts to email receivers that do not support TLS-RPT, and we introduce a SMTP connection flooding attack, which causes the reflectors to initiate a large number of TCP (and SMTP) connections to the victim.

All SMTP connections are built on top of TCP connections. Once the SMTP connection is established, two SMTP servers communicate with each other by exchanging SMTP commands. The sender typically sends the MAIL FROM and RCPT TO commands, and it can transmit the actual email content after receiving an “OK” response (250) from the receiver through the DATA command. An email receiver typically rejects an email if the address in the RCPT TO command is not associated with itself, for security purposes, by emitting a 5XX error code.⁴

This means that the email receiver (i.e. report sender) *must first establish a SMTP connection, regardless of the email destination*, which is the core of our second attack. Attackers can place many email addresses they control in the rua tag and *configure the MX record of their domains to point to the victim, causing the reflectors to initiate SMTP connections to the victim*. Additionally, the victim may *block* the incoming IP addresses of the reflectors, not allowing further SMTP connections due to the rate limit on incoming TCP connections on firewall [59].

Reflectors with EDV: Attackers can add n email addresses of their domain in the rua tag, all of which MX record maps to the victim’s MX record, which leads to the n amplification factor. Moreover, similar to [Figure 5.7](#), attackers can leverage DMARC EDV overriding by putting additional n email addresses in the rua tag of the DMARC authorization record. As shown in [Figure 5.8](#), attackers can add a wildcard MX record to map all subdomains to the victim’s MX record, `smtp.v.com` so that the reflector initiates n^2 SMTP connections to the victim with a single email. Since the only email hosting provider that implement rua tag overriding is FastMail; fortunately, we confirm that FastMail limits the total number of reports to 10 in total, thus the rua tag overriding does not further increase the amplification factor. However, since the rua tag overriding is necessary for EDV mechanism in the standard [74], attackers may be able to find such reflectors from their scanning.

⁴However, email receivers may accept an email if they are configured to support SMTP relay [73] (also known as open mail relay), which accepts all emails and forwards them to external recipients. This can be abused by spammers, thus typically disabled.

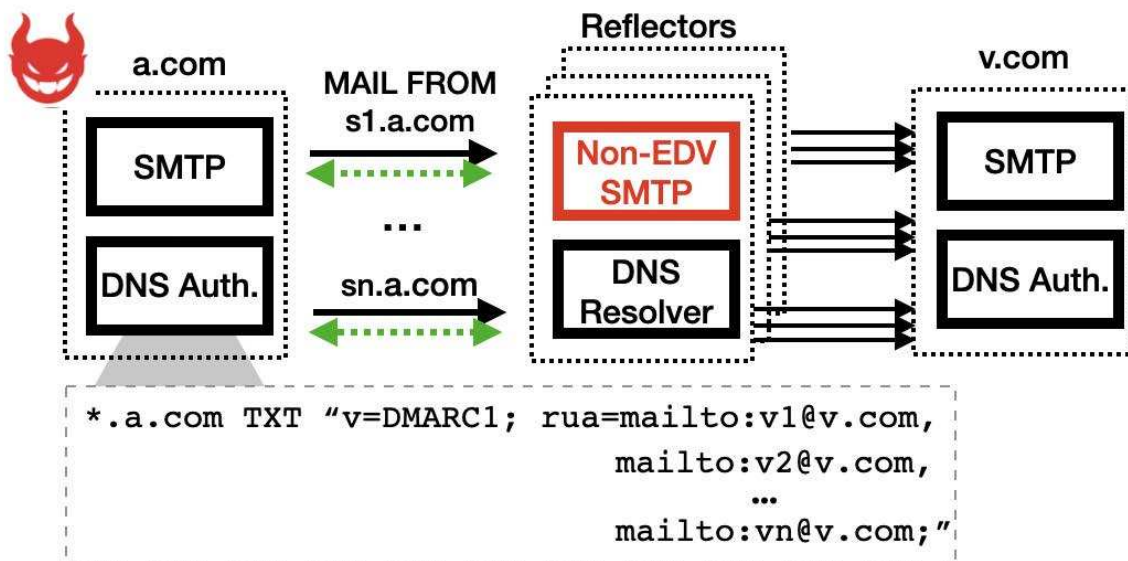


Figure 5.8: Attack exploiting the rua tag override. Since the TLS-RPT is not supported by the reflector, the attacker can host a wildcard MX record that maps to the victim's MX record, which makes the reflector initiate n^2 SMTP connections.

5.3.3 Summary

We have found that attackers can improve their attack efficiency (i.e., amplification factor) by putting multiple email addresses on the rua tag, redirecting TLS-RPT reports, and rua tag overriding. We also have introduced that attackers can induce the reflectors to initiate many SMTP connections to the victims by simply redirecting MX records to the victim's SMTP server.

Additionally, the following technique would not impact on the efficiency of the attack (i.e., amplification factor), but they can contribute to the attack;

- Attackers can utilize multiple reflectors to launch a DDoS attack; for example, the attackers can only focus on the reflectors that use OpenDMARC, which can be easily fingerprinted by extracting the report email headers.
- All email hosting providers and open-source software do not respect r_i values; however, their sending time can be easily measured and predicted, thus attackers can only use the reflectors that send out the reports on a given time.

Responsible Disclosure: Since January 2023, we have disclosed our findings to all the email hosting providers we tested and filed a bug report to the repository of the relevant open-source

software. Despite our efforts, only Google and Yahoo have responded back to us. Google classified this bug as P3-level while Yahoo dismissed our bug report as non-critical.

TLD	Measurement Period	Domains with MX Records	
		Number	Percent with MTA-STS
.com	09/20/2021 – 29/09/2024	73,939,004	53,800 (0.07%)
.net		6,248,969	6,183 (0.09%)
.org		5,781,423	7,355 (0.13%)
.se		822,449	692 (0.08%)

Table 6.1: Overview of our datasets; the number of the domains with MTA-STS records are as-of September 29th, 2024.

Chapter 6

Unraveling the Challenges of MTA-STS Deployment

In this part of the dissertation, we shift our focus to email transport security. Original design of SMTP lacked built-in encryption mechanisms, making it vulnerable to various security threats. Even with the later introduction of opportunistic TLS into SMTP via the STARTTLS [66] extension, it allowed for downgrade attacks, where an attacker strips the STARTTLS command from an intercepted session, preventing the use of encryption [30, 64].

To mitigate these attacks, DANE (DNS-based Authentication of Named Entities) [41] has proven to be a robust protective measure. DANE relies on DNSSEC-signed [11, 12, 13] TLSA records, securely mapping a mail server’s public key to its domain and verifying the recipient’s intent to engage in TLS encryption. Despite its advantages, DANE adoption is hindered by its dependency on DNSSEC, which maintains a low global implementation rate (around 4% [62]).

To circumvent these challenges and assure SMTP transport-level encryption, MTA-STS (Mail Transfer Agent Strict Transport Security) [85] has been introduced. MTA-STS wards off STARTTLS downgrade or traffic interception attacks by permitting recipient domains to declare MX host(s) with valid PKIX certificates through DNS and HTTPS. Additionally, MTA-STS outlines a policy specifying how compliant senders should proceed if secure TLS setup cannot be established. Leading email providers, such as Google [71] and Microsoft [57], employ and enforce MTA-STS for outgoing mail.

However, MTA-STS also introduces *additional complexity*. Not only do domain owners need to configure DNS records to publish MTA-STS policies and include the "_mta-sts" TXT record, but *they also need to run a web server to serve the policy file over HTTPS*, adding an additional service. To address this complexity, third-party operators have emerged to handle policy file hosting on behalf of domain owners, without necessarily being involved with mail transport; however, this setup requires domain owners to correctly configure CNAME records to allow these third-party operators to serve the policy using valid TLS certificates. Furthermore, the mx patterns specified in the policy file must match the actual MX records for the domain.

In this work, we focused on the deployment and management challenges associated with MTA-STS. Our findings revealed that while MTA-STS offers a promising solution to improve email security, its practical implementation often leads to misconfigurations and inconsistencies due to the complex setup process often involving multiple parties. Consequently, such problems can lead to email delivery failures [103] and potential downgrade attacks when senders revert to opportunistic encryption due to validation failures [61]. Overall, our contributions in this work are as follows [14]:

- We provided a comprehensive taxonomy of the prevalent MTA-STS misconfigurations in the wild, revealing (RQ1).
- We found a significant proportion (29.6% of domains with MTA-STS records in our latest scan) to have faulty MTA-STS setups, with 640 (3.2%) domains susceptible to email delivery failures from MTA-STS compliant senders (RQ2).
- Our sender-side dataset revealed a much wider adoption of MTA-STS in the client side with 19.6% of domains performing MTA-STS validation when sending email to an MTA-STS enabled domain (RQ1).
- As with sender authentication, we surveyed email administrators and policy providers to understand the practical landscape and operational reality of MTA-STS (RQ1).

6.1 Measuring MTA-STS Deployment

We begin by examining the deployment of MTA-STS, focusing specifically on domains that have implemented MTA-STS records.

6.1.1 Datasets

To span a wide array of registered domains, our methodology employs DNS scans across four TLDs: the .com, .org, and .net gTLDs, and the .se ccTLD. We choose these three gTLDs for their widespread adoption [118]. Meanwhile, the chosen ccTLD is notable for: (1) its proactive

promotion of security protocols through registry-backed financial incentives [52], and (2) its openly available DNS zone files, providing a valuable dataset for research.

For each TLD, we acquire daily zone files from their registries (.com and .net from Verisign, .org from Public Internet Registry, .se from Internetstiftelsen). Subsequently, for every SLD, we retrieve the MTA-STS records. Similarly, we also collected the MX, and NS records of these domains. To mitigate the risk of overloading small DNS authoritative servers due to frequent DNS queries, we opt for weekly snapshots of the MTA-STS records for each domain and rate limit our queries. We collected data over 36 months, from September 9, 2021, to September 29, 2024, see [Table 6.1](#).

6.1.2 MTA-STS Deployment

Overall we find a limited adoption of MTA-STS, while adoption slowly starts to accelerate from 2023 onward, see [Figure 6.1](#). Initial adoption in 2021-10 ranged from 12,148 (0.02%) for .com to 1,916 (0.03%) domains for .org. As of 2024-09, we find adoption to have risen 3-4 times, with adoption ranging between 53,800 (0.07%) domains for .com and 7,355 (0.12%) domains for .org.

Although the deployment ratio is modest, it has been accelerating as popular email service providers have started to support MTA-STS. These providers include Google [71], Microsoft Outlook [57], Yahoo [130], and Mail.com [92]. Note that, unlike with DANE where a single operator adding TLSA records enables DANE for all domains using that MX, MTA-STS requires *each individual domain owner* to take action. We attribute the more gradual adoption curve compared to DANE [82] to this mechanic.

Next, we examine whether popular domains are more likely to deploy MTA-STS. [Figure 6.2](#) shows the percentage of the domains with MX records in the Tranco top 1M domains in .com, .net, .org, and .se that also publish MTA-STS records, as of November 1, 2024. We first observe that popular websites are more likely to have MTA-STS records, but the overall MTA-STS deployment remains low even among the most popular domains. For example, the average percentage of domains with MTA-STS records among the top 10,000 popular domains is 1.2%, while that of the bottom 10,000 popular domains is 0.4%, which suggests that although there is a positive correlation between website popularity and MTA-STS adoption, the deployment rate is still relatively low across all popularity ranges.

However, as discussed in the background, simply deploying an MTA-STS record does not mean correct deployment; we have to fetch the policy file from the HTTPS server, and the SMTP server also has to provide a PKIX-valid certificate. The presence of an MTA-STS record alone does not guarantee that MTA-STS is properly configured and operational.

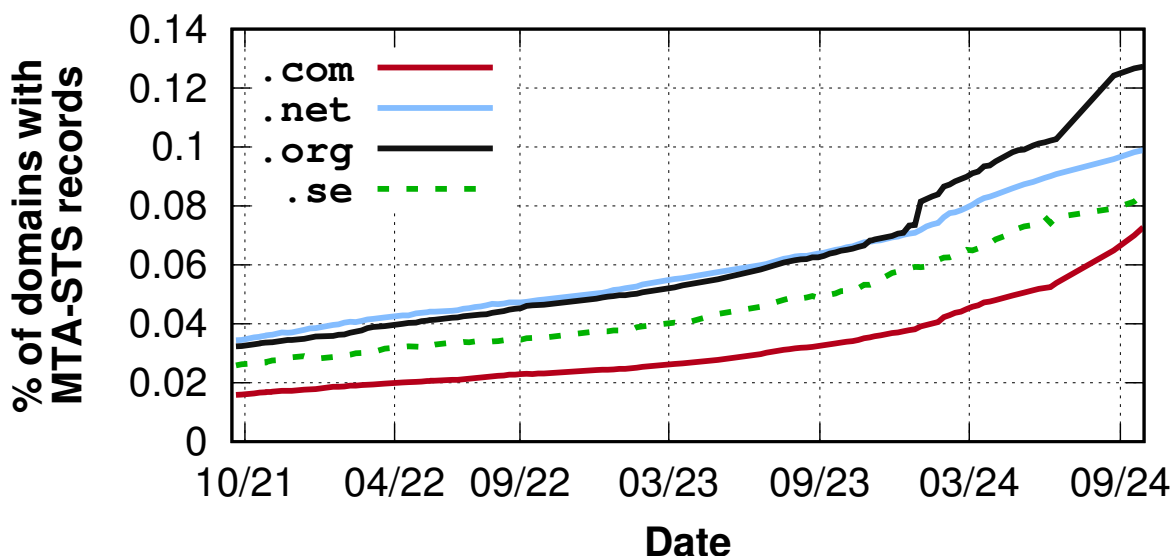


Figure 6.1: The deployment of MTA-STS records; 461 new domains related to the same organization under .org adopted MTA-STS in Jan 2, 2024 causing the notable spike.

6.2 MTA-STS Management

For an effective deployment and management of MTA-STS, a domain owner is required to: (1) publish an MTA-STS record on their name server, (2) issue a valid policy on their web server over HTTPS, and (3) provide a PKIX valid certificate from their MX hosts that match the patterns listed in the policy file. We now investigate whether domains with MTA-STS records actually fulfill these requirements.

6.2.1 Datasets

In this section, our aim is to delve into the *correct* deployment and functioning of MTA-STS among domains with MX and MTA-STS records. Therefore, in addition to the metrics of the dataset discussed in [section 6.2](#), we now also started to collect the following data:

1. For each domain, the MTA-STS policy retrieved via HTTPS from the well-known policy domain URI (e.g., `mta-sts.example.com`).
2. We connect to each MX for every domain using an instrumented SMTP client that:
 - (a) Connects to the server using SMTP from a host with correctly configured forward-confirmed reverse DNS (FCrDNS) set up.

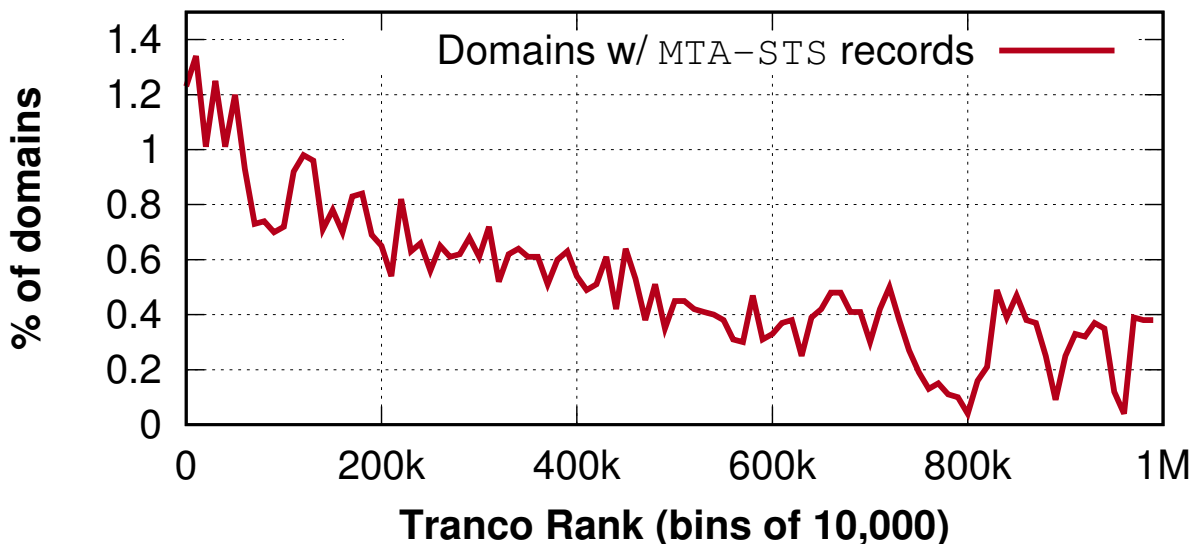


Figure 6.2: The percentages of domains publishing both MX and MTA-STS records as a function of website popularity based on Tranco 1M [99]; more popular websites are more inclined to deploy MTA-STS for their email services.

- (b) Issues an EHL0¹ with a name matching the reverse DNS to check for the STARTTLS capability²
- (c) Issues the STARTTLS command to transition the SMTP connection to TLS and retrieve the server's certificate.
- (d) Ends the connection *without* attempting to deliver an email.

This methodology was applied on a monthly basis from Nov 7, 2023 to Sep 29, 2024.

6.2.2 MTA-STS Validation

MTA-STS relies on several components to be implemented and in sync, recall [subsection 2.3.4](#). However, this also means that any error in any component may cause the entire MTA-STS validation to fail. Individual errors can occur in each component of MTA-STS including (1) MTA-STS records in DNS server (`_mta-sts.` and the delegation to the policy server on `mta-sts.`), (2) the policy file on the web server, or (3) the MXes themselves.

[Figure 6.3](#) shows the percentage of misconfigured MTA-STS enabled domains over time in four categories: (1) invalid MTA-STS records, (2) policy retrieval errors, and (3) PKIX-invalid MX hosts,

¹Falls back to HELO if EHL0 is unsupported.

²Some servers may not advertise STARTTLS due to greylisting [128] or CAPTCHAs; therefore, we focused on MXes that support at least some form of TLS for further analysis.

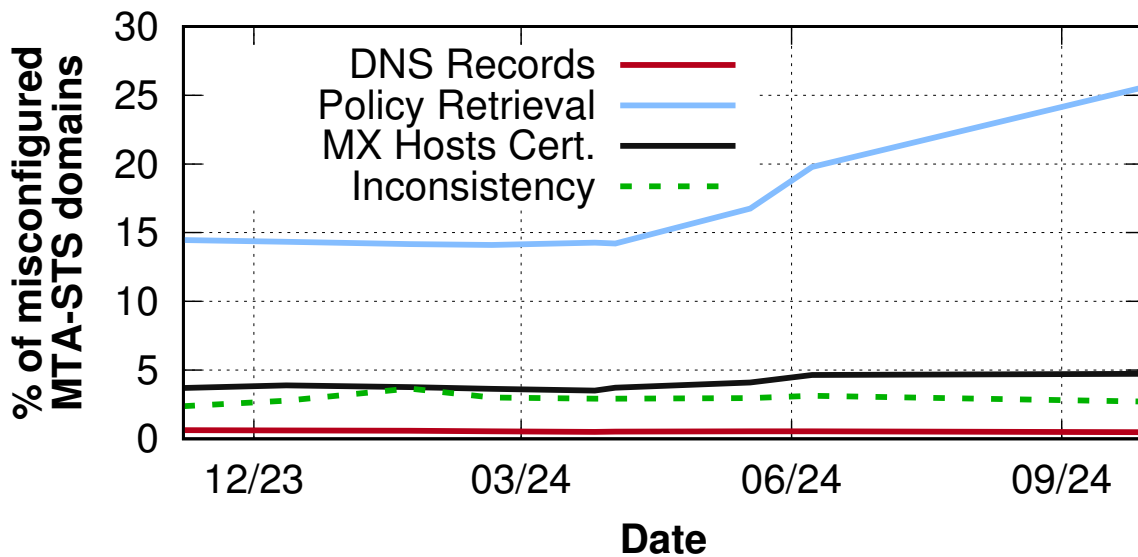


Figure 6.3: Percentage of MTA-STS enabled domains with errors in (1) MTA-STS records, (2) policy retrieval over HTTPS, (3) certificate of MX hosts, or (4) policy text where mx patterns of the policy file and MX records are inconsistent. Since August 2024, newly registered domains under Porkbun LLC [100] with invalid policy host certificates have increased policy server errors, affecting 7,237 domains in our latest snapshot.

which are individual, and (4) inconsistency errors where each component looks valid, but the mx patterns in the policy file do not match any MX records. Interestingly, we find a large portion of MTA-STS enabled domains having errors across all categories. For example, in our most recent snapshot, we find that among the 68,030 domains that have an MTA-STS record, 20,144 (29.6%) domains are incorrectly configured. Note that these errors are not exclusive, i.e., a domain may have multiple errors at the same time.

In the next section, we focus on each individual error and the inconsistency error to better understand why the errors occur and identify the most challenging aspects of correct MTA-STS deployment and management.

6.2.3 Misconfigurations: Individual Components

MTA-STS requires all components—the DNS authoritative server, MX hosts, and the policy server—to *correctly* adhere to their designated responsibilities. In this subsection, we explore misconfigurations in each of these components individually.

Self-hosted vs. Third-party Hosting

In the email ecosystem, both self-hosted solutions (e.g., Postfix [101]) and third-party hosting providers (e.g., Tutanota) are common. Management is simpler when the domain, MXes, and policy server are controlled by the same entity, as in fully self-hosted setups. In contrast, when using large email service providers, domain owners often *split* these components among multiple entities.

To investigate whether the DNS server, MX hosts, and policy server belong to the domain owner (i.e., self-managed) or a third party, we must rely on publicly available DNS records (NS, MX, A, AAAA). Drawing on the approach of prior work [81], we adopt a heuristic that leverages the popularity of hostnames and IPs.

1. *Heuristic 1: Identifying Third-Party Hosting.* We label an entity as third-party if it operates mail or DNS infrastructure for many domains (≥ 50)³ For example, Google hosts SMTP for over 11 million (5.8%) of domains in our dataset, and DMARCReport hosts policy services for 7,293 domains.

However, a complication arises when an administrator single-handedly manages multiple domains but does so using identical or nearby IP addresses. For example, 4,722 domains in our sample point their MX record to `mx.1.mxascen.com`, store policy files at `95.111.215.165` and `209.50.60.142` (both owned by the same operator), and share a uniform A record at `194.113.75.102`. Although these IPs appear “popular”, they actually belong to a single administrators self-hosted environment. To capture this nuance, we group domains by their MX records, A or AAAA entries, and policy-hosting IP addresses, labeling them as self-managed if they appear to be under a single administrator’s control.

Another exception arises, when third-party email providers assign unique hostnames to each customer yet point them all to same set of IPs. Therefore, we also consider A and AAAA records of MX entries to measure popularity for email hosting providers.

2. *Heuristic 2: Identifying Self-Managed Hosting.* Classifying self-managed domains solely by low popularity can be misleading: a small footprint might also represent a new or minor third-party provider. Thus, we treat DNS servers as self-managed if they share the same second-level domain (SLD) as the queried domain’s NS record. Similarly, an MX record sharing the same SLD indicates a self-managed mail server. For policy hosting, we designate any host that serves at most five⁴ domains as self-managed.

Limitation: Despite our careful methodologies, there is a possibility that our approach misclassifies domains as self-managed or third-party hosted. For example, if an administrator independently manages all email and web servers for ≥ 50 domains, our methodology might incorrectly classify

³We tallied effective SLDs for each MX and NS entry, then considered providers with 50 or more unique domains as third-party.

⁴Based on our dataset and operator insights, a single administrator commonly manages up to five domains in personal or small-scale setups.

these domains as third-party hosted due to their prevalence in our dataset. Similarly, even if a domain owner uses a third-party DNS provider, they may still configure the DNS records themselves, retaining full control over their DNS settings.

Invalid MTA-STS records

The initial step for a domain name owner is to publish an MTA-STS record in their zone. The primary task for domain owners is to ensure the syntactical accuracy of the record and also make sure only one such record exists; failing to do so may result in MTA-STS being considered as not deployed [85].

During our observation period, we note that the vast majority of domains successfully publish a correct record, irrespective of who manages the zone. For example, in our most recent snapshot, 67,699 domains (99.5%, out of 68,030 domains) accurately publish MTA-STS records. Among 331 domains with errors, 65 (19.6%) domains have no `id` field, 203 (61%) have an invalid `id` which is not permitted by RFC8461[85] such as including "-"; the standard allows only alphanumeric characters [85]. 52 (15.7%) domains start with an invalid version prefix and 2 domains contain invalid extension fields (e.g., "`v=STSV1; id=1; mx: a.com; mode: testing;`"). In summary, we find MTA-STS policy records in the DNS to be generally well-managed.

