# Leveraging Certificate Transparency to Mitigate Downgrade Attacks

Hyunsoo Kim
Seoul National University
Seoul, Republic of Korea

Myungbin Hwang
Seoul National University
Seoul, Republic of Korea

Taekyoung Kwon
Seoul National University
Seoul, Republic of Korea

## ABSTRACT

Despite the widespread adoption of TLS to secure many protocols such as the web, DNS, and email, downgrade attacks remain a significant vulnerability—particularly when clients opportunistically fall back to unencrypted communication. To address this, we propose leveraging Certificate Transparency (CT) as a verifiable source of truth regarding a server's security capabilities. Specifically, we introduce a custom X.509 certificate extension that explicitly declares a server's supported protocols, ports, and TLS versions. This information enables clients to detect downgrade attacks. To assess the feasibility of our approach, we conducted a measurement study of DNS-over-TLS, DNS-over-HTTPS, and SMTP servers. Our results show that the vast majority of certificates are already logged in CT logs. Building on this, we propose a CT oracle that aggregates data from all CT logs to provide a reliable and comprehensive view of certificates.

## CCS CONCEPTS

• **Security and privacy** → **Network security**; • **Networks** → *Application layer protocols.*

## KEYWORDS

Certificate Transparency, Downgrade Attack, X.509 Certificate

**ACM Reference Format:**
Hyunsoo Kim, Myungbin Hwang, and Taekyoung Kwon. 2025. Leveraging Certificate Transparency to Mitigate Downgrade Attacks. In *The 24th ACM Workshop on Hot Topics in Networks (HotNets '25), November 17–18, 2025, College Park, MD, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3772356.3772399


*HotNets '25, November 17–18, 2025, College Park, MD, USA*

ACM ISBN 979-8-4007-2280-6/25/11
https://doi.org/10.1145/3772356.3772399

## 1 INTRODUCTION

Over the past decade, significant efforts have been made to secure Internet protocols through the widespread adoption of traffic encryption, primarily via the Transport Layer Security (TLS) protocol. TLS now serves as the foundational security layer not only for web traffic but also for numerous other protocols that negotiate encryption dynamically, including those for email, messaging, and DNS. However, despite this progress, many real-world deployments continue to support backward compatibility mechanisms, inadvertently exposing applications to downgrade attacks across these diverse use cases [1].

For example, email protocols such as SMTP have been extended to support TLS when both the sending and receiving mail servers enable it. The sending server issues a `STARTTLS` command to determine whether the receiving server supports encryption. Given the sensitive nature of email content, over 90% of email traffic is now encrypted in transit between mail servers [22]. Similarly, DNS over TLS (DoT) [10] and DNS over HTTPS (DoH) [9] were introduced to protect DNS queries and responses from interception or tampering. These encryption mechanisms are increasingly adopted by major browsers and operating systems [12] [13].

However, simply adding encryption to existing protocols is not a complete solution, particularly when encryption is deployed opportunistically or with soft-fail behavior. To maintain compatibility, many protocols fall back to unencrypted or weaker configurations if one endpoint does not support stronger encryption. This fallback behavior creates a critical vulnerability: attackers can exploit the negotiation phase to downgrade the connection, coercing endpoints into using insecure communication. As a result, merely supporting encryption is not enough to defend against downgrade or version rollback attacks.

The fundamental challenge lies in enforcing strict security (i.e., hard-fail) without significantly disrupting legitimate connectivity. Mechanisms such as HTTPS Strict Transport Security (HSTS) and MTA-STS aim to address this issue by requiring encryption, but their adoption remains limited and the server's intention to use encryption is delivered plaintext (say, HTTP header or DNS TXT record), which is again vulnerable to downgrade attacks [4].

We propose a new approach that leverages Certificate Transparency (CT) to enable clients to detect downgrade attacks. CT maintains public, append-only logs that record TLS certificates for nearly all domain-named servers—including web servers, mail servers, and DNS resolvers. We repurpose this infrastructure as an external, verifiable source of truth for a server's security capabilities. To support this, we introduce a new X.509 certificate extension that specifies a server's security features. By comparing the capabilities declared in a server's CT-logged certificate with its actual behavior during connection setup, clients can detect potential downgrade attacks.

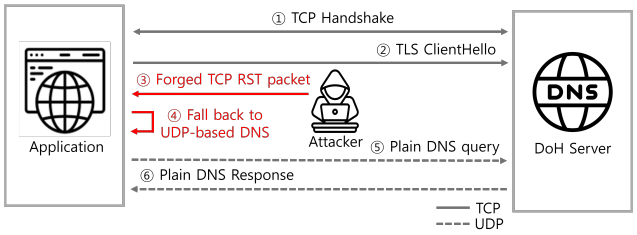


**Figure 1: The downgrade attack on DoH via TCP reset injection is illustrated. If the attacker injects a forged TCP RST packet, the client will abandon the DoH connection and fall back to plaintext DNS over UDP.**

## 2 BACKGROUND AND MOTIVATIONS

### 2.1 Secure DNS and Secure Email

DNS maps domain names to IP addresses or other related data. Traditionally, DNS queries and responses are transmitted in plaintext, making them vulnerable to eavesdropping and tampering. These vulnerabilities can be exploited to compromise user privacy or redirect users to malicious sites. To address this, DNS Security Extensions (DNSSEC) [8] were developed to protect the integrity and authenticity of DNS responses from authoritative name servers to resolvers. DNSSEC uses digital signatures to allow clients to validate DNS responses. However, DNSSEC has technical and operational issues like computational overhead and key management [2]. Due to the slow adoption of DNSSEC, the Internet community has increasingly focused on securing the "last hop" of DNS communication. Protocols like DNS over TLS (DoT) [10] and DNS over HTTPS (DoH) [9] encrypt DNS traffic between clients and recursive resolvers using TLS or HTTPS, providing confidentiality and integrity for DNS lookups and enabling authentication of the resolver's identity.