MTA-STS Policy Retrieval Errors

Next, we focus on errors related to retrieving the MTA-STS policy via HTTPS. Previously, recall [Figure 6.3](#), we found that the majority of misconfigurations in MTA-STS enabled domains occurs here. Although it may seem straightforward for a client to fetch a policy file from the web server, there are various operational aspects to consider:

1. DNS: The domain name owner must have an A or CNAME record for the `mta-sts` subdomain pointing to the policy server.
2. TCP: The host must have a webserver running and configured to listen on port TCP 443.
3. TLS: The web server must present a PKIX-valid certificate; otherwise, an invalid certificate (e.g., expired) will cause the TLS handshake to fail, blocking policy file retrieval.
4. HTTP: The web server must return an MTA-STS policy with an HTTP status code of 200.
5. Policy Syntax: A successful response does not guarantee a *correct policy file*; the file must also be semantically valid per the standard [85].

We now analyze each error in relation to the managing entity responsible for the policy server. In our latest snapshot, among 68,030 domains with MTA-STS records, 53,935 (79.3%) were classified into those using third-party (28,591) and self-managed (25,344) policy servers. We also examine these errors from the domain owner's perspective, as setting up a policy server involves actions like adding CNAME records to enable the policy server to obtain a domain-validated TLS certificate.

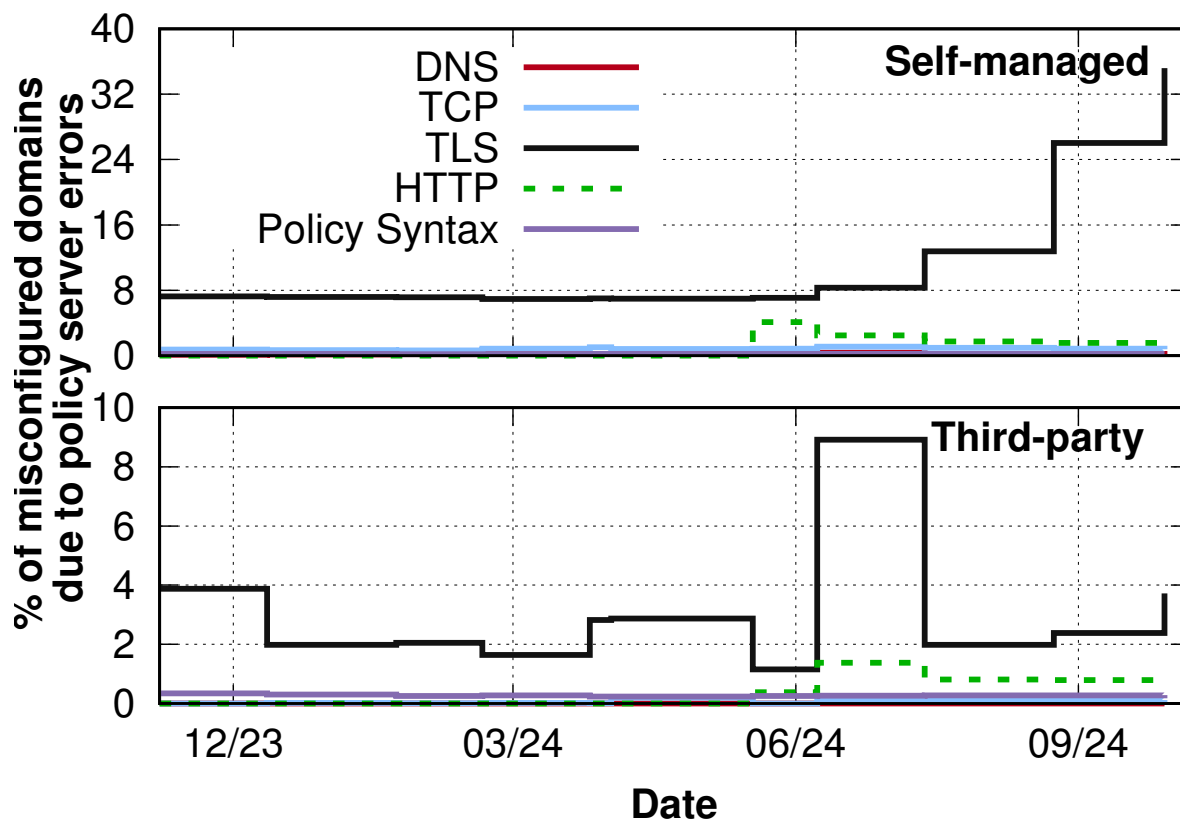


Figure 6.4: Percentage of MTA-STS enabled domains with misconfigured policy servers by error type and managing entity. In our latest snapshot, 9,588 (37.8%) self-managed and 1,393 (4.9%) third-party policy servers had misconfigurations. A spike on June 8, 2024, for third-party servers was due to a leading provider issuing self-signed certificates for 1,385 domains. Late spikes for self-managed domains were caused by Porkbun domains, as detailed in [Figure 6.3](#).

In general, third-party managed policy servers are more likely to correctly deliver an MTA-STS policy compared to self-managed servers. In our latest snapshot, 1,393 (4.9%) third-party managed servers were misconfigured, compared to 9,588 (37.8%) self-managed servers, as shown in [Figure 6.4](#). Below, we detail the individual failure cases encountered:

DNS errors: DNS errors (i.e., policy domain unresolvable) are rare in self-managed servers and non-existent in third-party policy servers. In our most recent snapshot, we could not resolve A or AAAA record for 42 domains that self-manage their policy file.

TCP errors: The majority of TCP errors result from closed ports or connection timeouts, indicating that these domains are not properly running their web servers. In our latest snapshot, 193 (0.7%) self-managed policy servers and only 34 (0.1%) third-party managed servers faced this

issue.

TLS errors: Interestingly, TLS is the primary cause of policy server errors across all categories. In our latest snapshot, 8,871 self-managed and 1,113 third-party policy servers failed to complete the TLS handshake due to errors. For self-managed domains, the majority of TLS errors (8,385, 94.5%) stem from `Common Name` or `Subject Alternative Name` mismatches, indicating that the presented certificate does not include the correct `mta-sts.` subdomain.

However, for domains managed by third-party providers, 463 (43.6%) fail due to missing certificates installed for the domain. This is unexpected, as prior research on DANE MX hosts [81] found third-party managed systems to be *less* error-prone. However, in our findings, one third-party provider, DMARCReport, accounts for 354 (76.5%) of the domains with SSL alert errors.

We found that all these domains have CNAME records delegating their policy servers to DMARCReport. However, upon contacting their support, we learned that these domains have *never* been hosted on their service. This indicates the issue stems not from third-party mismanagement but from a misunderstanding of MTA-STS policy delegation by email administrators. Some administrators may have mistakenly pointed their CNAME records to third-party policy servers, believing they could enforce MTA-STS without proper authorization or aligning the `mx` patterns with the actual MX records. This misconfiguration highlights the complexities of MTA-STS implementation, particularly in policy delegation. We explore this issue further in [section 6.3](#).

HTTP errors: HTTP errors are relatively rare, occurring in 1,336 domains. Among these, 377 (1.5%) are self-managed domains, and 215 (0.8%) are domains outsourcing policy management. As expected, the majority (387 (65.3%)) of these HTTP errors are due to 404 error codes.

Policy Syntax errors: Syntax errors in supplied policies are rare. Only 55 self-managed domains and 81 domains using third-party policy servers provide syntactically incorrect policies. Of these, 87 (64%) errors result from domain owners misunderstanding the standard, leading to invalid `mx` patterns such as using email addresses, trailing dots, or even empty patterns.

Interestingly, we found 5 domains using third-party hosting providers serving completely empty policy files despite presenting valid TLS certificates. All 5 domains are managed by DMARCReport, which handles policies for approximately 7,000 domains but returns empty policy files only for these 5. We will explore this issue further in [section 6.3](#).

PKIX-invalid MX hosts

In this section, we shift our focus to the MXes, which are *expected* to provide PKIX-valid certificates. Although senders may accept invalid certificates when the `mode` in the policy file is set to either `"testing"` or `"none"`, we validate the certificates regardless of the `mode` to assess their

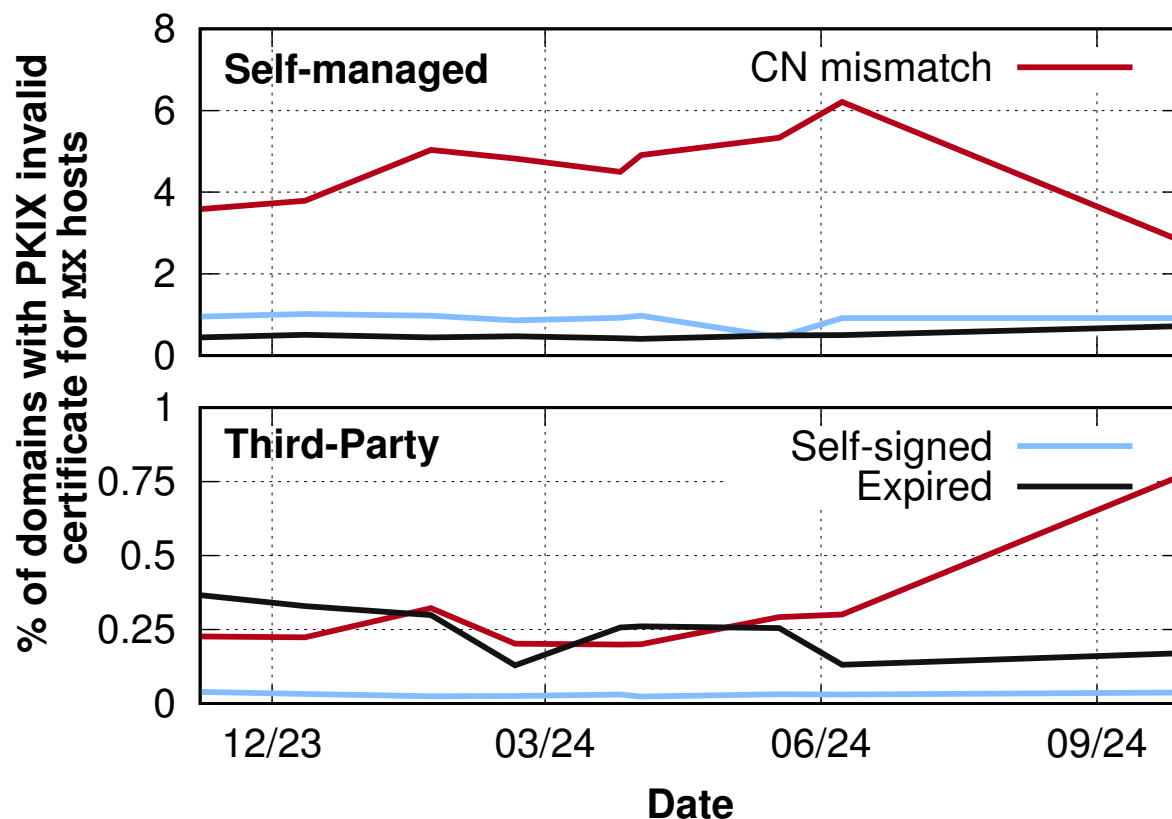


Figure 6.5: Percentage of MTA-STS enabled domains with PKIX invalid certificate for MX hosts; in our latest snapshot, 1,046 (4.4%) domains that self-manage their MX hosts and 397 (1%) domains that use email hosting services present PKIX-invalid certificates. 270 domains with self-hosted MX host fixed their Common Name mismatch error in our latest snapshot.

readiness for MTA-STS and identify any misconfigurations or weaknesses in the certificate setup. When the mode is set to "enforce", senders must not send emails if the host provides PKIX-invalid certificates [85].

Among the 68,030 domains with MTA-STS records in our most recent snapshot, we are able to classify 64,195 (94.4%) domains into 40,683 (59.8%) domains using third-party MXes, with the remaining 23,512 (34.6%) domains self-managing the MXes.

Figure 6.5 presents our finding. First, we note that third-party providers generally manage their MX hosts well as expected; only 397 (1%) domains using third-party email provider have at least one MX presenting an invalid PKIX certificate⁵, compared to 1,046 (4.4%) for domains that self-manage their MXes.

Since a domain can have multiple MX hosts and may have partially invalid settings, we further

⁵Upon further investigation, we find that this issue is attributed to one large provider (mxrouting.net) responsible for 122 (39%, out of 313) domains

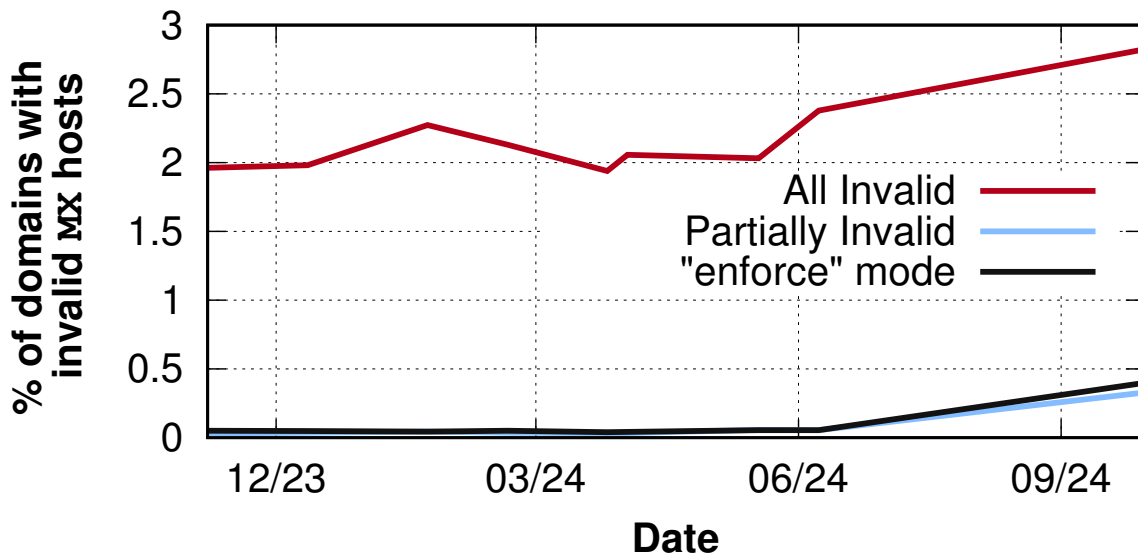


Figure 6.6: Percentage of MTA-STS enabled domains with all invalid, and partially invalid MX hosts. "enforce" mode presents domains with stricter policy and at least 1 invalid MX host; these are subject to potential email delivery failure from MTA-STS compliant senders and in our latest snapshot, we found 269 such domains.

divided the invalid domains into two categories as shown in Figure 6.6: 1) all invalid, 2) partially invalid. In our latest snapshot, 1,326 (1.9%) domains (149 (0.3%) third-party and 993 (4.2%) self-managed) are unable to present a valid TLS certificate on *all* MXes.

6.2.4 Misconfigurations: Inconsistency Errors

In the previous sections, we focused on individual errors that can occur in DNS records, policy servers, and MX hosts. However, even if each component of MTA-STS *appears* to be correctly configured, there is still a possibility that the sender may not be able to validate the recipient domain's policy when the MX records do not match the `mx` patterns listed in the policy file. *Especially if the mode in the policy file is "enforce" while the connected MX host does not match any of the patterns in the policy file, MTA-STS compliant senders must not deliver the message to that host.* Hence, we now focus on domains with an inconsistency between their MX record and the `mx` pattern in their policy.

When inconsistency occurs, we group the possible causes as follows:

- TLD mismatch: The top-level domain of the MX record differs from what is in the policy file.
- Complete domain mismatch: The domain name in the MX record is entirely different from the `mx` patterns, lacking any meaningful overlap.

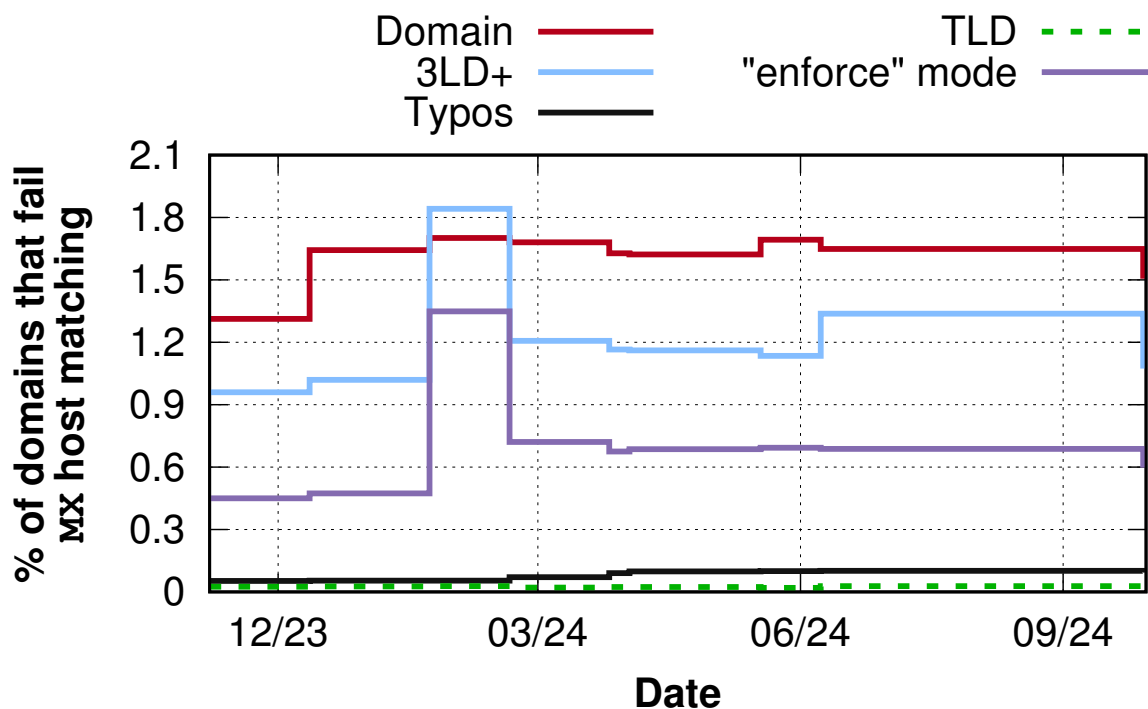


Figure 6.7: Percentage of MTA-STS enabled domains with mismatches between mx patterns listed in the policy file and actual MX records due to mismatches in TLD, SLD, or extra subdomains, or typos. "enforce" mode presents mismatched domains with stricter policy; these are subject to email delivery failure from MTA-STS compliant senders and in our latest snapshot, we found 406 such domains.

- Partial domain mismatch (3LD+): The SLD portion aligns, but further labels diverge.
- Typographical errors: Minor typos in the mx pattern (e.g., edit distance ≤ 3 [84]) prevent a match (note that TLD mismatches do not qualify as typos).

Figure 6.7 shows the results. Interestingly, we find a small but notable set of domains (63 in our latest snapshot) that have inconsistent MX patterns with an edit distance of ≤ 3 to at least one MX of the domain. Such typos are often caused by manual process [39], e.g., when the policy hosting provider asks the domain owner to manually enter their MX records without fetching them through DNS,⁶

Furthermore, our latest snapshot reveals 730 domains whose mx patterns match the effective SLD of the corresponding MX host but include extra labels, creating mismatches from the third label onward ("3LD+" in Figure 6.7). Upon closer inspection, we discover that among these 730 domains, 597 (81.8%) have the `mta-sts` subdomain in the policy file. We believe this is due to a

⁶A similar problem has been observed in the context of DNSSEC, where a previous study [34] pointed out that the chain of trust can be easily broken when a DNS registrar asks the domain owner to generate and upload DS records, rather than fetching the DNSKEYs and generating the records themselves. This manual process introduces the risk of misconfigurations and inconsistencies.

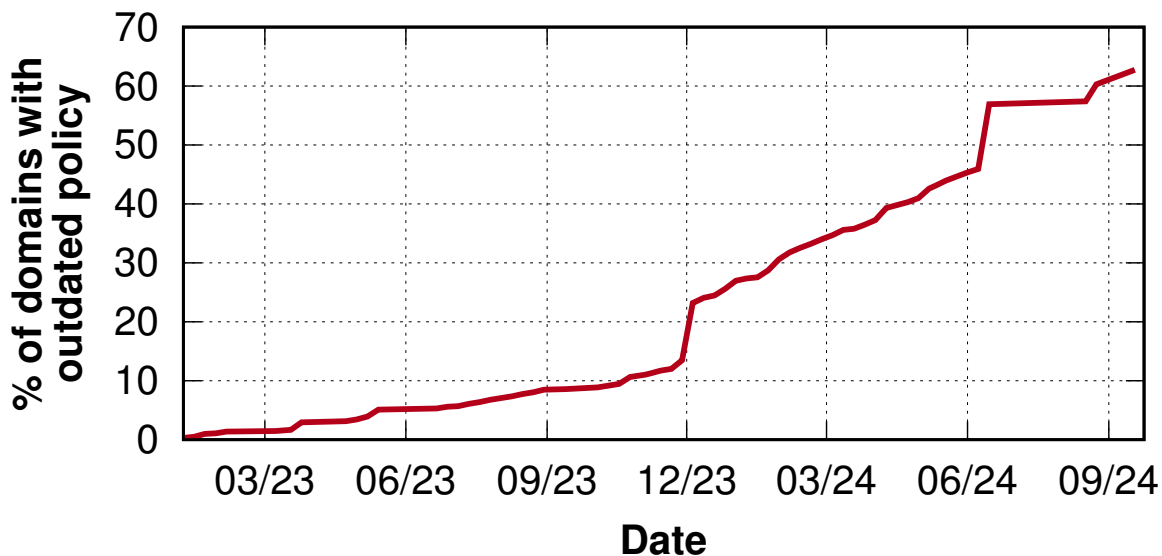


Figure 6.8: The percentage of domains with mismatched mx patterns in their MTA-STS policy file that can be correctly matched with historical MX records.

misunderstanding of RFC8461 [85].

Interestingly, this issue is not limited to users who manage their own MX host or policy server. We observe a spike in 3LD+ mismatches on January 23rd, 2024. This was caused by one provider, `lucidgrow.com`, which assigns a unique MX host to each domain but outsources policy server management to a third-party service, DMARCReport. On January 23rd, for all 246 of these domains, none of their MX record matched with mx pattern in their policy file. To make matters worse, they also had the mode in the policy file set to "enforce", which means they might have encountered email delivery failures from MTA-STS compliant senders during this period. Even though the issue was quickly resolved, it demonstrates that mismatches can occur even when domain name owners outsource their MX host and policy server management to *different entities*.

Finally, 1,023 domains have mx patterns that are completely different from their actual MXes, suggesting that the mismatch might not be related to the MX records captured *in the same snapshot*. We investigate whether these currently mismatched MX hosts can be correctly matched with any of the domain's historical MX records. This situation may arise when administrators forget to update MX records or mx patterns in the policy file after migrating mail servers.

Thus, to test our hypothesis, we first consider the MTA-STS enabled domains that have domain name mismatches in our latest snapshot. Then, for each historical snapshot, we check if we can find MX records that match with the mx host in the policy, which is shown in [Figure 6.8](#); interestingly, we observe an increasing trend: a majority (644, 63%) of these unmatched mx patterns stem from obsolete MX records where domain owners did not update their policy file following a change in MXes. This highlights another challenge in correctly managing MTA-STS, as it requires consistent synchronization between the policy hosts and MX records, which suggests that *this problem can be*

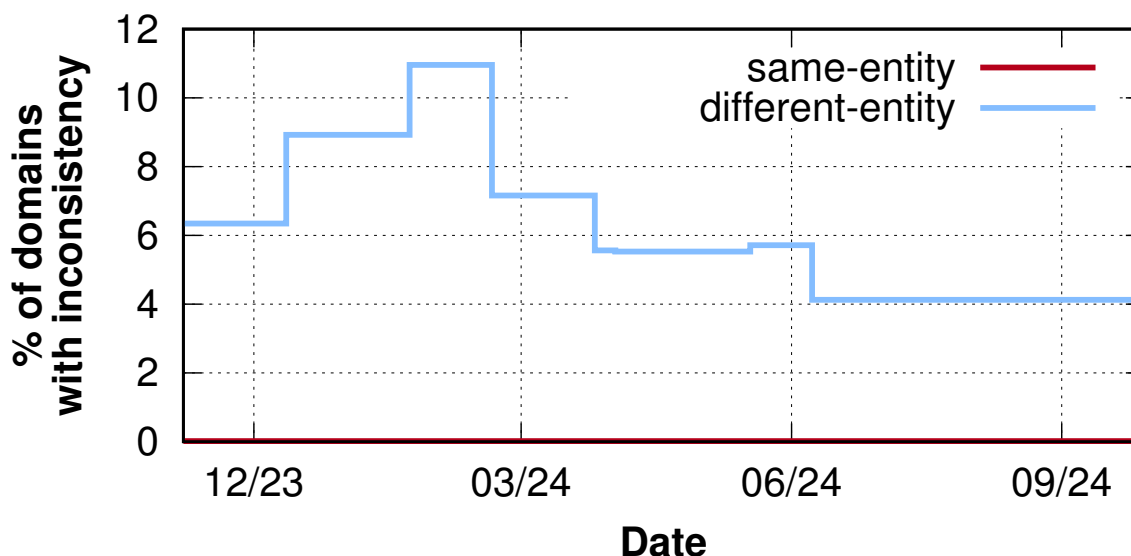


Figure 6.9: Inconsistency in domains outsourcing both policy servers and email hosting, based on whether the same provider manages both.

exacerbated if domain name owners use different third party entities for their policy host and MX servers.

6.2.5 Inconsistency From Multiple Third Parties

We have observed that mx mismatches are amplified when policy hosting and MX operations are outsourced to different third parties. In MTA-STTS, ensuring consistency is the domain owner's responsibility, which may be overlooked when multiple third-party services are utilized. Therefore, we now examine how inconsistencies manifest when both email and policy hosting are outsourced to different providers.

Methodology

For domains relying on third-party services for both MX hosts and policy servers, we determine whether the same provider manages both. This is straightforward, as policy delegation typically uses CNAME records; by comparing the SLDs or second labels of the policy host CNAME and MX records, we can infer if they share the same management. For example, Tutanota customers have MX records set to `mail.tutanota.de` and CNAME records for policy delegation set to `mta-sts.tutanota.com`, where the shared label `tutanota` indicates the same provider.

Providers	CNAME Patterns in TXT or A Records	# of Domains	Email Hosting Support	Behavior After Opt-out		
				Returning NXDomain	Reissuing Cert	Policy File Update
Tutanota	.mta-sts.tutanota.de	7,614	✓	✗	✗	✗
DMARCReport	a-com.mta-sts.dmarcinput.com	7,293	✗	✗	✓	↳ Empty File
PowerDMARC	a-com._mta.mta-sts.tech	3,753	✗	✓	✗	mode ↳ none
EasyDMARC	a-com._mta-sts.easydmarc.pro	2,222	✗	✗	✓	✗
Mailhardener	a.com._mta-sts.mailhardener.com	1,558	✗	✓	✗	mode ↳ none
URIports	a-com._mta-sts.uriports.com	1,100	✗	✓	✗	✗
Sendmarc	a.com._mta-sts.sdmarc.net	805	✗	✗	✓	✗
OnDMARC	.mta-sts.a.com._mta-sts.smart.ondmarc.com	451	✗	✗	✓	✗

Table 6.2: The list includes the top eight policy hosting providers, along with their hosting policies for customers who have opted out. The number of domains is based on our most recent data snapshot from September 29, 2024.

Results

Initially, we find 26,414 domains that use third-party services for both their MX host and policy server. Of those, we can further classify 18,922 domains where their hosting services are managed by different providers and 7,492 domains where the same provider manages both services in our latest snapshot. We now examine how the consistency differs depending on whether the two entities are managed by the same provider or not. As [Figure 6.9](#) shows, we observe that inconsistency is almost non-existent when both entities are managed by the same provider; only 1 domain `laura-norman.com` has inconsistency error due to a typo throughout all the snapshots. On the contrary, 640 (3.4%) domains have this issue when management of these components is split over to different third parties.⁷

Collectively, these observations again underscore the importance and challenge of coordination and communication between providers even when a domain outsources its MTA-STS management.

6.2.6 Key Takeaways

Our analysis of MTA-STS management highlights the following key insights:

1. Policy server misconfigurations are the most common individual error in MTA-STS deployment. Even with third-party services, many domains face issues. Across all snapshots, 70-85% of errors are related to misconfigured policy servers.
2. Self-managed email servers struggle more with maintaining PKIX-valid certificates. In our latest snapshot, 1,046 (4.4%) self-managed domains had broken PKIX configurations, compared to just 397 (1%) domains using email hosting providers.
3. Inconsistencies between policy mx patterns and MX records persist, particularly when policy and email management are outsourced to different entities. In our latest snapshot, only 1 domain

⁷The spike in observed in January is also due to an email provider, `lucidgrow.com`

had this issue with the same provider for both, compared to 640 (3.4%) domains using different providers.

These findings underscore the challenges of MTA-STS management and emphasize the need for improved coordination and communication between domain owners and third-party providers.

6.2.7 Responsible Disclosure

For misconfigured domains based on our latest snapshot, we sent notification emails to the `postmaster` address of 20,144 misconfigured domains from the latest snapshot between 22 Oct, 2024 and Nov 6, 2024. Unfortunately, more than 5,000 emails bounced for various reasons and as expected in prior work [109]. Given the comparably low impact and high visibility—likely accelerating mitigation—of the observed issues, and based on feedback from operators who *did* receive these mails, we decided to not attempt alternate notification channels. After the end of the notification campaign, we observed that 2,064 (10%) misconfigured domains had their issue(s) resolved, even though this may have been independent of our notifications.

6.3 MTA-STS Policy Delegation

In the previous section, we explored how the contents of delegated MTA-STS policies can lead to validation errors. However, errors may also arise from the delegation of MTA-STS policy hosting itself. To better understand this issue, we now examine popular policy hosting providers and how they implement policy delegation for domain owners.

Domain owners who delegate policy hosting must set up CNAME records pointing to their policy hosting provider. By analyzing these CNAME records for the MTA-STS policy label, we identify the providers responsible for hosting the policies. Table 6.2 lists the eight most popular policy hosting providers from our latest snapshot, along with their CNAME patterns and the number of domains using their services.

Despite these providers serving a significant customer base, we were surprised to find that some domains relying on them still experience issues (see section 6.2.3); for example, some policy hosts returned expired certificates or empty policy files, causing errors. Interestingly, these issues affect only subsets of each provider’s customers; for instance, only five domains using DMARCReport had empty policies.

Our qualitative analysis suggests that these cases often involve customers who discontinued the policy hosting service but left MTA-STS configurations pointing to the provider. To understand this phenomenon further, we contacted all providers via their support systems to inquire about their procedures for handling incomplete customer opt-outs. We observed four different approaches:

1. **NXDOMAIN Response:** Three policy hosting services (MailHardener, URIPorts, and PowerDMARC) return NXDOMAIN responses to the canonical name a CNAME points to, which makes senders unable to resolve the policy domain.
2. **Continued Certificate Issuance:** Four providers (DMARCReport, EasyDmarc, Sendmarc, and OnDMARC) continue issuing certificates for the mta-sts subdomain via ACME domain validation, even for inactive customers. However, EasyDmarc, OnDMARC, and Sendmarc do not update the policy file when a customer's MX records change. As a result, if the mode is set to "enforce" and the mx patterns are not updated to match new MX records, the domain may fail to receive emails from MTA-STS-compliant senders
3. **Empty Policy File:** One operator, DMARCReport, changes the policy to an empty file. This will cause a parsing failure according to the MTA-STS standard, making senders treat this error equivalent to a "none" mode.
4. **Mail Service Discontinued:** Tutanota rejects incoming emails if unsubscribed customers continue pointing their MX record to Tutanota, but the policy file remains unchanged. This can cause delivery issues if customers update their MX records without adjusting their MTA-STS settings. We received no response on whether Tutanota renews certificates for inactive customers. However, in our latest snapshot of 7,614 domains served by Tutanota, 10 domains with policy server errors still point their MX record and mta-sts IP to Tutanota, with 8 having expired SSL certificates.

In summary, none of the providers follow the best practices outlined in [subsection 2.3.7](#). Three providers return NXDOMAIN responses, effectively shutting down the policy file resolution, while four others serve stale or invalid policies, risking email delivery failures for inactive customers.

These findings underscore the need for policy hosting providers to adopt a standardized and graceful deprovisioning process when customers opt out or become inactive.

6.4 Sender-Side MTA-STS Validation

While our main analysis has concentrated on how domains configure MTA-STS for incoming email, it is equally important to understand whether sending MTAs actually validate these policies when delivering outbound mail. In other words, even if a domain publishes a valid policy, its security benefit hinges on sender-side adoption and enforcement. For instance, if major providers neglect MTA-STS validation, recipients gain little protection from the protocol's deployment. Consequently, we complement our recipient-focused perspective by briefly exploring how senders implement MTA-STS, the potential obstacles they encounter, and how these practices affect the overall ecosystem.

6.4.1 Sender Side Dataset

To assess whether senders implement MTA-STS, we utilized an aggregate dataset shared with us by email-security-scans.org, which examines MTA sending behavior by recruiting participants to send emails to their platform [61]; this includes domains that implement MTA-STS with varying configurations and policies. The dataset spans over 3,806 individual deliverability tests spanning 2,394 unique sender domains. Data was collected between February 2023 and November 2024. For our analysis, we consider the most recent test per sender domain.

Limitations: It is worth noting that major mail operators heavily influence the data provided by email-security-scans.org; of 11,564 recorded MX interactions, 3,043 (26.31%) ehlo responses are attribute to outlook.com, and a further 2,663 (23.03%) are from google.com. In total, the top 10 operators account for 7,019 (60.7%) of all recorded interactions.

6.4.2 MTA-STS Validation

Our findings reveal that 2,264 (94.6%) domains support TLS when delivering email. The vast majority, 2,232 (93.2%) domains, perform opportunistic TLS (i.e., accepting any TLS certificate). Only 31 (1.3%) domains always require PKIX-valid certificates, regardless of MTA-STS or DANE being in use.

Regarding MTA-STS validation, we find that a total of 469 (19.6%) domains perform MTA-STS validation when sending to an MTA-STS-enabled domain; this is encouraging compared to the domains configured with MTA-STS records (i.e., approximately 0.1%) measured in [section 6.1](#). Interestingly, 714 (29.8%) sender domains perform DANE validation; of these, 203 (8.5%) domains validate both MTA-STS and DANE. However, 62 (2.6%) of them prefer MTA-STS over DANE⁸, which is not recommended by RFC 8461 [85].⁹

⁸This is done by presenting a PKIX-valid certificate while our TLSA records for DANE did not match the certificate.

⁹This is a known bug in a common Postfix MTA-STS milter [91].

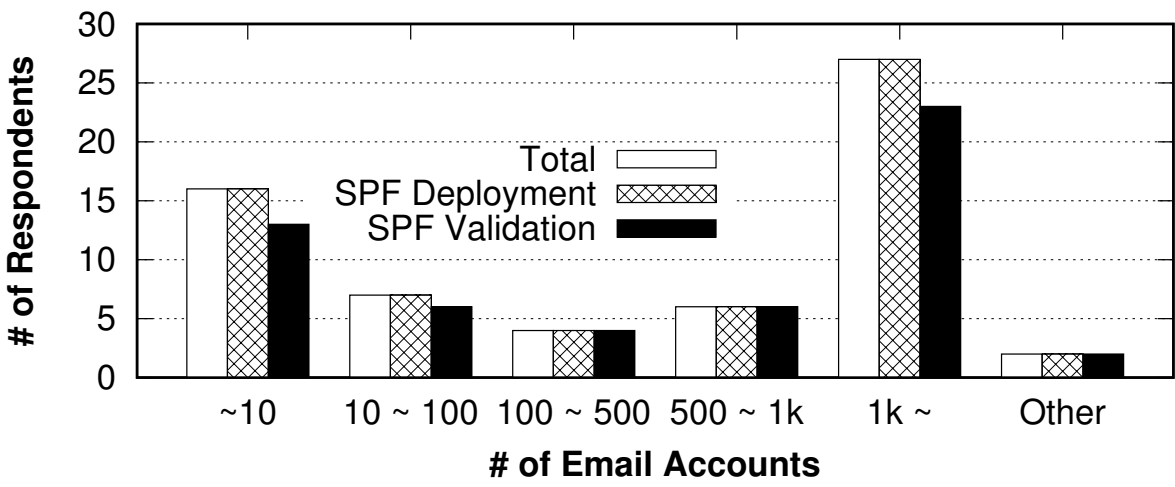


Figure 7.1: The figure shows the distribution of the number of email accounts managed by each of the 62 respondents who answered both questions regarding SPF deployment and SPF validation support; note that all respondents serving as SMTP administrators confirmed the deployment of SPF records.

Chapter 7

Email Security in Practice

In this chapter, we augment our empirical data with a couple of surveys regarding email sender authentication and email transport security. The first one was conducted in late 2022 to gain a more comprehensive view of SPF and DMARC ecosystem. The latter one was carried out in April 2024 to contextualize our measurements related to MTA-STS.

While our passive measurements and in-lab tests provide valuable quantitative insights, they are based on publicly accessible information and open-source software. The survey fills this knowledge gap, offering a qualitative look at operational practices and challenges in real-world deployments.

7.1 Operational Practice for SPF and DMARC

7.1.1 Survey Methodology

We sourced participants for our survey from specialized mailing lists, including the Mail Operators' List (MailOP) [5], the North American Network Operators' Group (NANOG) [96], and Email Security Standards in the European Union (MESSEU) [94]. Of the initial 95 respondents, 74 engaged with at least one survey question; The participant demographics, summarized in [Figure 7.1](#), reveal a wide range of mail-setup sizes: from 16 operators overseeing fewer than 10 accounts, to 25 operators responsible for more than 1,000 accounts. This variety suggests that our survey offers a broad view of the mail operator landscape. In summary, our diverse participant sample enhances our understanding of email infrastructure; also, the input from mail operators is particularly valuable as it plays a critical role in the standardization of protocols within the email ecosystem.

However, we also note that our survey has limitations worth noting for proper interpretation of the results. The small sample size offers a qualitative view rather than a basis for broader generalization. While we targeted knowledgeable operators, we did not verify their responses, which may introduce self-reporting and social desirability biases. Participation was voluntary, potentially leading to self-selection bias. Despite these limitations, our survey provides valuable qualitative insights into email operations, complementing our technical findings. We provide the full questionnaire in [Appendix D](#).

7.1.2 Survey Findings

SPF Deployment and Validation: We first understand how they *deploy* email security protocols by publishing SPF records; interestingly, we observe that all of the participants (74) answered that they publish SPF for source validation; among them, 45 (60.8%) and 39 (52.7%) deployed DKIM and DMARC for the same purpose as well. In contrast, the adoption rates for other protocols aimed at encryption and receiver validation are somewhat lower; for example, MTA-STS is used by 13 participants (17.6%) and DANE by 19 (25.7%).

Regarding SPF validation, 62 participants responded to whether their SMTP server validate sender's SPF record or not and 54 participants (87.1%) answered yes, while only 8 explicitly indicated they do not.¹; these findings underscore the critical importance of SPF in the realm of sender authentication. Firstly, among the 32 participants aware of when SPF validation occurs, we observe a multi-stage process: 3 perform validation after the HELO/EHLO command, 7 after the MAIL FROM command, 10 after the RCPT TO command, and 12 (37.5%) proceed after the DATA command. This variability underscores the feasibility of the attack vectors we propose, making it challenging for receivers to detect their unwitting role as reflectors when they check before the DATA command. Secondly, 14 participants (26%) report utilizing *multiple* extensions for SPF validation, potentially

¹The remaining 12 respondents did not answer the question.

increasing the volume of DNS requests for each validation process. This aligns with our experiments, showing that 13.2% of SMTP servers issue more than 51 queries during SPF validation.

RFC Recommendation vs. Practice: Among the 14 participants who were aware of their SPF lookup limits, three have opted to increase these limits, and one has additionally raised the void lookup limit, suggesting that the current RFC guidelines may not align with practical operational needs; interestingly, *all four operators serve customer bases exceeding 1,000 accounts*, suggesting that the existing RFC standards for lookup limits may not adequately serve large-scale email operators. We also note that four of these participants have extended their total SPF resolution timeout.

Software Update and Others: Of the 61 participants who responded to questions about software update frequency, 13 disclose that they *do not* update their SPF validation software periodically, with one explicitly stating they never do so. This raises concerns about the potential vulnerabilities in their SPF validation process. Lastly, we find it noteworthy that 9 participants report their SMTP servers support fewer than 10 concurrent connections; this limitation is especially concerning as it suggests attackers could potentially disrupt email traffic by exploiting delays in SPF validation.

SPF Validator Usage: Out of 54 participants who responded to questions about which SPF validator they use, 19 (35%) reported to use SpamAssassin [21], 14 (26%) use policyd-spf [139], 6 (11%) use libspf2 [136], 14 (26%) use Rspamd [7]. 14 (26%) participants use MTAs that natively support SPF validation while 4 (7.4%) of them did not know which validator they used. Out of the remaining 4 participants, one reported to use mtpolicyd [138], one iRedAPD [135], one amavisd-new [133], and one participant use Google's in-house solution. Note that 18 (33%) participants reported to have multiple SPF validators.

DMARC reports: To understand the general demographics of respondents, we first understand how they deploy email security protocols; out of 63 operators, we find that many of them deployed security protocols; SPF (100%), DKIM (60%), DMARC (52%), STARTTLS (74%), MTA-STS (18%), and DANE (25%), which indicates that the respondents are aware of security challenges in SMTP.

Out of 39 operators who published DMARC records, we also find that 26 (66%) also send DMARC reports to other mail senders. A closer look at [Figure 7.1](#) suggests that both large and small operators support DMARC reporting; for example, 10 (40%) operators that manage more than 1,000 accounts and 7 (43%) operators that manage less than 10 accounts both send DMARC report. Among the 26 operators that support DMARC reporting, only 5 (19%) of them also support TLS-RPT, which is in line with our findings that only two popular email hosting providers support it.

Security consideration: Out of 26 operators that support DMARC reporting, 18 (70%) answered the detailed questions regarding their configuration. We find that 6 (out of 18) uses OpenDMARC software for their DMARC reporting, which is vulnerable since it violates RFC recommendations in many ways as listed in [Table 5.3](#); interestingly 50% of them have less than 10 accounts in their infrastructure, which suggests that smaller operators are more inclined to use OpenDMARC.

Interestingly, 2 (out of 18) operators that manage more than 1,000 accounts reported that they do not implement EDV mechanism. 9 (out of 18) answered they do not know; however, 4 of these operators reported that they use OpenDMARC, thus we find that 6 (33%) operators in total do not implement EDV mechanisms for external domains, which include both small (≤ 10 accounts) and big ($\geq 1,000$ accounts) SMTP servers.

When focusing on the maximum number of rua tags, out of 9 (50%) respondents who answered the question, four of them reported a defined number with 1 (two operators), 3 (one operator), and 5 (one operator). One respondent did not answer, but they use OpenDMARC, which is limited to 255 characters on the rua tag. The other 4 providers indicated that they do not impose any limit, which can be easily used as reflectors for SMTP connection flooding attacks regardless of their implementation of EDV.

18 respondents answered the questions regarding the report interval (ri); 7 of them answered they respect ri tag, which is not common across the popular email hosting providers we surveyed. 4 of them indicated that they ignore the tag and the rest replied that they are not aware of them.

In summary, we find that 26 (41%) email providers currently support DMARC reporting. However, we also confirm the potential targets for reflection attack for those who do not impose the limit the number of report recipients in the rua tag (4 (6%)), do not implement EDV (6 (9%)), and use vulnerable software, OpenDMARC (6 (9%)).

While we focused on authenticating the sender of an email in these chapters, they leave open a complementary question: even if the sender is authenticated, can the message be delivered securely over the network? Our next couple of chapters address this gap by analyzing MTA-STS, a transport-layer security mechanism for SMTP, and examining how its deployment interacts with the realities we observed in email sender authentication.

7.2 MTA-STS in Practice

To contextualize our network measurements and contrast our findings with operational practice, we conducted a survey in April 2024 on MTA-STS use among operators running email services. The survey responses provide a qualitative perspective on the real-world challenges and practices associated with MTA-STS deployment. The complete set of survey questions is available in [Appendix E](#). A discussion of survey ethics is included in the ethics section in [Appendix B](#).

7.2.1 Survey Methodology

Recruitment We recruited survey participants from mailing lists, including the Mail Operators’ List (MailOP) [5], the North American Network Operators’ Group (NANOG) [96], and a mailinglist for Email Security Standards in the European Union (MESSEU) [94]; these mailing lists are populated by email administrators who demonstrate a high level of familiarity with email security protocols and standards. Out of 120 initial respondents, 117 engaged with at least one question in the survey.

Demographics The participant demographics, as shown in [Figure 7.2](#), cover a broad spectrum of mail-setup sizes. These range from 22 operators managing fewer than 10 accounts to 36 operators overseeing more than 500 accounts. This diversity indicates that our survey provides a comprehensive overview of the mail operator landscape.

Questionnaire Design When designing the questionnaire, we avoided leading questions, and utilized a Likert scale for opinion questions. We generally followed survey methodology best practices as outlined by Redmiles et al. [104]. We provide the full questionnaire in [Appendix E](#).

Limitation Although 117 survey responses yielded valuable qualitative insights, they cannot be taken as broadly representative. We recruited participants through mailing lists, which likely attracted more security-conscious email administrators than average, as noted in [7.2.2](#). Because the survey was unsupervised and had no pre-filtering, self-selection bias may be present, and common issues like self-reporting or social desirability biases could further influence results. Nevertheless, our goal was to *augment our technical observations* with qualitative perspectives; despite these limitations, the surveys findings offer useful context for understanding real-world email operations.

7.2.2 Survey Findings

Deployment Out of the 94 participants who responded to the question about their familiarity with MTA-STS, 89 (94.7%) had heard of the standard. Furthermore, of 88 respondents replying to the question whether they deployed MTA-STS for their primary domain, 50 (56.8%) confirmed its deployment. These high familiarity and deployment rates are likely related to our sample recruited from email expert groups, see [subsection 7.2.1](#).

Motivation for Deployment The primary reason for deploying MTA-STS, identified by 34 out of 42 respondents (80.9%), was to prevent downgrade or interception attacks caused by START-TLS stripping or DNS poisoning. Interestingly, 9 participants expressed greater trust in the web

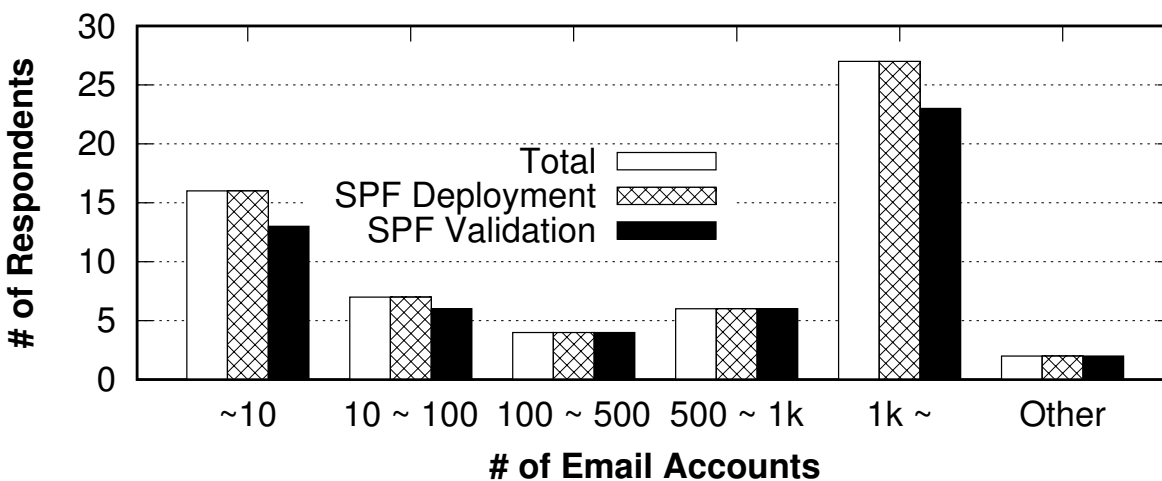


Figure 7.2: The distribution of the number of email accounts managed by each of the 92 respondents who answered the question regarding MTA-STS deployment.

PKI than in DANE. Similarly, 10 participants favored MTA-STS over DANE, citing DANE’s complexity in managing DNSSEC requirements; this complexity was also cited as a key motivation for the development of MTA-STS.

From a requirements standpoint, 13 out of 41 (31.7%) participants stated that customer demand drove their adoption of MTA-STS, while 14 (34.1%) participants indicated that regulatory compliance mandated its implementation. Furthermore, 5 participants responded that MTA-STS had a better reputation among large email providers, which contributed to their decision to adopt it.

Challenges in Deployment Among the 43 respondents who had deployed MTA-STS, 21 (48.8%) identified operational complexity as the primary bottleneck. Additionally, 17 (39.5%) believed that DANE is fundamentally more secure, while 5 (11.6%) indicated that they do not require email encryption.

As for the 33 respondents who have not deployed MTA-STS, 15 (45.4%) stated they use DANE instead. Another 9 (27.2%) found MTA-STS too complicated to manage.²

Management Of the 41 respondents who commented on the setup and management of MTA-STS, 8 (19.5%) found configuring the HTTPS policy file challenging, while 11 (26.8%) reported difficulties in managing policy updates. Moreover, regarding the policy update sequence, among the 42 respondents, 15 (35.7%) indicated that they never updated their policy. Notably, 10 (23.8%)

²One administrator wrote “We now have to manage a webserver cluster for redundancy [sic], a new website, and dns records, and sync them all, [...] , since mta-sts is a file on a webserver and not in DNS, the list of *valid* entries could be inaccurate.”

responded updating the TXT record first, a practice not recommended due to potential temporary disruptions in mail delivery, as outlined in the standard [85].

MTA-STS vs. DANE Out of 79 respondents to the related question, 78 (98.7%) answered that they are familiar with DANE. Among these, 26 (33.3%) stated that they do not serve any TLSA record. Additionally, 10 respondents responded that their DNS authoritative server and/or registrar lacked DNSSEC support, which shows the challenge of DANE deployment.

While both MTA-STS and DANE are designed to enhance email security, a majority of administrators (51, 72.8%) argued that DANE is the superior solution. One administrator highlighted the potential drawbacks of MTA-STS, writing: *“I think MTA-STS is actively harmful in the sense that it undermines the incentive to deploy DNSSEC and DANE because it offers a false alternative. Note that the TOFU principle of MTA-STS is objectively less secure than DANE which does not rely on TOFU.”*

Summary

While MTA-STS has limited general prominence, it is well known within sector-specific groups. Both MTA-STS and DANE are familiar to operators in our sample, with concerns focused on their respective complexities. For DANE, the primary challenge is its reliance on DNSSEC, despite its superior security. Although MTA-STS was envisioned as an alternative in environments where DNSSEC is impractical, operators cited other reasons for its adoption, including confidence in the established web PKI ecosystem and support from major providers such as Google. At the same time, they identified policy server setup and ongoing maintenance as persistent challenges, echoing issues we highlighted in [section 6.2](#).

Both sender authentication and transport security ultimately rely on the integrity of DNS records to convey policies and keys. Thus, after examining misconfigurations in these email-layer defenses, our next focus turns to DNSSEC, the security mechanism intended to protect DNS itself. This shift from higher-layer protocols to their foundational substrate allows us to trace how DNS fragility propagates upward and undermines ostensibly independent defenses.

Chapter 8

Understanding DNSSEC Debugging Pattern

DNS Security Extensions (DNSSEC) [11, 12, 13] were introduced in 1999 to prevent DNS spoofing and on-path tampering by adding cryptographic authenticity checks to DNS data. Each zone signs its records (RRSIG) using a private key and publishes the corresponding public key (DNSKEY), while the parent zone holds a DS record that attests to the child’s key. A resolver thus constructs a “chain of trust” from the queried domain up to the root, rejecting records with missing or invalid signatures.

Despite these security benefits, DNSSEC adoption remains strikingly low. Only around 5% of .com and .net domains [2] are signed, with a similarly small fraction (7%) overall [49]. Moreover, misconfiguration rates are alarmingly high: more than 30% of “signed” domains are effectively unresolvable [62]. Administrators must contend with intricate tasks such as key generation and rotation, RRSIG updates, and DS synchronization at the parent. A single oversight can cascade into multiple validation failures.

Real-world outages underscore how serious these misconfigurations can be. At the TLD level, .nz and .fj both experienced DNSSEC-related interruptions that broke resolution for large swaths of users [51, 53]. Even major platforms like Slack [119] suffered downtime when their DNSSEC setup lapsed. These episodes highlight the fragility of DNSSEC and the difficulty of maintaining secure, signed zones at scale.

Administrators seeking help often turn to diagnostic tools such as DNSViz [54]. While these tools accurately flag specific errors (e.g., missing DNSKEY, expired RRSIG), they do not isolate the root cause or supply a direct path to remediation. In practice, an operator may see dozens of messages in DNSViz without knowing which one or two errors trigger them all. Many resort to turning off DNSSEC rather than face repeated breakages and troubleshooting [23, 63].

In this work, we harness DNSViz’s historical logs to examine the full lifecycle of DNSSEC errors. We investigate which issues appear most often, how long they persist, and whether administrators eventually fix or abandon them. Our longitudinal analysis shows that certain errors such as nonzero NSEC3 iteration count or missing signatures and non-existence proofs appear more than others. These findings emphasize the need for a solution that goes beyond mere diagnosis and actually guides operators toward precise, actionable fixes.

Building on these insights, we introduce DFixer, a domain-specific tool that unifies error detection, root-cause identification, and automatic remediation. We curate a mapping from each commonly observed DNSSEC misconfiguration (e.g., missing or revoked key, invalid signature) to the series

Category	Root	TLD	SLD+
Total Snapshots	6,234	356,136	747,455
Unique Domains	1	4,196	319,277
w/ at least Two Snapshots	1	2,349	84,962
CD domains	0	642 (27.3%)	21,734 (25.5%)
SD domains	1	1,707 (72.7%)	63,228 (74.4%)

Table 8.1: Overview of the DNSViz dataset. It contains 1.1M total snapshots spanning from 2020-03-11 to 2024-09-25. Second-level and lower (SLD+) domains are determined using the Public Suffix List (PSL). CD and SD domains are defined later in [section 8.2](#).

of BIND commands needed to restore a valid chain of trust. Unlike general-purpose large language models, which often neglect error dependencies and produce vague suggestions, our approach yields correct and reproducible results for real misconfigurations found in DNSViz logs.

Our main contributions in this work are [15]:

- Using DNSViz’s historical data, we provide a fine-grained analysis of which DNSSEC errors dominate, how they propagate, and how quickly administrators address them (RQ1).
- We design DFixer, a pipeline that correlates multiple secondary errors into a few root cause(s) and then generates exact BIND commands (e.g., `dnssec-signzone`, `dnssec-keygen`) to fix them. Our evaluation with more than 296K erroneous zones shows that DFixer can fully resolve DNSSEC-related errors in 99.99% of them (RQ2).
- We replicate real misconfigurations in a local sandbox using our tool ZReplicator to ensure that DFixer’s recommended fixes genuinely resolve the issues when rechecked with DNSViz. Out of 296K zones with diverse combinations of DNSSEC errors, we were able to exactly emulate 79% of them using ZReplicator (RQ3).

8.1 Dataset

We obtained the complete historical database of DNSViz from DNS-OARC [120]. Our analysis focuses on data collected between March 2020 and September 2024, as summarized in [Table 8.1](#).

Terminology and Dataset Scope. DNSViz operates under a query-driven model: when an administrator or user analyzes a Query Domain, DNSViz recursively traverses the domain’s DNSSEC chain of trust, from the root down to the zone containing Query Domain (called Query Zone). Along the way, it queries various record types (e.g., A, AAAA, MX) and verifies DS/DNSKEY pairs in each delegation. To test negative responses (e.g., NSEC or NSEC3), DNSViz also queries random sub-labels that do not exist.

Every responseDNS data, RRSIG records, delegation details, negative proofsundergoes validation against DNSSEC standards, and DNSViz assigns specific error (or warning) codes for detected

failures. In this dissertation, we *only* consider DNSSEC-related errors that (1) can cause a validator to return SERVFAIL or (2) violate a DNSSEC “MUST” property from relevant RFCs [11, 12, 13], excluding advisory (“SHOULD”-level) warnings and non-DNSSEC issues.

Limitation. All analyses in the DNSViz dataset come from *user-initiated scans*, which may introduce self-selection bias, as the dataset might not reflect the broader DNS ecosystem. Even so, it captures a wide range of real-world DNSSEC configurations and offers a practical lens on how they behave *in practice*, especially for domains whose administrators *actively* sought diagnostic insight. While we understand that domains that never appear in DNSViz may behave differently, the collected data should be sufficiently representative to support the analysis presented in next sections.

DNSViz Prevalence. To assess the broader significance of this dataset, we compare the domains appearing in DNSViz logs to domain popularity rankings. Figure 8.1 highlights the percentage of Tranco top-1M domains [99] also present in our DNSViz dataset. Two observations stand out:

- *High Coverage of Popular Domains.* Among the top 10k domains, 20% appear in the DNSViz logs (red line, bottom). This indicates that well-known domains are frequently tested or monitored, either by their operators or by curious users, presumably because DNSSEC correctness is seen as especially critical for large or high-traffic sites; the top graph supports this, showing that DNSSEC misconfigurations are comparatively less common among top 20k domains.
- *Diverse Representation Among Signed Domains.* The blue line in Figure 8.1 (middle) shows that a substantial fraction of domains that have ever been DNSSEC-signed (at least once) appear across the entire popularity spectrum. In fact, more than 30% of these domains—even those consistently in the lower popularity bins—show up in the DNSViz logs. This finding indicates that smaller or niche domain owners also rely on DNSViz to troubleshoot or refine their DNSSEC configurations, suggesting that the tool’s usage extends beyond mere curiosity. Rather, it meets the genuine operational needs of a diverse set of domain operators, regardless of their popularity ranking.

8.2 Validation and Category Definitions

In order to systematically examine DNSViz’s diagnostic data, we first establish a scheme for categorizing individual snapshots (§8.2) and then group domains based on how their DNSSEC statuses evolve over time (§8.2). This approach allows us to distinguish truly *static* configurations (e.g., consistently valid DNSSEC setups) from those experiencing configuration changes, either intentional (operator debugging, key rollover) or accidental (signature expiry, missing DS record)¹.

¹We assume that the DNS response messages DNSViz extracts from authoritative servers of a zone are always authentic and observed errors/misconfigurations are not because of a failed Man-in-the-Middle attack.

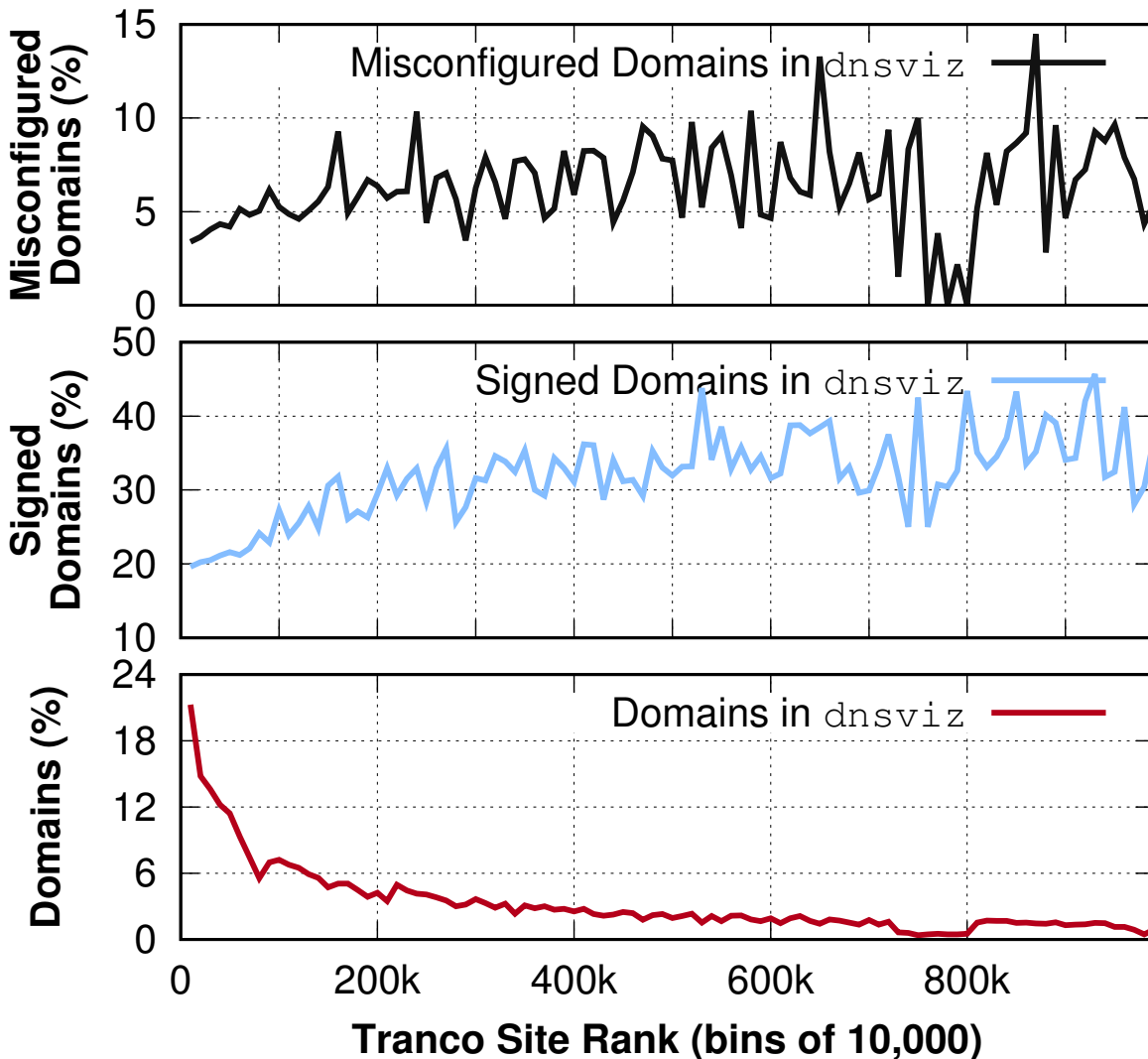


Figure 8.1: Percentage of domains from our DNSviz dataset appearing in each Tranco 1M bin [99], including the share of signed and misconfigured domains.

Snapshot Categorization

A *snapshot* corresponds to the state of a Query Domain at a specific time when DNSviz was invoked. We classify each into one of six categories:

- *sv (signed and valid)*. No DNSSEC errors are detected: all signatures (RRSIG) are current and verifiable, and a DNSSEC-validating resolver would successfully authenticate the domain.
- *svm (signed and valid with misconfiguration)*. For every query to each authoritative server, a valid authentication path up to root can be constructed; however, there exists at least one DNSSEC violation in the snapshot. Violations in this category typically do not cause valida-

tion failure but depends on the validating resolver. An example is exceeding NSEC3 iteration count [69], which most resolvers ignore [44]².

- *sb (signed and bogus)*. At least one query to at least one authoritative server of the domain fails cryptographic validation (e.g., an expired RRSIG). These errors produce a SERVFAIL response code, effectively breaking DNS resolution for validating resolvers.
- *is (insecure)*. The domain is *explicitly unsigned* in a valid way. For instance, the parent zone has a confirmed *proof of non-existence* (no DS record) that resolvers recognize. Thus, the zone is treated as “plain DNS” rather than bogus.
- *lm (lame)*. The name servers for the Query Zone do not respond correctly or cannot be resolved, resulting in a lame delegation. Both DNSSEC and standard DNS queries would fail.
- *ic (incomplete)*. The NS records for the Query Zone appear only within the zone itself but not in the parent zone. This typically disrupts normal delegation, leading to inconsistent or failed lookups.

Since our analysis centers on DNSSEC status, we primarily focus on four DNSSEC-related categories: *sv*, *svm*, *sb*, and *is*. The other two categories, *lm* and *ic*, reflect fundamental issues with the zone’s overall functionality rather than DNSSEC policy per se.

Domain Categorization

Domains can have multiple snapshots over time, each potentially falling into a different category. To highlight meaningful changes, we divide domains into two broad groups:

- *Changing Domains (CD)*: Domains that exhibit at least two snapshots with different DNSSEC-related error codes or statuses. Such differences indicate that something in the DNSSEC configuration changed between observations, whether intentionally (e.g., key rollover, DS update) or unintentionally (e.g., expired signatures). We analyze these domains extensively to understand how operators fix (or fail to fix) misconfigurations.
- *Stable Domains (SD)*: Domains whose snapshots all reside in the *same* DNSSEC category throughout the dataset. The rest of the domains with more than one snapshot are in this category as shown in [Table 8.1](#).

8.3 How Useful Is DNSViz?

DNSViz is widely recognized for identifying DNSSEC misconfigurations, but the extent to which operators use it to *fix* errors (rather than just diagnose them) remains unclear. To gauge its practical

²A minority of resolvers treat nonzero NSEC3 iteration counts as fatal, as highlighted by Daniluk et al. [44]. However, this is implementation-dependent [69].

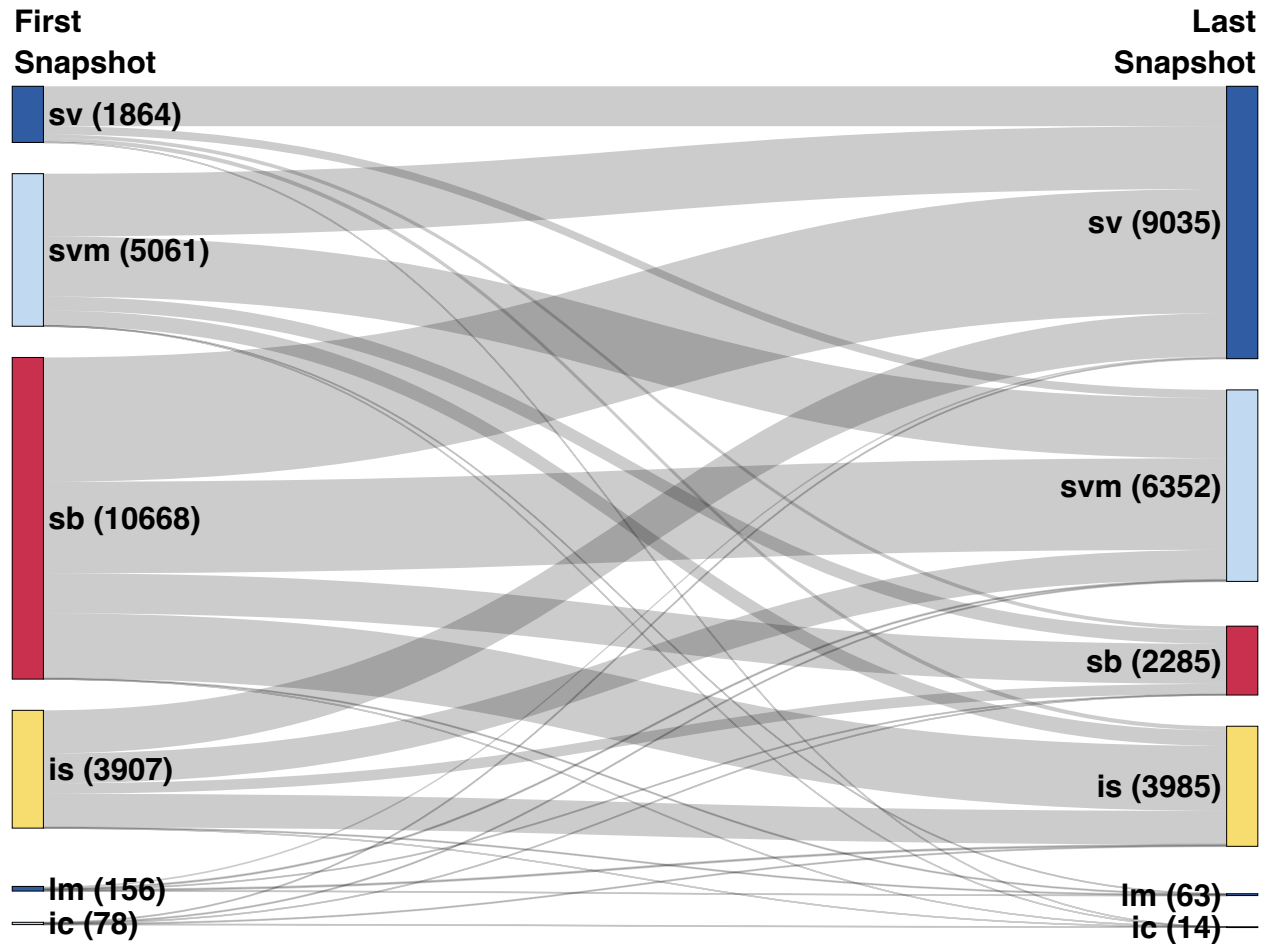


Figure 8.2: For domains in the CD category, comparing DNSSEC status between the first and last snapshots recorded by DNSViz. Note the moderate but significant fraction of domains that either enable or disable DNSSEC over time.

impact, we examine the *Changing Domains (CD)* subsetdomains with at least two snapshots and at least one shift in DNSSEC error codes.

Figure 8.2 illustrates how these domains transition from their *first* observed snapshot to their *last*:

- *Positive Trajectory.* Among the domains initially in sb (signed and bogus), 7,200 (67%, out of 10,668) eventually corrected their DNSSEC errors, reaching sv (signed and valid) or svm (signed and valid w/ misconfigurations). This indicates a meaningful fraction of operators do act on DNSViz’s diagnostics to remedy their zones.
- *Newly Signed.* Out of previously unsigned domains, 2,400 (62%, out of 3907) proceeded to enable DNSSEC by their final snapshot, suggesting DNSViz might also be used to guide the initial deployment of zone signing.

Previous State	Transitioned State	Cause	# of Domains
sv	sb	Total	4,064
		NS Update	272 (6.7%)
		Key Rollover	1,836 (45.2%)
		Algo. Rollover	1,230 (30.3%)
sv	is	Total	804
		NS Update	56 (7%)
		Key Rollover	241 (30%)
		Algo. Rollover	145 (18%)

Table 8.2: Causes of negative transitions from a valid (sv) DNSSEC state to either bogus (sb) or insecure (is). Key rollovers and algorithm changes together account for roughly two-thirds of $sv \rightarrow sb$ transitions, while a smaller fraction stems from nameserver (NS) updates. We observe a similar pattern for $sv \rightarrow is$ transitions as well.

On the other hand, DNSViz logs reveal *reverse* trends as well: about 650 domains (9.4%, out of 6,925) that began in a valid state (sv or svm) reverted to insecure (no DS records), and 588 (8.4%, out of 6,925) transitioned to sb (bogus) by their last snapshot. We analyze these cases more closely in [section 8.4](#) to explore the potential reasons (e.g., key rollover mishaps, expired signatures, or explicit DNSSEC deactivation).

Overall, [Figure 8.2](#) confirms that the dataset is quite invaluable in understanding the evolution of DNSSEC configuration on a large scale.

8.4 Exploring Negative Transitions

While many domains improve their DNSSEC configurations over time, a notable fraction transition from an sv (signed and valid) state to sb (signed and bogus) or is (insecure). To understand these “negative transitions” in more depth, we manually investigated a subset of such cases.

Key Triggers for Bogus States. Out of 4,064 instances where a domain shifted from sv to sb in any *two consecutive* snapshots, we identify three potential causes:

1. *Nameserver (NS) Updates.* Operators sometimes switch their authoritative servers (e.g., migrating to a new DNS operator). In doing so, they occasionally reuse old zone files to sign the zone or forget to upload new DS records at the registrar, resulting in invalid RRSIG or mismatched DS entries. To determine this, we looked into the NS record of the Query Zone for each snapshot and counted the number of times a shift to bogus state coincides with a change in NS record. For instance, `apopo.website` maintained sv status for 28 snapshots but switched to a new nameserver on the 29th, accidentally resigning the zone with stale DNS content. That single oversight triggered an invalid RRSIG error, dropping the domain to sb.
2. *Key Rollovers.* Regularly updating cryptographic keys helps prevent key compromise and protects the integrity of the “chain of trust”. However, if administrators fail to synchronize DS and

Category	Subcategory	# of snapshots (%)	# of domains (%)
Delegation	⑤ Missing KSK for Algorithm	63,004 (8.4%)	25,102 (7.9%)
	① Invalid Digest	1,103 (0.15%)	466 (0.15%)
Key	③ Inconsistent DNSKEY b/w Servers	19,330 (2.6%)	6,393 (2%)
	Revoked Key	302 (0.04%)	45 (0.014%)
	Bad Key Length	108 (0.01%)	21 (0.007%)
Algorithm	② Incomplete Algorithm Setup	6,859 (0.9%)	1,883 (0.5%)
Signature	Missing Signature	38,662 (5.2%)	18,306 (5.7%)
	④ Expired Signature	11,670 (1.6%)	4,494 (1.4%)
	⑥ Invalid Signature	10,336 (1.4%)	3,152 (1%)
	Incorrect Signer	1,961 (0.3%)	550 (0.2%)
	Not Yet Valid Signature	663 (0.09%)	125 (0.04%)
	Incorrect Signature Labels	99 (0.01%)	25 (0.008%)
	Bad Signature Length	42 (0.006%)	13 (0.004%)
TTL	⑧ Original TTL Exceeds RRSet TTL	4,999 (0.7%)	1,769 (0.6%)
	TTL Beyond Expiration	2,556 (0.3%)	864 (0.3%)
NSEC(3)	⑦ Missing Non-existence Proof	65,378 (8.7%)	17,768 (5.6%)
	Incorrect Type Bitmap	18,218 (2.4%)	4,070 (1.3%)
	Bad Non-existence Proof	9,678 (1.3%)	3,255 (1%)
NSEC(Only)	Incorrect Last NSEC	405 (0.05%)	214 (0.07%)
NSEC3(Only)	⑨ Nonzero Iteration Count (NZIC)	215,036 (28.8%)	62,870 (19.7%)
	Inconsistent Ancestor for NXDOMAIN	2,296 (0.3%)	1,410 (0.44%)
	Incorrect Closest Encloser Proof	1,278 (0.17%)	415 (0.13%)
	Invalid NSEC3 Hash	456 (0.06%)	200 (0.06%)
	Invalid NSEC3 Owner Name	301 (0.04%)	152 (0.05%)
	Incorrect Opt-out Flag	186 (0.02%)	64 (0.02%)
	Unsupported NSEC3 Algorithm	74 (0.01%)	11 (0.003%)
-	w/ at least One DNSSEC Error	296,813 (39.7%)	81,805 (25.6%)

Table 8.3: Prevalence of various DNSSEC error types in our DNSViz dataset, covering 319,277 second-level and their lower-level domains (total 747,455 snapshots); for example, the “Nonzero Iteration Count” in NSEC3 appears in 215,036 snapshots (28.8%) spanning 62,870 domains (19.7%). The markers (①–⑨) highlight notable or especially frequent misconfigurations in each category, which are also presented in Figure 8.4. Last row represents the # of snapshots (and domains) with at least 1 error from above categories.

DNSKEYs updates properly, a zone can briefly or indefinitely become bogus. To identify negative transitions due to key rollover, we examined the DNSKEY record of the Query Zone for each snapshot and counted how often a shift to sb coincided with a DNSKEY change. For example, `ciast.edu.my` had a valid DNSSEC configuration with one KSK and one ZSK, but an attempted ZSK rollover introduced four additional DNSKEYs, resulting in an inconsistent key set and a misconfigured DNSSEC setup. This is a common mistake during rollovers, as reported in [81].

3. *Key Algorithm Rollovers.* A more complex type of rollover where the algorithm associated with the DNSKEY record (e.g., from RSA to ECDSA) is changed as well, not just the key itself [78]. Operators can do this to switch to a more secure and/or shorter key and signatures or use multiple algorithms simultaneously to prevent validation failure due to unsupported algorithms. However, given DNSSEC algorithm rollover is a complicated process [125], this creates more window to break DNSSEC validation.

To identify whether any of the transitions to bogus state was caused by this, we looked into the DNSKEY algorithms of the Query Zone for each snapshot and counted the number of times a shift in state coincides with an update in DNSKEY algorithm. Out of 4,064 such transitions, we observed a key algorithm change in 1,230 (30.3%) instances.

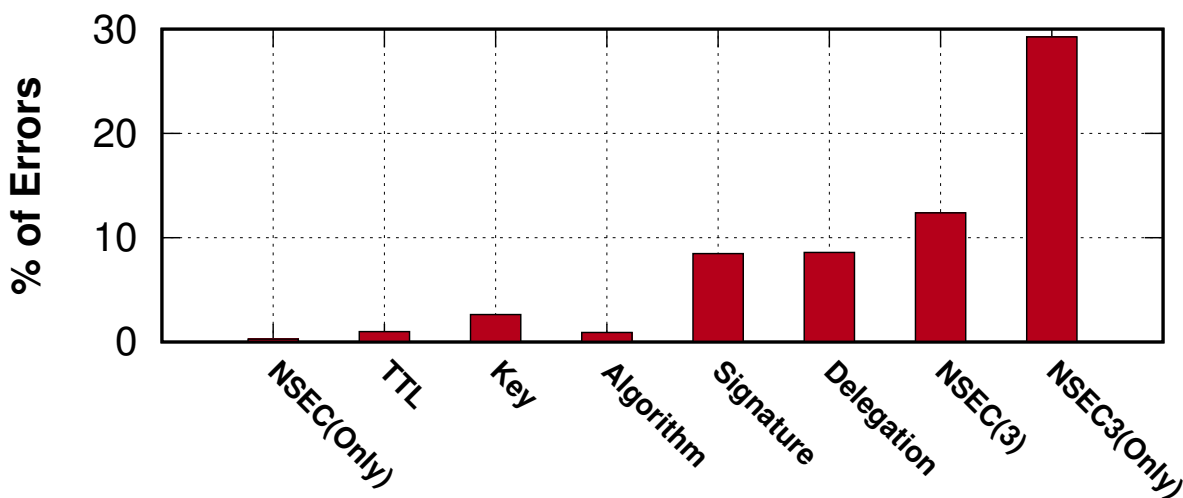


Figure 8.3: Percentage of DNSSEC error categories out of 747,455 snapshots for second-level and lower domains. NSEC(3)-related errors, including nonzero iteration counts and missing negative proofs, account for a significant fraction. We also see many delegation failures and signature (RRSIG) issues.

Collectively, these three categories account for 81% of the observed negative transitions (see [Table 8.2](#)).

Switching to Insecure. Similarly, we identified 7% of cases where valid domains transitioned to `is` following nameserver updates, while in 30% and 18% cases, domains removed their DS record and turned off DNSSEC following key rollover and algorithm rollover respectively. Moreover, we found 610 domains that first became bogus from valid and then disabled DNSSEC permanently as per their latest snapshot. Although we cannot confirm the exact administrative intentions, these patterns highlight usability challenges and point to potential mismanagement during server migrations or key transitions.

8.5 Error Prevalence in DNSSEC

To identify which DNSSEC issues arise most frequently, we examine 47 distinct DNSViz error codes spanning our entire dataset. [Table 8.3](#) groups them under 8 parent categories (e.g., “Delegation,” “Key,” “Signature,” etc.) and 26 subcategories to facilitate broader analysis.

- *NSEC/NSEC3 Errors.* NSEC and NSEC3 issues are the single most prominent category. In particular, the “Nonzero Iteration Count” code appears in 215,036 snapshots (covering 62,870 domains) by far the largest single error. Another 65,378 snapshots (17,768 domains) exhibit a

“Missing Non-existence Proof” error, indicating that negative responses (NXDOMAIN, NODATA) lack the appropriate NSEC/NSEC3 records. These two alone underscore that negative-proof mechanics remain a persistent challenge for administrators.

- *Delegation Errors.* Errors in establishing a valid chain of trust also loom large. For instance, having a “Missing KSK for Algorithm” (i.e., a DS record referencing a key algorithm not actually present) appears in 63,004 snapshots and affects 25,102 domains. Meanwhile, “Invalid Digest” (a DS record whose hash does not match any DNSKEY) occurs in 1,103 snapshots across 466 domains. Although smaller in absolute terms, such mismatches directly trigger SERVFAIL and are often among the most critical to fix.
- *Signature Anomalies.* Signature-related issues are widespread: “Missing Signature” arises in 38,662 snapshots (18,306 domains), while “Expired Signature” affects 11,670 snapshots (4,494 domains), and “Invalid Signature” appears 10,336 times (3,152 domains). Collectively, these reflect a broad set of operational lapses, from forgetting to re-sign the zone before signatures expire to incorrectly applying the private key.
- *Key-Related Inconsistencies.* Among the errors in the “Key” category, the most common is having inconsistent DNSKEY sets across different authoritative servers (19,330 snapshots, 6,393 domains). Although less prevalent than major NSEC (3) or signature issues, this inconsistency can break validation and lead to intermittent SERVFAIL. Other key-level errors (e.g., “Bad Key Length,” “Revoked Key”) are relatively rare but still appear in hundreds of snapshots.
- *Algorithmic Gaps.* A noteworthy 6,859 snapshots (1,883 domains) exhibit “Incomplete Algorithm Setup,” where an algorithm referenced by the zone is not consistently present in all RRSIG records or does not align with the parent zone’s DS records. This often arises when an operator attempts to upgrade or change algorithms but omits essential steps (e.g., forgets to sign the zone with all unique algorithms present in the zone [126]).

Figure 8.3 provides a visual summary, highlighting the disproportionate frequency of NSEC3 iteration count violations and missing RRSIG sets. These findings suggest that some DNSSEC intricacies such as negative proof management remain persistent trouble spots.

8.6 Understanding Error Resolution Patterns

While identifying the *kinds* of DNSSEC errors is useful, understanding *how quickly* they are addressed is equally crucial. Table 8.4 shows the number of transitions in the CD set between any two consecutive snapshots and median time for these transitions. As we can see, transition from bogus to valid has a median time of only 0.7 hours. We further identified 1,856 domains (out of 4,064) that transitioned from a valid to a bogus state and subsequently recovered (back to valid). For this subset, the median time for the valid to bogus transition is 238.6 hours, whereas the median for bogus to valid is 0.6 hours. This clearly indicates that administrators tend to respond much more promptly to errors that cause resolution failures.

From \ To	sv	svm	sb	is
sv	–	1310 34.2h	4064 133.7h	804 58.6h
svm	3132 73.4h	–	5573 104.2h	1486 71.8h
sb	8052 0.7h	8065 0.87h	–	3922 1.6h
is	2150 2.7h	2097 3.3h	2001 1.8h	–

Table 8.4: Adjacency matrix of state transitions. Each cell shows # of transitions (top) and median time in hours (bottom). A dash (–) means no observed transition. We can observe administrators reacting promptly when their domain becomes bogus.

To further understand which individual error code is easier to fix or needs quick operator attention, we now focus only on domains that eventually corrected their DNSSEC status, transitioning from sb or svm to sv. For each error code with at least 100 *fix* instances, we define two timestamps:

- t_1 : The first snapshot with sb (signed-bogus) or svm (signed-valid w/ misconfig) category where the error is present. If this snapshot is sb, the error is deemed *critical* (leading to SERVFAIL); under svm, it is *non-critical*.
- t_2 : The first snapshot in which the error is no longer detected and the domain is fully valid (sv).

We compute the time difference between snapshots t_1 and t_2 for each instance and plot the distributions in [Figure 8.4](#). We note two *limitations*:

Overestimation in fix time. Since DNSViz rescans are triggered by users, a domain might fix an error shortly after t_1 yet go unrecorded until the next user-initiated scan. Hence, we could overestimate problem durations (or miss short-lived fixes that revert). Nevertheless, as shown in [Figure 8.5](#), 65% of the domains have a median time difference of less than a day between their consecutive two snapshots which suggests that these scanning gaps are not very common, so the overall trends remain informative.

Underestimation of critical errors. Some error codes in svm category may also lead to SERVFAIL. It is impossible to identify them deterministically without complete validation context. For example, “Incomplete Algorithm Setup” error code may or may not produce SERVFAIL depending on whether the validating resolver supports one of the algorithms that can form a valid chain of trust up to root. Therefore, our estimation of critical errors may potentially be an underestimation of actual number of critical errors.

Regardless of the caveats, we make the following useful observations from this analysis:

- *Critical Errors Are Fixed Sooner.* Delegation problems such as DS and DNSKEY mismatches that break trust chains are often resolved within 2–3 days in 80% of cases. Similarly, in 80% cases,

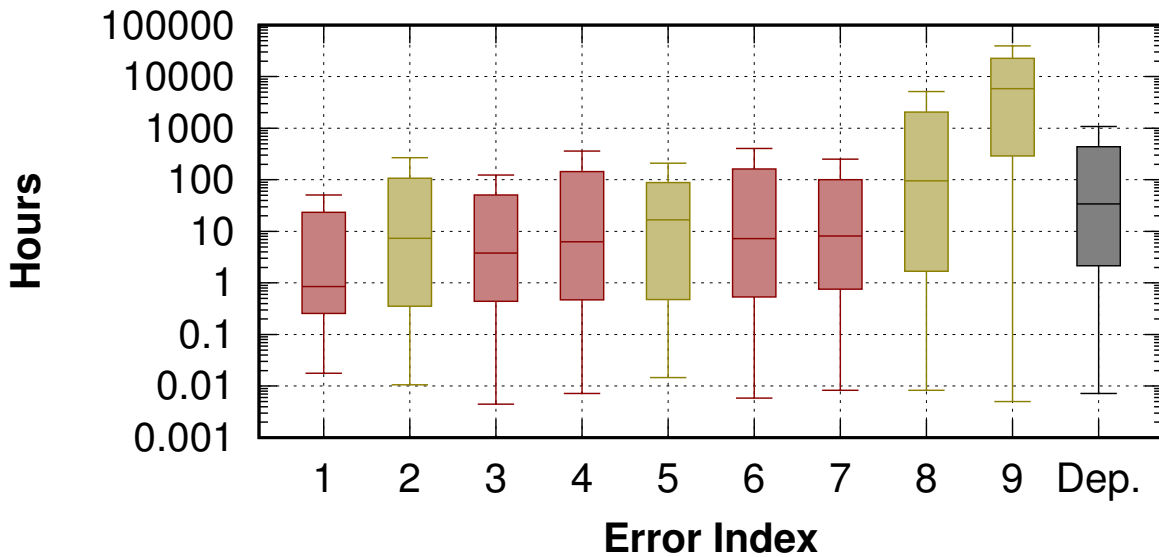


Figure 8.4: Resolution times for critical (red boxes) vs. non-critical (orange boxes) DNSSEC errors. The numbers on the x-axis (19) are indices that map directly to the numbered error categories detailed in [Table 8.3](#). Note that the median time to deploy DNSSEC (black box) is more than a day; this is likely attributed to the operational challenges in DNSSEC identified by prior works [34].

Category	# of Domains	
	w/ at least One Snapshot	Not Resolved
sb (signed & bogus)	15,209	2,731 (18%)
svm (signed & valid w/ misconfig.)	9,052	5,603 (61.9%)
is (insecure)	7,149	2,611 (36.5%)

Table 8.5: For each DNSSEC state of interest, we list (i) the total number of domains that appear in that state at least once, and (ii) how many never rectified the issue or re-enabled DNSSEC according to their latest snapshot in our dataset.

inconsistent DNSKEY sets take about four days while expired or invalid signatures take a bit longer (10 days). This is somewhat surprising, since re-signing can be automated, suggesting that some operators may still use manual signing procedure.

- *Non-Blocking Errors Are Overlooked.* Errors typically not causing SERVFAIL such as TTL mismatch with Original TTL field in RRSIG record or a nonzero NSEC iteration count persist for weeks or months (60 and 250 days, respectively, in 80% of occurrences). Because they do not break resolution, administrators often de-prioritize them indefinitely.

Finally, [Table 8.5](#) shows a large fraction of domains *never* remedied their DNSSEC errors during our observation period. Roughly 18% of domains once in sb remained in that status, and 36.5% of those that once turned off DNSSEC (i.e., reverted to is) never re-enabled signing. These persistent shortcomings highlight the operational complexities of DNSSEC and the tendency for domains to abandon signing when errors become unmanageable.

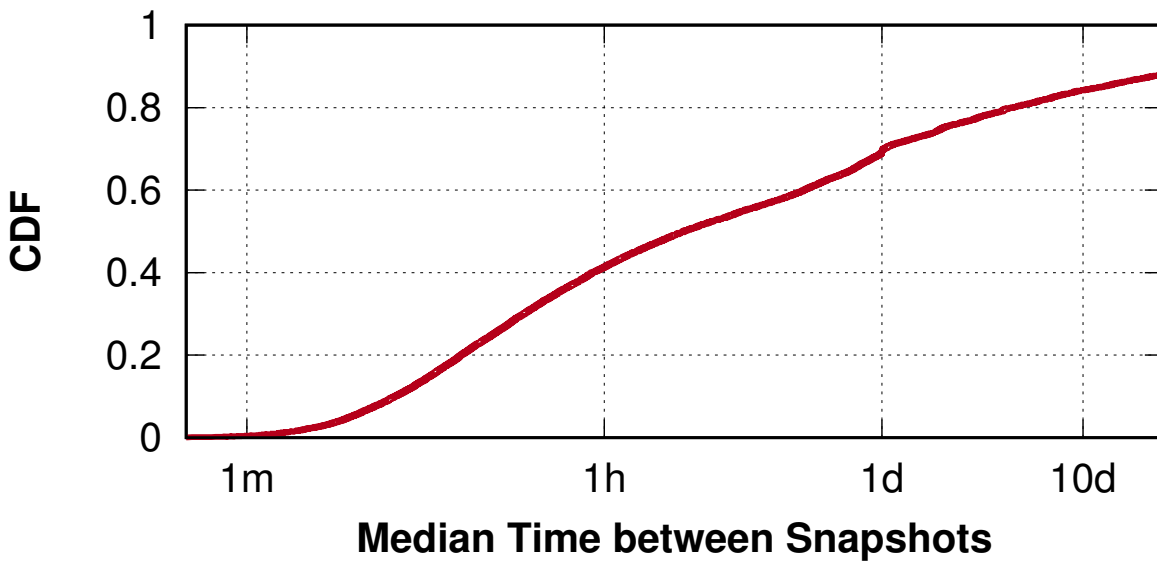


Figure 8.5: 65% of the domains have a median time difference of less than a day between their consecutive two snapshots

Implications. Overall, despite DNSViz’s thorough diagnostics, the path from “error found” to “error fixed” often remains lengthy or incomplete. Critical errors do tend to be addressed (albeit not instantly), but non-critical issues remain in place for long durations, and many domains drop DNSSEC altogether rather than maintain it. This underscores the need for a more *automated*, domain-aware remediation mechanism. We explore such a solution in the next chapter.

Chapter 9

DFixer: A Comprehensive DNSSEC Fixing Framework

We introduce DFixer, a framework that pinpoints DNSSEC misconfigurations and translates these diagnoses into actionable BIND commands. Compared to purely diagnostic tools (e.g., DNSViz) or naive Large Language Model (LLM) suggestions, DFixer provides concrete, verifiably correct fixes for the *root causes* of DNSSEC errors.

9.1 Rationale and Key Observations

The design of DFixer rests on two central insights:

1. *Common Patterns of DNSSEC Errors.* Despite DNSSEC’s complexity, many domains exhibit the same recurring mistakes: stale DS records, incomplete algorithm transitions, expired RRSIG signatures etc. Consequently, the distinct *root causes* are relatively few, even if individual domains show multiple secondary errors.
2. *Small Repertoire of BIND Commands.* Although DNSSEC provisioning can be intricate, BIND ultimately relies on a limited set of commands (e.g., `dnssec-signzone` for signing the zone, `dnssec-keygen` for generating keys, `dnssec-dsfromkey` for generating DS records). These suffice for most scenarios if one knows how to apply them correctly and *in the right order*.

Leveraging these observations, DFixer pairs real-world DNSViz misconfiguration data with a manually curated knowledge base of *error-to-command* mappings. The end goal: for each snapshot of a domain, DFixer identifies the root cause(s), devises a minimal remediation plan, and then presents precise commands to achieve a valid DNSSEC state.

9.2 Why Existing Tools and Naive LLMs Fall Short

Diagnostic-Only Tools. Well-known utilities, such as DNSViz [54], Zonemaster [132], DNSSEC Debugger [48] excel at detecting DNSSEC anomalies. However, they typically produce extensive

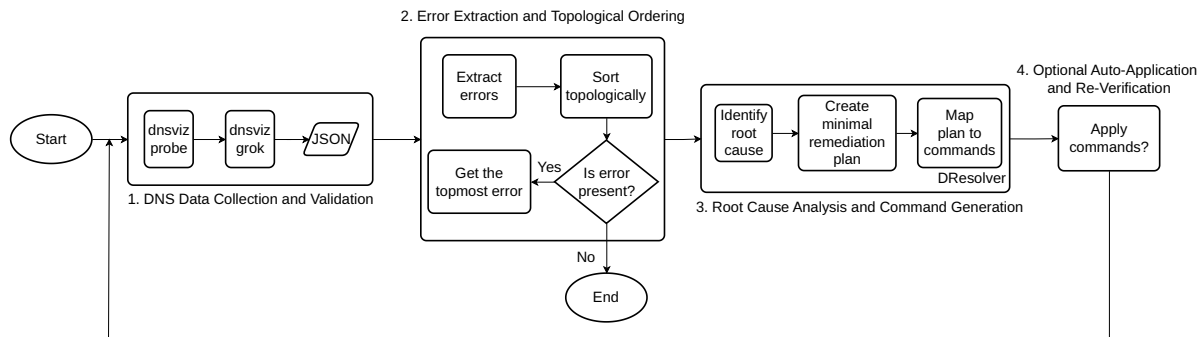


Figure 9.1: Overview of the DFixer pipeline. Each iteration collects DNSSEC data with probe and grok filters relevant error codes, resolves root causes via DResolver, and produces fix commands. The process repeats until no blocking errors remain.

lists of error messages that do not isolate the few root causes. This can overwhelm novice administrators in understanding *which* errors directly break the chain of trust and *which* are mere side effects. For instance, a simple extraneous DS record can trigger over a dozen distinct error messages. Without further guidance, operators may struggle to fix the *correct* underlying issue.

Naive LLM-Based Approaches. Large Language Models (e.g., GPT-4o) show promise in synthesizing text, and are able to provide highly dynamic and interactive solution. Thus, we wanted to verify how naive LLM-based prompt engineering techniques perform in our problem domain. As an experiment (Appendix [Appendix G](#)), we used GPT-4o to help us diagnose and fix common DNSSEC-related problems and observed that it often:

- *Produced generic suggestions* (e.g., “Verify your DS record or check whether your zone is signed with the correct key”).
- *Hallucinated details* such as fictitious key tags, algorithms, or flags when given partial logs or images from DNSViz.
- *Missed interdependencies* among multiple errors, suggesting partial fixes that fail to address the root cause.

Meanwhile, existing natural-language-to-shell-command tools [8] from NLP research fail to capture DNSSEC-specific semantics. Thus, a carefully curated mapping from *diagnosed error* to *BIND command sequence* remains indispensable.

Limitation. Our findings about naive LLMs should be read as illustrative rather than definitive; in Appendix [Appendix G](#), the examples show generic advice, hallucinated key tags or algorithms, and confusion between parent and child issues, but they do not constitute a comprehensive cross-model benchmark.

First, the baseline uses a single general-purpose model and a small prompt family, without any knowledge retrieval, or fine-tuning mechanism. Second, LLM outputs are non-deterministic and models evolve; we did not control seeds, compare vendors, or sweep sampling parameters, so results should not be over-generalized. Finally, several pitfalls we observe reflect *intrinsic DNSSEC operational complexity*, including multi-phase key or algorithm rollovers, not merely LLM shortcomings [78, 125]. As future work, we plan to evaluate whether DFixer can be extended into an agentic AI system to further enhance its usability.

9.3 Design and Workflow of DFixer

DFixer follows an *iterative* pipeline, illustrated in [Figure 9.1](#). Below, we highlight the major steps:

1. *DNS Data Collection and Validation*: DFixer invokes the `dnsviz probe` and `dnsviz grok` commands on the target domain.
2. *Error Extraction and Topological Ordering*: DFixer parses the grok JSON, isolates DNSSEC-related error codes, and constructs a dependency graph. Many errors are *cascading* (e.g., a missing key can trigger numerous signature or delegation errors). By topologically sorting these errors, DFixer ensures that it addresses the root cause first.
3. *Root-Cause Analysis and Command Generation*: A module called DResolver first identifies the root cause from zone context i.e., topmost error code, presence or absence of some companion errors, and state of the zone. For example, if the topmost error picked is “Revoked Key”, DResolver checks whether:
 - a “No secure entry point” error is also present. This indicates that there exists a DS record that is linked to the revoked key which needs to be purged.
 - there are other valid KSK present. If yes, just removing any associated DS record and deactivating the revoked key is fine. Otherwise, a new KSK key pair needs to be generated along with the deactivation of revoked key.

In this way, DResolver then synthesizes a remediation plan with a set of instructions (e.g., “Generate a new KSK, upload new DS, remove the incorrect DS, re-sign the zone”). Finally, it translates each instruction in the plan to *BIND commands*.¹ The associated parameters for these commands are populated from zone context.

4. *Optional Auto-Application and Re-Verification*: The operator can run DFixer in two modes:
 - *Suggest Only (dry-run)*: DFixer prints out the exact commands (with placeholders for key file or zone file paths).
 - *Auto-Apply*: If the operator grants the necessary privileges (e.g., shell access) and local data paths, DFixer can directly execute these commands (except the ones that require manual interaction), re-running probe and grok after each fix.

¹`dnssec-keygen`, `dnssec-signzone`, `dnssec-settime`, `dnssec-dsfromkey`

This iterative re-verification continues until no DNSSEC errors remain. [Figure F.1](#) in Appendix shows a sample fix plan for a domain when its only KSK has the REVOKED flag on and is linked to a DS record.

9.4 System Model and Assumptions

In parallel with these pipeline stages, DFixer makes certain assumptions about the operator and environment.

Actor and Capabilities. We assume a *DNS operator* with `rwX` privileges on an authoritative server (running BIND). This operator has basic DNSSEC knowledge (record types, signing logic) and wants an automated solution to *pinpoint* errors and *fix* them.

Objectives. DFixer aims for:

- *Correctness*: Precisely identify root causes and map them to valid BIND commands.
- *Comprehensiveness*: Cover a broad range of DNSSEC failures, from stale DS records to incorrect algorithm rollovers.
- *Extensibility*: Although focused on BIND, our approach can be adapted for other servers like NSD or PowerDNS by creating analogous error-to-command mappings, which will be shown in [subsection 9.6.6](#).

9.5 ZReplicator: Replicating Real Errors in a Local Environment

Although DNSViz data reveals myriad real DNSSEC errors, DFixer cannot directly correct zones owned by third parties. To *fully* validate whether DFixer’s recommended fixes actually work, we developed ZReplicator, a custom local replication module.

Motivation. We want to recreate real-world DNSSEC errors in a sandbox where we can:

1. *Inject* the exact misconfigurations (e.g., stale DS, invalid signatures) from DNSViz logs.
2. *Run DFixer*, apply suggested commands, and confirm that the zone becomes valid after re-checking.

This offers a true “test-fix-verify cycle, unlike passively analyzing logs.

Dataset	# of Snapshots	$GE \neq \emptyset$	$IE \subseteq GE \ \& \ IE \neq \emptyset$	$AE = \emptyset \ \& \ IE \subseteq GE$	RR	FR
NZIC Only (S1)	168,482	166,591	166,470	166,470	98.81%	100.0%
Remaining (S2)	128,331	119,362	101,003	100,998	78.71%	99.99%
Total	296,813	285,953	267,473	267,468	90.11%	99.99%

Table 9.1: Performance of ZReplicator and DFixer among the snapshots with $IE \neq \emptyset$.

Replication Process. ZReplicator proceeds as follows:

1. *Create a Base Zone:* We first set up a fully signed BIND zone (e.g., a.com) to serve as the root for our local environment. This enables us to automate communication with parent zone (upload DS record) during evaluation.
2. *Emulate Parent and Child Zones:* We create sub-zones (e.g., par.a.com and inv-chd.par.a.com) mirroring the parameters (# of DNSKEY records and their flags, algorithms, and key size, # of DS records, and their digest type, usage of NSEC vs. NSEC3, etc.) extracted from the real domain’s DNSViz logs.
3. *Inject DNSSEC Errors:* By tweaking the zone files (e.g., setting an RRSIG expiration in the past, removing a DNSKEY on one nameserver), we replicate the target error scenario. This can also include multi-server inconsistencies or algorithm mismatches.

This local replication ensures we can *empirically* verify DFixer’s effectiveness and correctness. In [section 9.6](#), we detail how we use ZReplicator to rigorously evaluate DFixer across thousands of misconfiguration scenarios extracted from DNSViz logs.

9.6 DFixer: Evaluation

We now present a comprehensive evaluation of DFixer, leveraging the large-scale DNSViz dataset described in [section 8.1](#). Our primary goals are to assess whether DFixer (1) accurately reproduces real-world DNSSEC errors through the ZReplicator module ([§9.5](#)), and (2) effectively fixes these replicated misconfigurations in an automated manner.

9.6.1 Evaluation Pipeline

As summarized in [Table 8.1](#), our DNSViz dataset comprises 747K snapshots (JSON files) for SLD and lower domains.

[Figure 9.2](#) illustrates our experimental pipeline, which proceeds as follows:

1. *Select JSON Snapshot.* We iterate over the DNSViz dataset (747K snapshots for SLD+ domains), taking one JSON file at a time.

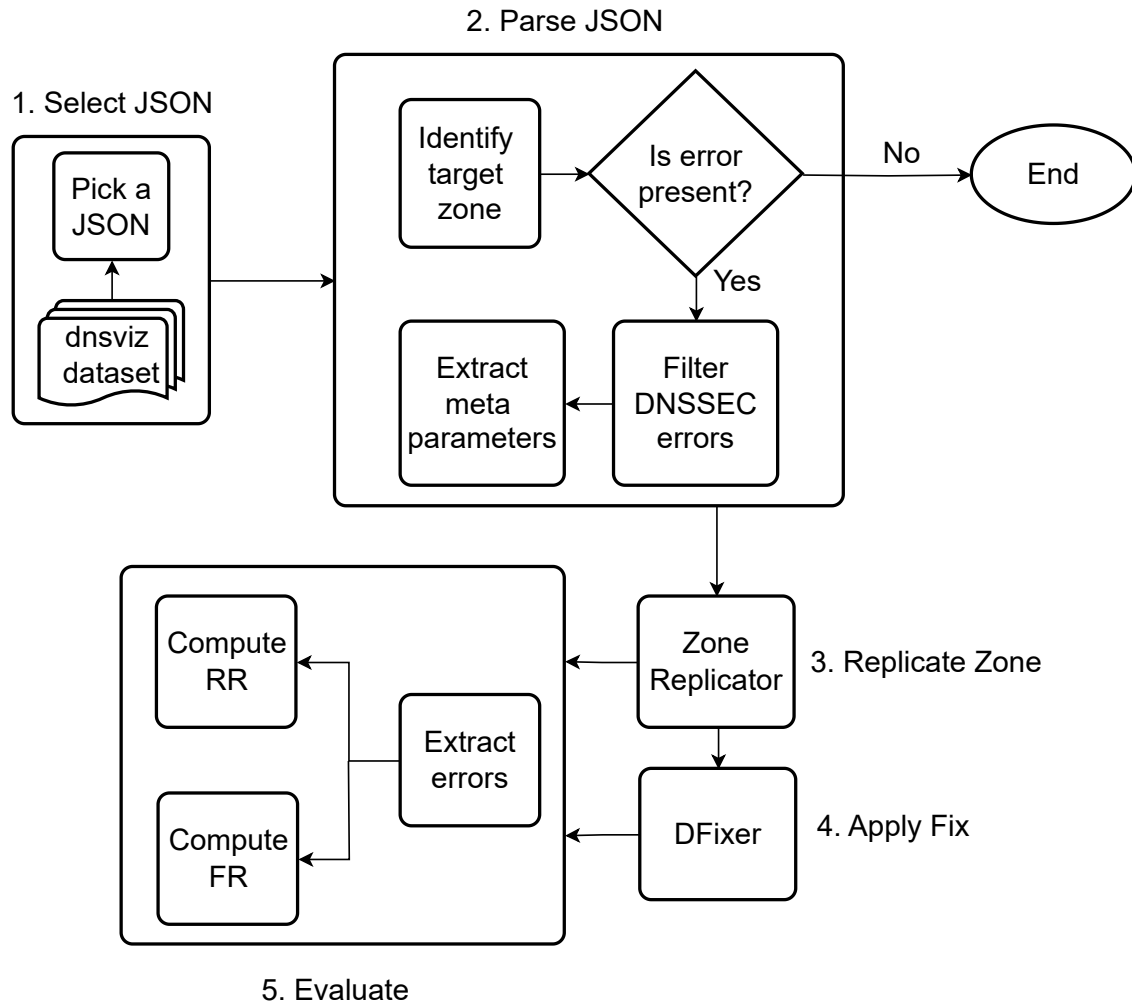


Figure 9.2: High-level illustration of our evaluation pipeline. Each JSON file from the DNSViz dataset is parsed, replicated locally via ZReplicator, and then processed by DFixer.

2. *Parse JSON.* For each snapshot, we extract all *DNSSEC-related* errors present in the leaf zone (“target zone”) such as invalid RRSIG or missing DNSKEY. We also record *zone meta-parameters* that are critical for accurate replication and subsequent fixes:

- *DNSKEY properties:* Number of keys, their algorithms, and key sizes. These details are essential for generating correct BIND commands (e.g., `dnssec-keygen` arguments) so that we can match the zone’s actual cryptographic setup.
- *Delegation settings:* Parent-zone data, such as DS record digest types and algorithms, needed for consistent DS creation or removal.
- *NSEC vs. NSEC3 usage:* Whether the zone uses NSEC or NSEC3 (and if NSEC3, value of iteration count, salt, and flags). This information ensures that commands like `dnssec-signzone`

reflect the same negative-proof mechanism as in the original zone.

3. *Replicate Zone via ZReplicator.* We feed the extracted error codes and zone parameters to our ZReplicator module, which locally reproduces the misconfigurations using two authoritative BIND name servers to mimic multi-server inconsistencies.
4. *Apply DFixer.* DFixer runs on the replicated zone, iteratively applying fixes.
5. *Evaluate.* Finally, we *re-invoke* `probe/grok` on both the replicated (pre-fix) and rectified (post-fix) zones to verify: (1) whether ZReplicator successfully replicated the original errors, and (2) whether DFixer’s recommended commands eliminated those errors.

9.6.2 Evaluation Metrics

We first define three sets of error codes:

- *Intended Errors (IE):* Set of DNSSEC errors present in the target zone
- *Generated Errors (GE):* Set of DNSSEC errors present in the corresponding replicated zone by ZReplicator
- *After-fix Errors (AE):* Set of DNSSEC errors present after applying DFixer on the replicated zone

Given the above sets, we define two core metrics:

- **Replication Rate (RR).** We measure how often ZReplicator successfully reproduces *all* intended errors from the original snapshot².

$$RR = \frac{\# \text{ of cases where } IE \subseteq GE \text{ and } IE \neq \phi}{\# \text{ of JSON where } IE \neq \phi}$$

A higher RR implies comprehensive coverage of real-world misconfigurations.

- **Fix Rate (FR).** We also measure how often DFixer eliminates all generated errors in the replicated zone:

$$FR = \frac{\# \text{ of cases where } IE \subseteq GE \text{ and } GE \neq \phi \text{ and } AE = \phi}{\# \text{ of cases where } IE \subseteq GE \text{ and } GE \neq \phi}$$

A higher FR indicates that DFixer’s prescribed commands are both accurate and complete for the replicated scenarios.

²In certain cases, ZReplicator may produce benign companion errors alongside the errors in *IE* that does not have any bearing in DFixer’s corrective instructions. For instance, to simulate a “Missing KSK for algorithm” error, ZReplicator creates a DS record referencing a non-existent key with an algorithm ‘a’ that is not present in the zone, rather than precisely matching the algorithm found in the DS record of the target zone. This can trigger “Missing Signature for Algorithm” error when no signatures for ‘a’ exist.

9.6.3 Implementation and Environment:

We implement the entire pipeline inside Docker containers running Ubuntu 22.04. Each container hosts two BIND (v9.18) instances to emulate multi-server authoritative zones. We run these containers on a server with 38 CPU cores and 187 GB of RAM, processing 747K DNSviz JSON files within 36 hours. We will open-source our container images and orchestration scripts to encourage reproducibility and future DNSSEC research.

9.6.4 Experiment Results

We evaluate DFixer using 747K snapshots for SLD and lower level subdomains in our dataset; we summarize the result in [Table 9.1](#). Among these, 296,813 contain at least one DNSSEC-related error. As shown in [Table 8.3](#), the “Nonzero Iteration Count” (NZIC) error is disproportionately common compared to other issues. Because fixing NZIC is relatively straightforward (mostly involves with re-signing the zone with zero iterations), we divide the dataset into two subsets to give a more representative view:

- *S1*: Snapshots where NZIC is the *only* DNSSEC error.
- *S2*: All other snapshots, which contain either NZIC alongside other errors or completely different errors.

Replication Rate: From the 296,813 snapshots that have at least one DNSSEC-related error, we observe 2,058 unique error combinations. Our ZReplicator module successfully replicates 90.11% of these. In subset *S1* (NZIC-only), ZReplicator achieves a near-perfect replication ratio of 98.81%. In contrast, *S2* (which excludes NZIC-only scenarios) reflects more realistic conditions and attains a slightly lower replication ratio of 78.71%. For the remaining 21.29% *S2* snapshots where ZReplicator failed to replicate all the intended errors ($IE \not\subset GE$), it was able to generate a subset of intended errors ($GE \subset IE$ and $GE \neq \emptyset$) in 67.18% cases, while for the remaining 32.82% cases, it could not generate any of the intended ones ($GE = \emptyset$). We discussed the reasons behind this in [subsection 9.6.5](#).

Fix Rate: DFixer achieves an overall fix rate of 99.99% across all snapshots, indicating that once a misconfiguration is reproduced, DFixer effectively resolves it. Even in the more complex *S2* subset, DFixer’s fix rate remains near-perfect; for the small fraction of *S2* snapshots (5) that DFixer failed to resolve, we manually inspected them and found the parent zone to have a bogus state with DS records but no DNSKEY. In such scenarios, parent zone operators must address their errors before the child zone administrators can proceed with DFixer.

Instruction	1 st iteration	2 nd iteration	3 rd iteration	4 th iteration
Sign the zone	62,406 (41.67%)	13,845 (89.98%)	1,148 (62.19%)	7 (19.44%)
Remove the incorrect DS record	46,242 (30.87%)	1,319 (8.57%)	668 (36.19%)	29 (80.56%)
Upload the DS record	14,066 (9.39%)	117 (0.76%)	12 (0.65%)	-
Generate a KSK	13,148 (8.78%)	83 (0.54%)	-	-
Synchronize the DNS authoritative server	11,391 (7.61%)	-	-	-
Generate ZSK	1,491 (1.0%)	-	-	-
Reduce TTL of a specific record	947 (0.63%)	1 (0.01%)	-	-
Remove the revoked key	82 (0.05%)	22 (0.14%)	18 (0.98%)	-

Table 9.2: Instructions issued by DFixer during its iterative remediation process in the S2 subset (i.e., zones with more complex DNSSEC errors). This highlights how DFixer repeatedly removes incorrect DS records, re-signs zones, or updates keys until all misconfigurations are resolved.

Command Distribution in DFixer: DFixer uses an iterative remediation strategy, issuing multiple rounds of BIND commands until all DNSSEC errors are resolved (Figure 9.1). In our experiments, we find that *no zone required more than four iterations to become fully compliant*, and most errors were cleared by the second iteration—highlighting DFixer’s efficiency even for interdependent misconfigurations. Table 9.2 details the most frequently applied instructions in the S2 subset (zones with non-trivial DNSSEC problems). Several insights stand out:

- *Single-Iteration vs Multi-Iteration Fix.* A single iteration is sufficient when all observed errors share one root cause. For example, a zone with a revoked KSK, missing DNSKEY signature, and an invalid DS referencing a non-existent KSK can be repaired in one pass by 1) generating a new KSK pair, 2) re-signing with the correct keys, 3) publishing the correct DS, 4) deactivating the bad DNSKEY, and 5) removing the incorrect DS. In this case, a single iteration suffices because all the observed errors share the same underlying cause: faulty KSK setup. On the other hand, zones with multiple independent errors typically require multiple iterations to achieve full resolution. For example, if a zone simultaneously exhibits a nonzero NSEC3 iteration count error and an extraneous DS record, DFixer will proceed incrementally: in the first iteration, it removes the extraneous DS record, and in the second, it re-signs the zone with an NSEC3 iteration count of zero.
- *Recurring Need for Zone Signing.* Unsurprisingly, “Sign the zone” dominates across iterations, accounting for 41.67% of all commands in the first pass, 89.98% in the second, and 62.19% in the third. Zone signing is inherently necessary whenever cryptographic material changes (e.g., after removing or adding keys, regenerating signatures).
- *DS Record Management is Second Most Common.* Removing incorrect DS records (30.87% in the first iteration and 8.57% in the second) is critical for fixing common misconfigurations, such as stale or mismatched DS entries. This finding corroborates prior work showing that nearly 30% of DNSSEC-signed domains struggle with DS-related errors [33]—often because DS maintenance must be coordinated through registrars [34].
- *Rapid Convergence in Most Cases.* Only a small fraction of domains requires a third or fourth iteration, typically when administrators must introduce a new key, retire an old one, or wait out TTLs. Even then, the majority of these multi-iteration scenarios still rely on the same recurring

command set: DS updates, zone signing, and key generation or removal. Once the correct sequence is applied, the zone swiftly transitions to full compliance.

Overall, it illustrates that while DNSSEC debugging can appear daunting, particularly given the diverse error codes, a finite set of well-ordered, repeatable commands is sufficient to address the vast majority of misconfigurations.

9.6.5 Limitations of ZReplicator and DFixer

While ZReplicator and DFixer cover a broad range of DNSSEC errors, few constraints limit their scope:

ZReplicator

- *Some zone-file errors remain unreproducible:* Among the 47 distinct DNSSEC error codes present in our dataset, a few could not be consistently replicated; for example, some negative-proof anomalies or a DNSKEY with an impossible bit length permitted by a buggy nameserver defy replication in our local environment, often because BIND or our signing utilities *refuse* to load blatantly invalid records. However, the proportion of these errors are relatively very low; in our dataset, only 2% snapshots have these errors. Because these errors are often intermingled with other replicable errors, they reduce the overall efficacy of ZReplicator.
- *Only leaf-zone replication is supported:* ZReplicator recreates a target zone and (optionally) its immediate parent, assuming full control of both. However, a domain may contain CNAME or DNAME records pointing to external zones beyond an operator's authority. For example, if `c.example.com` has a CNAME to `example.org`, ZReplicator treats `example.org` as out of scope. This design choice ensures we faithfully replicate *local* errors a zone owner can fix, without attempting to mirror unrelated external dependencies.
- *Algorithm-distribution constraints:* We attempt to preserve the DNSSEC algorithm usage from each original snapshot. In practice, some algorithms (e.g., DSA-NSEC3-SHA1) are no longer supported by BIND. When encountering such algorithms, we substitute them with available alternatives (e.g., RSASHA256, ECDSAP256SHA256), provided they are not already in use. A small fraction of zones, however, exhaust all supported algorithms, making exact replication or even generation impossible.

DFixer

- *Requires manual update of DS records:* In our evaluation of DFixer, we were able to automate everything including interaction with the parent zone by setting up a base zone under our control. In real-world deployments, however, such interactions would need to be performed manually. Although CDNSKEY and CDS [55] records could have been used for this purpose, their adoption

remains very limited, so we chose not to rely on them. Additionally, some registrars use different formats for submitting DS records; for instance, AWS Route 53 requires uploading the public key instead. These edge cases are not covered in our instructions, as we follow the common patterns used by most registrars.

- *Optimality not guaranteed:* DFixer’s command order is derived from a manually crafted dependency graph. While this typically yields few-iteration fixes, it is *not guaranteed to be globally minimal in every scenario*.
- *Unreplicated errors and the evaluation gap:* ZReplicator cannot reproduce a small set of anomalies (e.g., negative proofs and buggy-nameserver artifacts), so we could not empirically test DFixer on those cases. Nevertheless, our DFixer implementation addresses all error codes currently defined in DNSViz. For instance, all the unreproducible negative proof anomalies can be resolved through a straightforward zone re-signing while a DNSKEY with an invalid bit length can be handled by generating a valid key, removing the invalid one, and re-signing the zone. Based on this coverage, we argue that DFixer demonstrates a high degree of generalizability. However, we acknowledge the need for further empirical validation, which we plan to pursue by integrating DFixer with live DNSViz in future work.

Overall, these limitations reflect our primary goal of assisting domain owners in fixing *zone-level* DNSSEC errors. They do not diminish DFixer’s utility for the majority of commonly observed misconfigurations, nor do they undercut ZReplicator’s ability to reproduce most real-world scenarios where the zone file itself is at fault.

9.6.6 Testing Extensibility

Although our primary pipeline focuses on generating BIND commands, we also explore DFixer’s applicability to other authoritative server software. DNSSEC operations—such as key generation, zone signing, and DS management—share common principles across implementations, suggesting that a direct mapping from BIND-centric commands to other server CLIs is often feasible. Below, we discuss how our manually curated commands can be adapted to NSD and PowerDNS.

NSD: NSD is a widely used, high-performance authoritative DNS server [6]. To validate our approach, we manually map the BIND commands produced by DFixer to equivalent NSD workflows using the `ldns` utilities (e.g., `ldns-keygen`, `ldns-signzone`). We replicate each DNSSEC error code on two NSD instances (v4.8.0) and confirm that *every replicated error could be resolved through these mapped commands*. This outcome demonstrates that our error-to-command logic can be extended to NSD with minimal effort.

PowerDNS: PowerDNS is another open-source authoritative server supporting multiple backends (e.g., PostgreSQL, MySQL, BIND files) and various DNSSEC modes (live-signing, pre-signed). Its

live-signing mode generates signatures on the fly, which thwarted our attempts to inject controlled misconfigurations. We therefore used the BIND backend in pre-signed mode, replicating all misconfigurations except involving NSEC or NSEC3 records due to a known PowerDNS bug [142]. We also built a one-to-one mapping from BIND to `pdnsutil`, but PowerDNS does not permit fixing errors in a pre-signed zone via `pdnsutil`. Therefore, we could not validate the effectiveness of our mapped commands in PowerDNS. As a workaround, we used the BIND commands generated by DFixer to fix the zone locally, then imported the repaired zone back into PowerDNS. Using this mechanism, we confirmed that DFixer was also able to successfully repair PowerDNS zones as well.

Knot DNS: Knot DNS is another open-source high-performance authoritative DNS server [4]. To evaluate the potential applicability of DFixer to Knot, we investigated the feasibility of mapping our existing BIND commands and parameters set to Knot DNS environment. Our analysis indicates that this mapping can be achieved without significant difficulty. Specifically, operations such as key generation with configurable algorithm and size, as well as key retirement and removal, are supported through the `keymgr` utility. Furthermore, parameters related to NSEC/NSEC3 handling and signature lifetime management can be configured within the `policy` section of the Knot DNS configuration.

Takeaway: Any authoritative software that exposes fundamental operations such as zone signing, key generation, key activation or deactivation with basic parameter customization can host DFixer's repair plan with a thin translation layer.

Chapter 10

Concluding Discussion

This dissertation has explored the security, reliability, and operational realities of email and DNS, the twin pillars underpinning modern Internet communication. Through large-scale empirical analyses and system-level investigations, it has demonstrated that while the community has made significant strides in extending and securing legacy protocols, the Internet's trust fabric remains deeply fragile due to persistent misconfigurations, design assumptions that no longer hold, and the complex interdependencies between layers of the protocol stack.

Our first two studies dissected the landscape of email sender authentication through SPF and DMARC, uncovering widespread inconsistencies in their deployment and evaluation. Drawing from over 176 million domains, we showed that operational errors not only degrade deliverability but also create unforeseen attack surfaces, including exploitable DNS amplification channels within SPF evaluation processes. These findings underscore that even mature, widely adopted mechanisms can expose the ecosystem to new threats when deployed at scale without rigorous operational validation.

Building upon this, our third study investigated email transport security in the context of the SMTP MTA Strict Transport Security (MTA-STS) protocol. Despite being grounded in the well-established web PKI model, MTA-STS suffers from substantial adoption barriers and configuration pitfalls; 28% of active deployments in our dataset were found to be misconfigured. These misconfigurations effectively nullify the intended security guarantees, highlighting the challenges of operationalizing secure transport in a decentralized, policy-driven infrastructure such as email.

Recognizing that these email security mechanisms are fundamentally dependent on DNS, the final component of this work examined DNS security itself. Through a longitudinal study of over one million DNSSEC diagnostic traces, we revealed that low adoption and persistent errors continue to erode DNSSEC's practical benefits. To address this, we introduced DFixer, an automated remediation system that diagnoses cascaded DNSSEC errors, maps them to root causes, and generates concrete repair actions. Our evaluation demonstrated that DFixer can automatically correct 99.99% of observed misconfigurations within seconds, showing that practical automation can substantially improve DNS reliability and, by extension, the security of dependent protocols.

Taken together, these four lines of inquiry reveal a consistent pattern: security protocols for Internet infrastructure often fail not because of cryptographic weakness, but because of systemic operational fragility. Protocols that rely on DNS as a control and trust plane inherit its weaknesses; conversely, improvements at the DNS layer propagate upward to strengthen higher-level services

such as email. This observation suggests that enhancing Internet security demands a holistic, ecosystem-wide approach that bridges protocol design, deployment tooling, and operational best practices.

Looking ahead, several promising directions emerge. First, extending automated diagnostic and repair systems like DFixer to cover SPF, DMARC, and MTA-STS could provide self-healing capabilities across the email security stack. Second, cross-layer monitoring frameworks that correlate DNS, TLS, and SMTP behaviors could offer early warning of misconfigurations or downgrade attacks in real time. Finally, greater emphasis on usability and deployment support, including standardized configuration guidance and automated validation at registration or hosting providers, could significantly reduce the rate of systemic errors.

Beyond the specifics of SPF, DMARC, MTA-STS, and DNSSEC, this dissertation highlights several general principles for secure email and DNS infrastructure. First, protocol designs must account for operational reality: mechanisms that are fragile, complex, or under-documented tend to be misconfigured at Internet scale, weakening their intended guarantees. Second, secure-by-default and easy-to-validate configurations are critical, as inconsistent implementations and ambiguous semantics create exploitable gaps between specification and practice. Third, defenses must be robust to partial deployment and legacy systems, since email and DNS are inherently heterogeneous and long-lived ecosystems. Fourth, effective security requires observability and feedback: operators need clear diagnostics, measurements, and reporting channels to understand and correct issues. Finally, automation plays a key role in achieving resilience at scale, as demonstrated by our automated tools for diagnosing and repairing DNSSEC errors. Together, these principles frame our empirical studies not just as protocol-specific analyses, but as case studies in how to design, deploy, and maintain secure Internet infrastructure.

Bibliography

- [1] BIND9. <https://www.isc.org/bind/>.
- [2] DNSSEC SCOREBOARD. https://www.verisign.com/en_US/company-information/verisign-labs/internet-security-tools/dnssec-scoreboard/index.xhtml.
- [3] Internet Society. <https://www.internetsociety.org/>.
- [4] KnotDNS. <https://www.knot-dns.cz/>.
- [5] Mail Operators' List. <https://www.mailop.org/>.
- [6] NSD. <https://www.nlnetlabs.nl/projects/nsd/about/>.
- [7] Rspamd. <https://rspamd.com/>.
- [8] Shell-AI: let AI write your shell commands. <https://github.com/ricklamers/shell-ai>.
- [9] D. E. 3rd. Transport Layer Security (TLS) Extensions: Extension Definitions. RFC 6066, IETF, 2011.
- [10] K. Andersen, B. Long, S. Blank, and M. Kucherawy. The Authenticated Received Chain (ARC) Protocol. IETF, 2019.
- [11] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. RFC 4033, IETF, 2005. <http://www.ietf.org/rfc/rfc4033.txt>.
- [12] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Protocol Modifications for the DNS Security Extensions. RFC 4035, IETF, 2005. <http://www.ietf.org/rfc/rfc4035.txt>.
- [13] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Resource Records for the DNS Security Extensions. RFC 4034, IETF, 2005. <http://www.ietf.org/rfc/rfc4034.txt>.
- [14] M. I. Ashiq, T. Fiebig, and T. Chung. Unraveling the Complexities of MTA-STS Deployment and Management in Email. *IMC*, 2025.
- [15] M. I. Ashiq, O. Hureau, C. Deccio, and T. Chung. Decoding DNSSEC Errors at Scale: An Automated DNSSEC Error Resolution Framework using Insights from DNSViz Logs. *IMC*, 2025.

- [16] M. I. Ashiq, W. Li, T. Fiebig, and T. Chung. You've Got Report: Measurement and Security Implications of DMARC Reporting. *USENIX Security*, 2023.
- [17] M. I. Ashiq, W. Li, T. Fiebig, and T. Chung. SPF Beyond the Standard: Management and Operational Challenges in Practice and Practical Recommendations. *USENIX Security*, 2024.
- [18] Add your SPF record at your domain provider. <https://support.google.com/a/answer/10684623?hl=en>.
- [19] Anonymized for reviews. Anonymized for reviews.
- [20] Anonymized for reviews. Anonymized for reviews.
- [21] Apache SpamAssassin Project. <https://spamassassin.apache.org/>.
- [22] B. Blechschmidt and B. Stock. Extended Hell(o): A Comprehensive Large-Scale Study on Email Confidentiality and Integrity Mechanisms in the Wild. *USENIX Security*, 2023.
- [23] M. Brown. Calling time on DNSSEC: The costs exceed the benefits. 2023. <https://www.mattb.nz/w/2023/06/02/calling-time-on-dnssec/>.
- [24] N. Bennett, R. Sowards, and C. D. SPFail: Discovering, Measuring, and Remediating Vulnerabilities in Email Sender Validation. *IMC*, 2022.
- [25] R. Barnes, J. Hoffman-Andrews, D. McCarney, and J. Kasten. Automatic Certificate Management Environment (ACME). RFC 8555, IETF, 2019.
- [26] S. Blank, P. Goldsten, T. Loder, T. Zinkn, and M. Bradshaw. Brand Indicators for Message Identification (BIMI). IETF, 2021.
- [27] D. Crocker, T. Hansen, and M. Kucherawy. DomainKeys Identified Mail (DKIM) Signatures. RFC 6376, IETF, 2011. <http://www.ietf.org/rfc/rfc6376.txt>.
- [28] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, IETF, 2008. <http://www.ietf.org/rfc/rfc5280.txt>.
- [29] F. Cangialosi, T. Chung, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson. Measurement and Analysis of Private Key Sharing in the HTTPS Ecosystem. *CCS*, 2016.
- [30] J. Chen, V. Paxson, and J. Jiang. Composition kills: a case study of email sender authentication. *USENIX Security*, 2020.
- [31] M. Crispin. INTERNET MESSAGE ACCESS PROTOCOL. IETF RFC 3501, IEFT, 2003.
- [32] S. Czybik, M. Horlboge, and K. Rieck. Lazy Gatekeepers: A Large-Scale Study on SPF Configuration in the Wild. *IMC*, 2023.

- [33] T. Chung, R. van Rijswijk-Deij, B. Chandrasekaran, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson. A Longitudinal, End-to-End View of the DNSSEC Ecosystem. *USENIX Security*, 2017.
- [34] T. Chung, R. van Rijswijk-Deij, D. Choffnes, A. Mislove, C. Wilson, D. Levin, and B. M. Maggs. Understanding the Role of Registrars in DNSSEC Deployment. *IMC*, 2017.
- [35] Check a DANE TLS Service. <https://www.huque.com/bin/danecheck>.
- [36] Cloudflare. <http://www.cloudflare.com>.
- [37] C. Deccio, J. Sedayao, K. Kant, and P. Mohapatra. Quantifying and Improving DNSSEC Availability. *2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)*, 2011.
- [38] C. Deccio, T. Yadav, N. Bennett, A. Hilton, M. Howe, T. Norton, J. Rohde, E. Tan, and B. Taylor. Measuring Email Sender Validation in the Wild. *CoNEXT*, 2021.
- [39] C. Dietrich, K. Krombholz, K. Borgolte, and T. Fiebig. Investigating system operators' perspective on security misconfigurations. *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018.
- [40] D. Dittrich and E. Kenneally. The Menlo Report: Ethical Principles Guiding Information and Communication Technology Research. 2012. https://www.dhs.gov/sites/default/files/publications/CSD-MenloPrinciplesCORE-20120803_1.pdf.
- [41] V. Dukhovni and W. Hardaker. The DNS-Based Authentication of Named Entities (DANE) Protocol: Updates and Operational Guidance. RFC 7671, IETF, 2015.
- [42] V. Dukhovni. DANE in SMTP the sky is not falling. 2020. <http://dnssec-stats.ant.isi.edu/~viktor/test.html>.
- [43] Z. Durumeric, D. Adrian, A. Mirian, J. Kasten, E. Bursztein, N. Lidzborski, K. Thomas, V. Eranti, M. Bailey, and J. A. Halderman. Neither Snow Nor Rain Nor MITM ... An Empirical Analysis of Email Delivery Security. *IMC*, 2015.
- [44] C. A. Daniluk, Y. Nosyk, A. Duda, and M. Korczynski. Zeros Are Heroes: NSEC3 Parameter Settings in the Wild. *IMC*, 2024.
- [45] DANE SMTP Validator. <https://dane.sys4.de/>.
- [46] DANE SMTP Validator by SIDN Labs. <https://check.sidnlabs.nl/dane/>.
- [47] DMARC Data Providers. <https://dmarcian.com/dmarc-data-providers/>.
- [48] DNSSEC Debugger. <http://dnssec-debugger.verisignlabs.com>.
- [49] DNSSEC Deployment Report. <https://rick.eng.br/dnssecstat/>.

- [50] DNSSEC Statistics. <https://www.internetsociety.org/deploy360/dnssec/statistics/>.
- [51] DNSSEC chain validation issue: technical incident report. <https://internetnz.nz/news-and-articles/dnssec-chain-validation-issue-technical-incident-report/>.
- [52] DNSSEC deployment in Sweden. <https://archive.icann.org/meetings/london2014/en/schedule/wed-dnssec/presentation-dnssec-deployment-sweden-25jun14-en.pdf>.
- [53] DNSSEC issues take Fiji domains offline. <https://blog.cloudflare.com/dnssec-issues-fiji/>.
- [54] DNSViz. <http://dnsviz.net>.
- [55] A. Eden. Announcing CDS/CDNSKEY Support. 2019. https://blog.dnssimple.com/2019/02/cds_cdnskey/.
- [56] D. Eastlake. Domain Name System Security Extensions. IETF RFC 2535, IETF, 1999.
- [57] Enhancing mail flow with MTA-STS. <https://learn.microsoft.com/en-us/purview/enhancing-mail-flow-with-mta-sts>.
- [58] I. 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. CCS, 2015.
- [59] <https://access.redhat.com/solutions/396273>. <https://access.redhat.com/solutions/396273>.
- [60] M. T. Goodrich, R. Tamassia, and D. Yao. Accredited DomainKeys: A Service Architecture for Improved Email Validation. *CEAS 2005 - Second Conference on Email and Anti-Spam, July 21-22, 2005, Stanford University, California, USA*, 2005.
- [61] F. Holzbauer, J. Ullrich, M. Lindorfer, and T. Fiebig. Not that Simple: Email Delivery in the 21st Century. *USENIX ATC*, 2022.
- [62] G. Huston. Measuring the use of DNSSEC. 2023. <https://blog.apnic.net/2023/09/18/measuring-the-use-of-dnssec/>.
- [63] G. Huston. Calling time on DNSSEC? 2024. <https://blog.apnic.net/2024/05/28/calling-time-on-dnssec/>.
- [64] H. Hu and G. Wang. End-to-End Measurements of Email Spoofing Attacks. *USENIX Security*, 2018.
- [65] H. Hu, P. Peng, and G. Wang. Towards Understanding the Adoption of Anti-Spoofing Protocols in Email Systems. *IEEE Cybersecurity Development (SecDev)*, 2018.

- [66] P. Hoffman. SMTP Service Extension for Secure SMTP over Transport Layer Security. IETF RFC 3207, IEFT, 2002.
- [67] P. Hoffman and J. Schlyter. The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. RFC 6698, IETF, 2012.
- [68] R. Holz, J. Amann, O. Mehani, M. Wachs, and M. A. Kaafar. TLS in the wild: an Internet-wide analysis of TLS-based protocols for electronic communication. *NDSS*, 2015.
- [69] W. Hardaker and V. Dukhovni. Guidance for NSEC3 Parameter Settings. RFC 9276, 9276, RFC Editor, 2022.
- [70] How to Set Up SPF for QQ Mail? <https://www.tencentcloud.com/document/product/1097/44886>.
- [71] Increase email security with MTA-STS and TLS reporting About MTA-STS and TLS reporting. <https://support.google.com/a/answer/9261504?hl=en>.
- [72] P. Jeitner and H. Shulman. Injection Attacks Reloaded: Tunnelling Malicious Payloads over DNS. *USENIX Security*, 2021.
- [73] J. Klensin. Simple Mail Transfer Protocol. RFC 5321, IETF, 2008. <http://www.ietf.org/rfc/rfc5321.txt>.
- [74] M. Kucherawy and E. Zwicky. Domain-based Message Authentication, Reporting, and Conformance (DMARC). RFC 7489, IETF, 2015. <https://tools.ietf.org/html/rfc7489>.
- [75] P. Krumviede, R. Catoe, and D. J. C. Klensin. IMAP/POP AUTHorize Extension for Simple Challenge/Response. RFC 2195, 2195, RFC Editor, 1997.
- [76] S. Kitterman. Sender Policy Framework (SPF) for Authorizing Use of Domains in Email. RFC 7208, IETF, 2014. <https://tools.ietf.org/html/rfc7208>.
- [77] W. Kumari, E. Hunt, R. Arends, W. Hardaker, and D. C. Lawrence. Extended DNS Errors. RFC 8914, RFC Editor, 2020.
- [78] Key Management, DNSSEC Guide : Chapter 6. Advanced Discussions. <https://dnsinstitute.com/documentation/dnssec-guide/ch06s04.html>.
- [79] B. Laurie, G. Sisson, R. Arends, and D. Blacka. DNS Security (DNSSEC) Hashed Authenticated Denial of Existence. RFC 5155, IETF, 2008.
- [80] E. Lewis. Where did DNSSEC go wrong? <https://blog.apnic.net/2024/07/05/where-did-dnssec-go-wrong/>.
- [81] H. Lee, M. I. Ashiq, M. Muller, R. van Rijswijk-Deij, T. Kwon, and T. Chung. Under the Hood of DANE Mismanagement in SMTP. *USENIX Security*, 2022.

- [82] H. Lee, A. Girish, R. van Rijswijk-Deij, T. T. Kwon, and T. Chung. A Longitudinal and Comprehensive Study of the DANE Ecosystem in Email. *USENIX Security*, 2020.
- [83] S. Liu, I. Foster, S. Savage, G. M. Voelker, and L. K. Saul. Who is .com? Learning to Parse WHOIS Records. *IMC*, 2015.
- [84] Levenshtein distance. <https://xlinux.nist.gov/dads/HTML/Levenshtein.html>.
- [85] D. Margolis, M. Risher, B. Ramakrishnan, A. Brotman, and a. J. Jones. SMTP MTA Strict Transport Security (MTA-STS). RFC 8461, IETF, 2018.
- [86] D. Margolis, A. Brotman, B. Ramakrishnan, J. Jones, and M. Risher. SMTP TLS Reporting. RFC 8460, 8460, RFC Editor, 2018.
- [87] J. Myers and M. Rose. Post Office Protocol - Version 3. RFC 1939, IETF, 1996.
- [88] M. Müller, M. Thomas, D. Wessels, W. Hardaker, T. Chung, W. Toorop, and R. van Rijswijk-Deij. Roll, Roll, Roll your Root: A Comprehensive Analysis of the First Ever DNSSEC Root KSK Rollover. *IMC*, 2019.
- [89] M. Müller, W. Toorop, T. Chung, J. Janssen, and R. van Rijswijk-Deij. The Reality of Algorithm Agility: Studying the DNSSEC Algorithm Life-Cycle. *IMC*, 2020.
- [90] T. Mori, K. Sato, Y. Takahashi, and K. Ishibashi. How is E-Mail Sender Authentication Used and Misused? *Proceedings of the 8th Annual Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference*, Association for Computing Machinery, 2011.
- [91] MTA-STS Overrides DANE. <https://github.com/Snawoot/postfix-mta-sts-resolver/issues/67>.
- [92] Mail.com MTA-STS Policy. <https://mta-sts.mail.com/.well-known/mta-sts.txt>.
- [93] Mail::SPF::Query. <https://metacpan.org/pod/Mail::SPF::Query>.
- [94] Modern Email Security Standards for EU (MESSEU). messeu@sys4.de.
- [95] Y. Nosyk, M. Korczyk, and A. Duda. Extended DNS Errors: Unlocking the Full Potential of DNS Troubleshooting. *IMC*, 2023.
- [96] North American Network Operators' Group. <https://www.nanog.org/>.
- [97] D. Poddebniak, F. Ising, H. Böck, and S. Schinzel. Why TLS is better without STARTTLS: A Security Analysis of STARTTLS in the Email Context. *USENIX Security*, 2021.
- [98] V. Pathak, L. Iftode, and D. Yao. Improving Email Trustworthiness through Social-Group Key Authentication. *CEAS 2008 - The Fifth Conference on Email and Anti-Spam, 21-22 August 2008, Mountain View, California, USA*, 2008.

- [99] V. L. Pochat, T. V. Gothem, S. Tajalizadehkhoob, M. Korczyński, and W. Joosen. TRANCO: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. *NDSS*, 2019.
- [100] Porkbun LLC. <https://porkbun.com/>.
- [101] Postfix. <http://www.postfix.org/>.
- [102] Postfix Configuration Parameters. <https://www.postfix.org/postconf.5.html>.
- [103] Potential Risks with MTA-STS Usage? <https://support.dmarcreport.com/support/solutions/articles/5000885320-potential-risks-with-mta-sts-usage->.
- [104] E. Redmiles, Y. Acar, S. Fahl, and M. Mazurek. A summary of survey methodology best practices for security and privacy researchers. 2017. <https://drum.lib.umd.edu/items/683d78b0-a0e3-4fae-9c93-b75aae4ad11b>.
- [105] P. Resnick. Internet Message Format. RFC 5322, IETF, 2008. <https://www.rfc-editor.org/info/rfc5322>.
- [106] R. Roberts, Y. Goldschlag, R. Walter, T. Chung, A. Mislove, and D. Levin. You Are Who You Appear to Be: A Longitudinal Study of Domain Impersonation in TLS Certificates. *CCS*, 2019.
- [107] S. Roth, R. van Rijswijk-Deij, and T. Chung. Tracking Registrar Support for DNSSEC: It’s Slowly Getting Better. *IMC*, 2019.
- [108] F. Y. Rashid. Time is running out for NTP. <https://www.infoworld.com/article/3144546/time-is-running-out-for-ntp.html>.
- [109] B. Stock, G. Pellegrino, F. Li, M. Backes, and C. Rossow. Didnt You Hear Me? Towards More Successful Web Vulnerability Notifications. *NDSS*, 2018.
- [110] 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. *USENIX Security*, 2021.
- [111] M. Salganik. Bit by Bit: Social Research for the Digital Age. 2016.
- [112] S. Scheffler, S. Smith, Y. Gilad, and S. Goldberg. The Unintended Consequences of Email Spam Prevention. *PAM*, 2018.
- [113] SIDN Labs DANE Deployment Statistics. <https://stats.sidnlabs.nl/en/mail.html#dane>.
- [114] SMTP DANE TLS Adoption Survey. <https://stats.dnssec-tools.org/explore/>.

- [115] Sabotage! Coping with the Joe Job. <https://www.sitepoint.com/sabotage-coping-joe-job/>.
- [116] D. Tatang, F. Zettl, and T. Holz. A First Large-Scale Analysis on Usage of MTA-STTS. *DIMVA*, 2021.
- [117] D. Tatang, F. Zettl, and T. Holz. The Evolution of DNS-Based Email Authentication: Measuring Adoption and Finding Flaws. *RAID*, 2021.
- [118] TLD Distribution. <https://domainnamestat.com/statistics/tld/others>.
- [119] The Case of the Recursive Resolvers: What Happened During Slacks DNSSEC Rollout. <https://slack.engineering/what-happened-during-slacks-dnssec-rollout/>.
- [120] The DNS Operations, Analysis, and Research Center (DNS-OARC). <https://www.dns-oarc.net/oarc/programme>.
- [121] The current state of SMTP STARTTLS deployment. <https://www.facebook.com/notes/protect-the-graph/the-current-state-of-smtp-starttls-deployment/1453015901605223/>.
- [122] C. Wang and G. Wang. Revisiting Email Forwarding Security under the Authenticated Received Chain Protocol. *WWW*, 2022.
- [123] 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. *USENIX Security*, 2022.
- [124] M. Wong and W. Schlitt. Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1. RFC 4408, IETF, 2006.
- [125] P. Wouters, O. Sury, and I. S. Consortium. Algorithm Implementation Requirements and Usage Guidance for DNSSEC. IETF, 2019.
- [126] S. Weiler and D. Blacka. Clarifications and Implementation Notes for DNS Security (DNSSEC). RFC 6840, IETF Request for Comments 6840, IETF, Feb. 2013.
- [127] What is SPF Flattening? <https://acme.com/software/spfmilter/>.
- [128] What is greylisting and how does it work? <https://www.mail.com/blog/posts/what-is-greylisting/33/>.
- [129] M. Yajima, D. Chiba, Y. Yoneya, and a. T. Mori. A First Look at Brand Indicators for Message Identification (BIMI). *Proceedings of the 24th Passive and Active Measurement (PAM)*, 2023.
- [130] Yahoo! Mail MTA-STTS. <https://mta-sts.yahoo.com/.well-known/mta-sts.txt>.

- [131] L. Zhu, D. Wessels, A. Mankin, and J. Heidemann. Measuring DANE TLSA Deployment. *TMA*, 2015.
- [132] Zonemaster. <https://zonemaster.net/en/run-test>.
- [133] amavisd-new. <https://www.ijs.si/software/amavisd/>.
- [134] dmarcian. <https://dmarcian.com>.
- [135] iRedMail - Open Source Mail Server Solution. <https://www.iredmail.org/>.
- [136] libspf2. <https://www.libspf2.org/>.
- [137] milter-greylis. <http://hcpnet.free.fr/milter-greylis/>.
- [138] mtpolicyd. <https://mtpolicyd.org/index.html>.
- [139] policyd-spf. <https://manpages.debian.org/testing/postfix-policyd-spf-python/policyd-spf.1.en.html>.
- [140] pyspf. <https://github.com/sdgathman/pyspf/tree/master>.
- [141] spfmilter. <https://acme.com/software/spfmilter/>.
- [142] zone2sql and load-zone are not suitable for presigned zones; set-presigned is a useless button. <https://github.com/PowerDNS/pdns/issues/8892>.

Appendices

Appendix A

Copyright Statement

This dissertation is based on our previously published work in the USENIX Security Symposium 2023 [16], 2024 [17], and ACM Internet Measurement Conference 2025 [14, 15]. I have permission from my co-authors and publisher to use the work for my thesis.

Appendix B

Ethical Considerations

B.1 Data Collection

For this dissertation, we scanned DNS records, MTA-STS policy files, and STARTTLS certificate for MX hosts. All of these scans were done at low rate limits to ensure that we do not adversely affect any users ability to obtain these data in a timely manner. For DNS scans, we opted to use public resolvers to avoid the risk of overloading small recursive resolvers. None of the measurement data we obtained throughout our study involved human subjects or any personally identifiable information.

B.2 Survey with Email Operators

Both of our surveys focus on organizations rather than individuals and collects factual data on system deployments. Our Institutional Review Board (IRB) confirmed that they do not qualify as human subject research, eliminating the need for studying protocol evaluations. Nonetheless, we adhered to best practices similar to those for human subject research, informing participants of their data access rights and the ability to withdraw at any time.

B.3 SMTP Server Scanning Experiment

Our approach does not involve sending unsolicited emails; however, we intend to discuss the potential implications for email service providers explicitly before sharing our findings. Firstly, we correctly set up reverse DNS record and remove duplicates from the MX records and their resolved IP addresses to ensure that we communicate with no more than one server.

We also took extra care to ensure our SMTP scanner is fully compliant with the RFC standards; for instance, we have programmed our scanner to disconnect following the QUIT command, mirroring the behavior of a well-behaved SMTP client. This attention to detail is crucial; failing to comply

with this standard, such as by disconnecting without the proper command, could unnecessarily alert system administrators. It is worth noting that throughout our scanning period, we did not receive any automated abuse reports or personalized feedback from operators.

We acknowledge that we have not asked for any explicit consent to connect to the IP addresses mapped to MX records, which may violate one of the four principles of the Menlo Report [40], “Respect for Persons”; unfortunately, it is practically impossible for us to obtain an informed consent from all email operators and also, emailing them to ask for approval will be considered as unsolicited email. Thus, we decided not to ask for consent. However, it is important to note that the inability to obtain informed consent does not imply a disregard for respect towards individuals [111]. To ensure we follow the guidelines by the Menlo report [40], we did not send any emails and tried our best not to contact multiple email operators by deduplicating the addresses. We believe in the value of publishing our findings, considering the potential benefits and the preventative aspect of our research in mitigating future harm by informing the community, thereby justifying our approach amidst ethical concerns.

B.4 Disclosure Emails for MTA-STS Misconfigurations

We initially did not notify misconfigured domain owners, as such notifications can result in administrators becoming overwhelmed with alerts, often treating them as spam [109]. However, based on feedback from the scientific community, we decided to notify the affected domains via email. We also included a simple feedback mechanism in our email to understand whether our initial conjecture is valid or not; Of 497 feedback responses, 341 considered our message to be helpful. We also received 45 acknowledgments thanking us for the notification.

B.5 ZReplicator and DNSViz

Our dataset only contains public DNS information for a zone and does not have any personally identifiable information. For ZReplicator, we configured our replicated zones inside a local containerized environment, thus not facing public DNS servers. Additionally, during our experimentation with probe utility, we ensured to not query root and TLD servers by leveraging the `-x` and `-a` flag in DNSViz.

Appendix C

SMTP TLS Reporting

SMTP TLS reporting [86] is a mechanism that allows the senders to inform the recipient MTA of any problems with its TLS negotiations or MTA-STS/DANE policy validation. This enables administrators to address misconfigurations and potential security vulnerabilities. When a sending MTA encounters issues related to Transport Layer Security (TLS) while attempting to deliver emails to a domain that has a TLS Reporting (TLSRPT) policy, it compiles a report detailing these issues and sends it to the designated reporting address specified by the receiving domain. A domain's TLSRPT policy is found at the TXT record of `_smtp._tls` subdomain which is referred to as TLS-RPT record. Along with MTA-STS, MX, NS, and A, we also collected TLS-RPT record of all the SLDs in our dataset from [section 6.1](#).

While unfortunately not many domains currently support report sending¹ (only 20 in [subsection 6.4.1](#)), number of domains that support TLS-RPT records are far greater as shown in [Figure C.1](#); this implies domain owners are open to receive TLS reports to debug their potential misconfigurations. Initial adoption in 2021-10 ranged from 11,531 (0.02%) for `.com` to 1,527 (0.03%) domains for `.org`. As of 2024-09, we find adoption to have risen 3-4 times, with adoption ranging between 52,641 (0.07%) domains for `.com` and 7,192 (0.12%) domains for `.org`.

Although the adoption rate is still relatively low, we can see a high percentage of domains supporting MTA-STS has SMTP TLS reporting enabled. In our disclosure emails ([subsection 6.2.7](#)), we recommended domain owners to adopt TLS Reporting if they have not already enabled it.

¹Only 2 major providers send TLS reports currently: Google and Microsoft

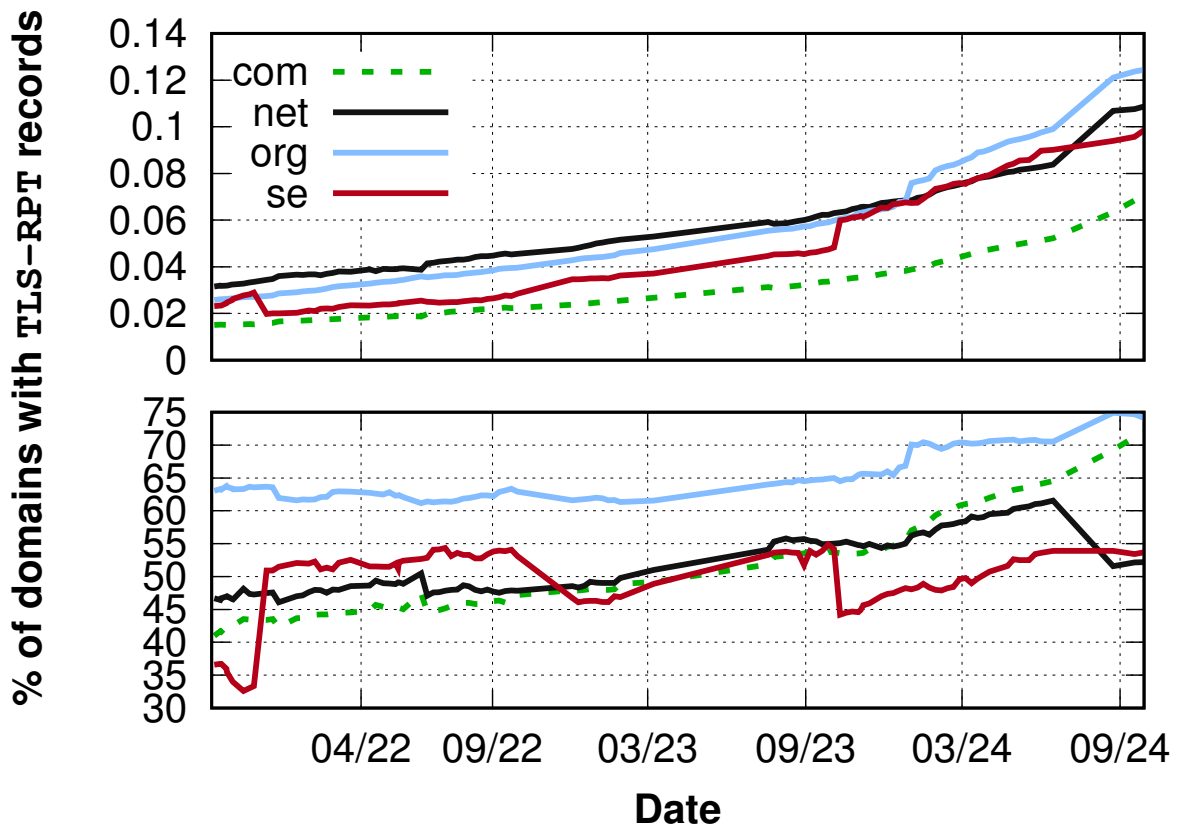


Figure C.1: The deployment of TLS-RPT records; top graph shows the % of domains with MX records having TLS-RPT records; bottom graph shows the % of domains with MTA-STS records having TLS-RPT records. At December 21, 82 .se domains revoked their TLS-RPT records causing the downspike in the top graph. Additionally, from Jun' 24 to Aug '24, 1,411 .net domains added TLS-RPT records; only 198 of them had MTA-STS records which explains the downward spike around that time in the bottom graph.

Appendix D

Email Sender Authentication Survey

Our study centers on organizations rather than individuals, gathering data on system deployments. The Institutional Review Board (IRB) has reviewed our methodology and determined that it does not constitute human subject research, thereby exempting it from the protocols typically required for such studies. Despite this exemption, we maintained a commitment to best practices comparable to those used in human subject research, ensuring that participants were informed about their rights regarding data access and the option to withdraw at any time.

All the questions except in Page 1 were optional. In questions where we had Other (please specify) as the last option, a textbox was there for the participants to specify their answer. SCQ denotes single choice question, MCQ denotes multiple choice question, and YN denotes yes no question.

D.1 Questionnaire

Page 1: Consent Form

Participants were presented with the following two mandatory consent questions:

- I consent voluntarily to be a participant in this study and understand that I can refuse to answer questions and I can withdraw from the study at any time, without having to give a reason.
- I understand that information I provide will be used for scientific reports and publications.

If the participant answered No to any of the above questions, the survey would end with no further input.

Page 2: Email Service

Both the questions came with a textbox.

- Would you be willing to provide the name of the organization whose e-mail service you manage? (If you don't want to, please enter No.)

- Would you mind sharing the name of the domain whose e-mail service you manage? We'd like the main domain name, not the domain name of your mail server. For example, example.com, not mx.example.com. (If you don't want to give the name, please enter No.)

Page 3: SMTP Service

SCQ What is the name of your email server software? Or, what is the name of your MTA? Options were Postfix, Exim, Sendmail, Qmail, Exchange, Haraka, MDAemon, hMailServer, and Other (please specify).

SCQ Are you using any framework, setup tool, or how to for your mail server? Options were iRedMail, Mailcow, OpenExchange, Plesk, Webmin, Mail-in-a-box, (ISP) Mail Server Howto, and Other (please specify).

SCQ How many email accounts exist under your operated infrastructure? Options were < 10, 10 - 50, 50 - 100, 100 - 500, > 500, and Other (please specify). This was a single choice question.

SCQ How many emails does your system receive on average per day? Options were < 10, 10 - 100, 101 - 1000, > 1000, and Other (please specify).

SCQ How many concurrent connections can your mail server/ MTA handle on average? Options were < 10, 10 - 100, 101 - 1000, > 1000, and Other (please specify).

SCQ Do you frequently update your MTA or associated plugins and mail filters? By update, we mean following the releases for the associated software and installing them regularly. Options were Always, Often Sometimes, Rarely, and Never.

Date If yes, approximately when did you last update your MTA or any of the plugins and mail filters? This had a datepicker.

MCQ What security email protocols do you use? Options were SPF, Often DKIM, DMARC, BIMI, MTA-STS, DANE, STARTTLS, ARC, and Other (please specify).

Page 4: SPF Validation

YN Does your email server validate the senders SPF record when it receives an email? Options were Yes and No. If the answer to this question is No, survey ends with no further input.

Page 5: SPF Validator

MCQ What is the name of your SPF validator? In other words, which plugin (i.e. milter, policy server, content filter) is doing the SPF validation when an email comes to your server? If

your MTA natively supports SPF validation, please mark “MTA Native”. Options were SpamAssassin, policyd-spf, iRedAPD, libspf/libspf2, RSpamd, MTA Native, Don’t Know, and Other (please specify).

YN Did you set up the SPF validator yourself? If the answer to this question is No, survey ends with no further input.

Page 6: Self-managed SPF Validator

MCQ When does your SPF validator perform the SPF check of the sender domain? Options were After the HELO/EHLO command, After the MAIL command, After the RCPT command, After the DATA command, and After the email is received.

YN Did you change any of the default values in your SPF validator? (Like changing the default DNS timeout of SPF validation). If the answer to this question is No, survey ends with no further input.

Page 7: SPF Validator Default Value Change

- Did you change any of the following default values in your SPF validator? If yes, please indicate how did you change them. Otherwise, please select the “leave as-is” option. FYI, the RFC recommendation for total DNS lookup and void lookup limits is 10, and 2 respectively. This question had three options: Number of permitted DNS lookups per SPF check before an SPF permanent error is raised, Number of void DNS lookups per SPF check before an SPF permanent error is raised, and DNS timeout per SPF check before an SPF permanent error is raised. Each option had three single choice radio buttons: Increase, Leave as-is, and Decrease.
- Please provide any rationales behind such updates (if you did so). This question had a textbox.

Appendix E

MTA-STS Survey

All the questions except in Page 1 are optional. In questions where we had Other (please specify) as the last option, a textbox was there for the participants to specify their answer. SCQ denotes single choice question, MCQ denotes multiple choice question, and YN denotes yes no question. TB denotes open-ended response questions with a textbox. GS denotes grid-style questions with a matrix where respondents rate multiple items or statements (rows) against a set of consistent response options (columns). LS denotes Likert scale questions.

E.1 Questionnaire

Page 1: Consent Form

Participants are presented with the following two mandatory consent questions:

- I consent voluntarily to be a participant in this study and understand that I can refuse to answer questions and I can withdraw from the study at any time, without having to give a reason.
- I understand that information I provide will be used for scientific reports and publications.

If the participant answered No to any of the above questions, the survey would end with no further input.

Page 2: Basic Info

TB Would you be willing to provide the name of the organization whose e-mail service you manage? (If you don't want to, please enter No.)

TB Would you mind sharing the name of the domain whose e-mail service you manage? We'd like the main domain name, not the domain name of your mail server. For example, `example.com`, not `mx.example.com`. (If you don't want to give the name, please enter No). If you enter the domain name, we may fetch MX/NS/TLSA/MTA-STS records.

SCQ How many email accounts exist under your operated infrastructure? Options are < 10, 10 50, 50 100, 100 500, > 500.

Page 3: MTA-STS check 1

YN Have you heard about MTA-STS (SMTP MTA Strict Transport Security)?

If the participant answered No to this question, the survey would end with no further input.

Page 4: MTA-STS check 2

YN Does your domain support MTA-STS? In other words, do you have MTA-STS TXT record in your domain? If you have multiple domains under your infrastructure, please fill this for your most used domain.

If the participant answered No to this question, the survey would jump to Page 10.

Page 5: Deployment for inbound emails

GS Select the best option for each of the following statements for your most used domain.

Statements are: 1. Have a valid MTA-STS TXT record, 2. Have a valid MTA-STS policy with proper HTTPS setup, 3. DNS MX records are consistent with mx values in HTTPS policy body, 4. Email server (i.e. MX record of the domain) supports STARTTLS, 5. Have a PKIX valid certificate for some of my SMTP servers, 6. Have a PKIX valid certificate for all my SMTP servers.

Column options are: Yes, No, Not sure.

LS Why did you choose to adopt MTA-STS for your domain?

Statements are: 1. Prevents downgrade or interception attack by STARTTLS stripping or DNS poisoning, 2. Dependency on web PKI sounds more trustworthy, 3. Provides optional testing only mode, 4. Other alternative DANE requires DNSSEC and is harder to manage. Columns are scaled from most important to not important. Here the participants also had the option to provide open-ended comments in a textbox.

LS Why do you think operators roll out MTA-STS?

Statements are: 1. Because customers asked us to, 2. Because we are required by regulation (e.g. DNSSEC regulation is the US/NL), 3. Because we wanted to play with it, 4. Because we believe it will make google accept our mails more, 5. Because we always want to be on the pulse of tech-dev.

Columns are scaled from most important to not important.

LS What is the largest bottleneck for MTA-STS deployment in your opinion?

Listed assumptions are: 1. Operational complexity, 2. Better alternative in DANE, 3. Do not need email encryption.

Columns are scaled from most important to not important.

Page 6: Misconfigurations

SCQ Is the MTA-STS setting of your domain valid? Options are yes, no, and don't know.

LS What is the most difficult thing you found in setting up and managing MTA-STS for your domain?

Statements are: 1. Setting up associated DNS records, 2. Configuring HTTPS policy file, 3. Configuring SMTP server with a PKI valid certificate, 4. Managing policy update, 5. Opting out of MTA-STS.

Columns are scaled from most difficult to not difficult. Participants also had the option to provide open-ended comments in a textbox.

LS What do you think is the main reason behind the prevalent invalid MTA-STS configurations?

Row options are: 1. Dependency error between policy and DNS (e.g. mismatch between mx pattern in policy file and DNS MX record), 2. SMTP server error (e.g. lacking PKIX-valid certificate), 3. HTTPS policy server error (e.g. TLS certificate failure), 4. DNS error (e.g. wrongly configured DNS setting).

Columns are scaled from matters most to does not matter.

SCQ While updating your policy, which sequence do you maintain? Options are: 1. Update MTA-STS TXT record first, and then update HTTPS policy body, 2. Update HTTPS policy body first, and then update MTA-STS TXT record, 3. Never updated, Don't Know (my policy management is automated/ outsourced/ i am not sure about the order).

Page 7: Policy Host Management

SCQ How do you manage your MTA-STS policy host? Options are: 1. outsourced to a 3rd-party policy hosting provider, and 2. self-managed.

If the participant selected self-managed in this question, the survey would jump to Page 11.

Page 8: Management 1

SCQ Which 3rd-party policy host service do you use? Options are: Tutanota, URIPorts, Mailhardener, PowerDMARC, EasyDMARC, OnDMARC, DMARCReport, Other (please specify).

LS To what extent do you agree with the following statements regarding hosted MTA-STS services?

Statements are: 1. Using hosted MTA-STS service reduces operational complexity to manage MTA-STS policy, 2. Using hosted MTA-STS service reduces error rate due to misconfigurations in HTTPS policy.

Columns were scaled from strongly agree to strongly disagree.

SCQ How do you manage your incoming SMTP server? Options are: 1. outsourced to a external email hosting provider, and 2. self-managed.

If the participant selected self-managed in this question, the survey would jump to Page 11.

Page 9: Both outsourced

YN Does your email hosting provider manage your MTA-STS policy?

Page 10: MTA-STS not supported

SCQ Why do you NOT deploy MTA-STS for your domain? (Please add options if there are any other reasons.) Options are: 1. I do not understand how it works, 2. I understand how it works, but I don't think I need it, 3. I understand how it works but it is too complicated to deploy and manage, 4. I use DANE, 5. Other (Please specify).

YN Have you ever used MTA-STS?

Page 11: DANE check 1

YN Have you heard about DANE (DNS-based Authentication of Named Entities)?

If the participant selected No in this question, the survey would jump to Page 13.

Page 12: Comparison w/ DANE

GS Does your email server support DANE for inbound emails?

Statements are: 1. My mail server domain (i.e. MX address of the domain) has a valid TLSA record, 2. My mail server supports STARTTLS, 3. My DNS authoritative server and registrar both have support for DNSSEC, 4. My SMTP server has a TLS certificate and TLSA record is consistent with the SMTP server certificate.

Column options are: Yes, No, Don't know.

LS In your opinion, which protocol is better in design for mandating email encryption? MTA-STS or DANE?

Options are: 1. Which protocol is easier to deploy, 2. Which protocol has less requirements, 3. Which protocol is easier to maintain/keep functional, 4. Which protocol has the higher security benefit, 5. Which protocol has the higher general benefit, 6. Which protocol incurs lower total cost.

Column scales are: 1. Definitely MTA-STS, 2. More MTA-STS, 3. Balanced, 4. More DANE, 5. Definitely DANE.

TB Are there any other implementation considerations around MTA-STS and DANE that you would like to share?

Page 13: MTA-STS check 3

YS Does your email server(s) validate MTA-STS for outbound connections? Options are: Yes, No, Don't Know.

If the participant selected No in this question, the survey would end with no further input.

Page 14: Validation

SCQ Which tool do you use to validate MTA-STS for outbound connections? Options are: postfix-mta-sts-resolver, mox, proprietary, other (please specify).

Page 15: Validation Support

LS What do you think is the major bottleneck behind lack of MTA-STS validation support?

Row options are: 1. Lack of incentive from the sending side, 2. Difficulty in policy cache maintenance, 3. Low deployment rate among domains, 4. Lack of awareness of its benefits. Column options scaled from most important to not important.

Appendix F

Sample DFixer Output

[Figure F.1](#) presents a sample remediation workflow suggested by DFixer when the target zone has the REVOKED bit set in its only KSK.

Root Cause: Your DS record is linked to a DNSKEY (key_tag=*k*) with REVOKED flag on.

Remediation Plan with BIND commands: Replace the path variables in angle brackets with values of your own environment. Parameters in braces and variables will be automatically populated by DFixer.

1. Generate a new KSK key pair. Execute: `cd <key_dir> && dnssec-keygen -f KSK -a {algo} -b {key_size} -n ZONE {zone}`. This command should create two new key files inside your key directory; please note the name of the public key file (with .key extension).
2. Generate the DS record from the generated public key file. Replace `public_key_file` with the name of the public key file from previous step and execute: `cd <key_dir> && dnssec-dsfromkey -2 <public_key_file>`. This command should show the contents of your DS record in standard output.
3. Upload the DS record to the parent zone. This must be done manually via your registrar.
4. Remove the DS record linked to the revoked DNSKEY (key_tag=*k*) from the parent zone. This also needs to be done manually via your registrar.
5. Wait at least one full TTL (*tll*) for the removed DS record to expire from the cache of any validator. Nothing to execute; in “auto-apply” mode, DFixer will automatically infer the TTL and wait out this period before executing the next command.
6. After *tll* seconds, delete the DNSKEY (key_tag=*k*, pre-revoked key_tag=*pk*) from your zone. Replace `key_file` with the name of the public key file associated with key_tag=*k*, and execute: `dnssec-settime -D {current_time} <key_dir/key_file>`
7. Re-sign the zone. Execute: `cd <key_dir> && dnssec-signzone -N INCREMENT -3 {salt} -S -o {zone} -t <zone_dir/unsigned_zone_file>`

Figure F.1: Sample remediation workflow produced by APM for a domain with its only KSK revoked and linked to a DS record

Appendix G

Effectiveness of LLM-Based Approach in Resolving DNSSEC Misconfigurations

Although large language models (LLMs) can be helpful for troubleshooting, our experiments show that prompt engineering alone is insufficient for guiding administrators to fully correct DNSSEC misconfigurations. To demonstrate this, we created several invalid zones using ZReplicator, analyzed them with DNSViz, and queried GPT-4o using the formats shown in [Figure G.1](#) and [Figure G.2](#). We present representative results from one particularly problematic test case here.

In this test zone, we introduced two major errors:

1. *Extraneous DS records* using an algorithm for which no DNSKEY exists.
2. *A bad non-existence proof* in the parent zone, designed to test whether the LLM can distinguish child from parent zone issues.

[Figure G.3](#) shows GPT-4o’s analysis response when prompted per [Figure G.1](#). While the LLM correctly identified the main problem, we observed several limitations:

1. It did not specify *which* DS records were invalid.
2. It conflated errors between the child and parent zones.
3. It failed to rank or prioritize the issues.
4. It presented additional warnings, contrary to our request for concise root-cause analysis.

[Figure G.4](#) shows the model’s suggested remediation plan (following [Figure G.2](#)). Unfortunately, the guidance and commands were partially incorrect:

1. It advised “replacing” the DS record, even though valid one was already in place; the correct and minimal fix was just to remove the extraneous DS entries at the parent zone.
2. It recommended re-signing the zone, which was irrelevant since the primary issue lay in the parent’s NSEC3 records.
3. The proposed `signzone` commands lacked essential parameters.

Moreover, when we presented GPT-4o with a screenshot of the DNSViz output ([Figure G.5](#)), the model *hallucinated* specific `key_tags` and algorithms, further undermining its reliability. These

Description
<p>I'm encountering issues during the process of enabling DNSSEC for my domain. I ran a check with <code>dnsviz.net</code> for debugging, which highlighted several configuration errors, but the root cause of these problems is unclear to me. I have attached the debugging analysis data and visualization diagram from DNSViz in JSON and PNG format respectively. I have also copied and pasted the unique error messages from DNSViz.</p> <pre>};error messages; };attachments;</pre>
Question
<p>Given the files and messages from DNSViz, can you assist me in understanding the specific points of failure, and determining the root cause of the misconfigurations? Give me the identified root cause(s) in concise and short bullet points.</p>
Directives
<ul style="list-style-type: none">• Be as specific as possible with variable parameters such as DNSKEY algorithms, <code>key_tags</code> etc. For example, if your root cause is a DNSKEY with REVOKED flag on, please associate the <code>key_tag</code> of the DNSKEY you are referencing.• Do not report any misconfigurations that belong to the ancestor zones.• If there are multiple root causes, list them in order of importance, with the most critical issues appearing first.• For privacy, please do not refer to the domain name directly.

Figure G.1: Prompt format used for root cause identification

observations underscore that while LLMs can offer partial assistance, they often struggle with nuanced DNSSEC errors and require additional domain-specific logic to be truly effective.

<p>Question</p> <p>For each of the identified root cause(s), please provide specific instructions on how to resolve them.</p>
<p>Directives</p> <p>Extract the variable parameters such as DNSKEY algorithms, key_tags, NSEC or NSEC3 usage, DS digest type etc. from the current zone file settings when possible. For example, if I need to resign the zone, specify NSEC if the zone is currently using NSEC and NSEC3 otherwise.</p>

Figure G.2: Prompt formats used for devising remediation plan with commands

Based on the DNSViz diagram, error messages, and extracted zone data, here are the **specific root causes** for DNSSEC validation failures in your zone, written concisely and **with domain name redacted**:

● Root Causes (Concise & Specific)

1. No NSEC3 record covers the queried DS name

- **Error:** “No NSEC3 RR matches the SNAME”
- **Cause:** The NSEC3 record with hash BOP2VBUG5S3T2K7G0SCPORD6JN4MD5D9 does **not cover** the hashed owner name EPU4Q6F5QVO668BNN740G8CHE7UQERV3

2. NSEC3 record uses non-empty salt

- **Salt:** 8d4557157f54153f
- **Warning:** Salt should be **empty** to conform with [RFC 9276 §3.1]

3. DS record in parent zone uses algorithm 16 (Ed25519)

- But no DNSKEY in the child zone uses algorithm 16
- DNSKEYs present: algorithm 7 (RSASHA1-NSEC3-SHA1), key tags 35595, 53136
- **Error:** “No DNSKEY matched the DS algorithm 16”

4. No RRSIG using algorithm 16 over the DNSKEY RRset

- Affects DNSKEYs:
 - alg 7, tag 35595 and alg 7, tag 53136
- **Cause:** The DS expects Ed25519 signatures, but only RSASHA1 keys exist

Figure G.3: Response from GPT-4o when passed the prompt in [Figure G.1](#) for our invalid setup

Certainly! Here's a concise summary of the steps to fix your DNSSEC issues in BIND:

What's Wrong

- DS record in parent uses **alg 16 (Ed25519)** — but your DNSKEYs use **alg 7 (RSASHA1-NSEC3-SHA1)**.
- NSEC3 is used, but:
 - Has **non-empty salt** (violates RFC 9276).
 - Fails to cover the queried DS name properly.

How to Fix It (BIND-specific)

1. **Replace the DS record:**
`dnssec-dsfromkey -2 -f zone.db Kyourzone.+007+53136.key`
 → **Submit this DS to your registrar; delete the old one using alg 16.**
2. **Re-sign the zone with proper NSEC3:**
`dnssec-signzone -A -3 '' -H 0 -N increment -o yourzone yourzone.db`
3. **Deploy:**
`mv yourzone.db.signed /var/named/yourzone.db`
4. **Reload the zone:**
`rndc reload yourzone`

Let me know if you want this as a script.


Figure G.4: Response from GPT-4o when passed the prompt in [Figure G.2](#) for our invalid setup

Absolutely! Based on the DNSViz visualization, I've reviewed the DNSSEC validation chain and here's a detailed breakdown of what's going wrong:

Problems Identified

1. DS Record Mismatch

- The **parent zone** contains a **DS record** with:
 - Algorithm: 13 (ECDSAP256SHA256), Key tag: 40580
- However, your **zone's DNSKEY RRset** does **not include** a key matching this DS record. Instead, your zone is using:
 - DNSKEY with key tag 2482 (algorithm 13) and 40745 (algorithm 13)

 **Result:** DNS resolvers **cannot verify** that your DNSKEYs are trusted, because the parent zone points to a **different key** than what you're using.

2. Invalid Signatures on Zone Records

All of the RRsets under your domain (e.g., A, SOA, NS, TXT, AAAA) are signed using the **untrusted keys** mentioned above. Because the DS mismatch breaks trust in your DNSKEYs, **all RRSIGs fail validation**, and therefore:

- These records are treated as **bogus** by validating resolvers. The AD (Authenticated Data) bit **will not be set** in DNS responses.

3. Extra/Unused DNSKEY (algorithm 0)

- There is also a **DNSKEY using algorithm 0** (with key tag 66427), which is **invalid for use in DNSSEC. Algorithm 0 means "reserved", and it's not a real signing algorithm.**


 Not a cause of breakage on its own, but it's a sign of a misconfigured key set or outdated key material.

Figure G.5: A hallucination example from GPT-4o when passed the DNSViz visualization for our invalid setup