Email delivery protocols have similarly evolved to incorporate stronger security. SMTP, originally designed for unencrypted communication, now supports TLS encryption through two main mechanisms: SMTPS and STARTTLS [7]. SMTPS refers to SMTP over TLS on a dedicated port (TCP 465), where a TLS handshake occurs immediately upon connection. In contrast, STARTTLS is an SMTP extension that upgrades an existing plaintext connection to a TLS-encrypted one on standard SMTP ports (25 for server-to-server delivery and 587 for mail submission). In a typical STARTTLS exchange, the client connects in plaintext, then issues a STARTTLS command to initiate TLS if the server supports it. Both SMTPS and STARTTLS use X.509 certificates to authenticate mail servers and establish encrypted channels, helping prevent attackers from intercepting or modifying email content in transit. When properly enforced by both sending and receiving servers, these mechanisms significantly improve protection against passive surveillance and man-in-the-middle (MitM) attacks.

### 2.2 Downgrade Attacks on DNS and Email

While extending DNS and SMTP to support encrypted message delivery represents a significant security advancement, it also introduces opportunities for downgrade attacks. In a downgrade attack, an active adversary deceives one or both endpoints into abandoning a more secure or newer protocol or mode in favor of a weaker or older alternative—thereby enabling the attacker to exploit known vulnerabilities. Both DoT/DoH and STARTTLS are vulnerable to such attacks if clients are permitted to "fall back" to unencrypted communication. [11] [16] [18].

In the case of DNS, most clients—typically implemented in browsers and operating systems—use DoT/DoH in an opportunistic mode. That is, the client attempts to establish a DoT or DoH connection by initiating a TLS handshake (via a `ClientHello` message). If the TLS connection fails, the client silently falls back to traditional unencrypted DNS. This behavior creates an opening for attackers, who can disrupt DoT/DoH connections (e.g., by interfering with TCP or TLS handshakes), thereby forcing clients to revert to plaintext DNS. Huang et al. [11] studied several such downgrade vectors and showed that all major browsers they tested eventually reverted to insecure DNS under specific failure conditions—without notifying the user. Figure 1 illustrates a TCP RST injection attack, where an on-path adversary sends a forged TCP RST packet to abruptly terminate the session. This interruption causes the client to fall back to plaintext DNS, exposing the user to spoofing and surveillance—despite the illusion that DoH is still in use. The lack of user warnings and the automatic fallback behavior make downgrade attacks a significant threat to DNS privacy and integrity.

In email delivery, the primary downgrade threat is the STARTTLS stripping attack—a form of man-in-the-middle (MitM) attack targeting SMTP. Because the SMTP session begins in plaintext, an active attacker can intervene early in

the connection. By modifying the server's response to the client's EHLO command, the attacker can strip the "STARTTLS" capability advertisement. If the client never sees the 250-STARTTLS line, it will assume the server does not support TLS. Because most SMTP implementations adopt a soft-fail policy, the sending mail server will transmit email in plaintext if TLS appears unavailable. Studies have shown that a large fraction of SMTP clients and servers are vulnerable to such downgrade tactics [18].

## 2.3 Certificate Transparency (CT)

The Public Key Infrastructure (PKI) faces a longstanding challenge of certificate misissuance—cases where a Certificate Authority (CA) mistakenly or maliciously issues a certificate for a domain it does not control. To mitigate this risk, Google launched the Certificate Transparency (CT) project in 2013 [15], aiming to make certificate issuance publicly visible and auditable. CT operates as a system of public, append-only logs that record TLS certificates. These logs are cryptographically secured using Merkle tree structures, ensuring that once a certificate is added, it cannot be altered or removed. Before issuing a new certificate, a CA submits a pre-certificate to at least two CT logs. In response, each log returns a Signed Certificate Timestamp (SCT)—a cryptographic promise that the certificate will be included in the log within a specified time frame. The SCT is then embedded in the final certificate,[1] allowing clients and monitors to verify its inclusion. This system ensures transparency and accountability: if a certificate is not publicly logged, browsers and monitors can flag it as potentially illegitimate.

CT logging gained rapid adoption as browser vendors began enforcing it to prevent undetected certificate misissuance. Today, nearly all publicly trusted certificates are logged in CT by default, as CAs are required to do so to comply with browser policies. As a result, CT has effectively become the de facto global database of publicly trusted certificates.

In addition to preventing misissuance, CT logs serve as a valuable resource for security monitoring and research. Since the logs are public and include nearly all valid certificates, they can be mined to discover newly issued certificates or track emerging domain names [14]. However, CT logs are not optimized for domain-based queries. To address this limitation, CT monitors were introduced. These services aggregate log entries and provide searchable web interfaces or APIs that allow users to query certificates issued for particular domains. CT monitors have become essential tools for both administrators and researchers. Domain owners can quickly detect unauthorized certificates, while researchers can use the data to build comprehensive lists of active domains—without relying on large-scale port scans. As CT logs reflect nearly all TLS-enabled sites, mining them for domain data has become a standard and reliable practice.

[1]Alternatively, an SCT can be delivered to the TLS client during the handshake instead of being embedded.

## 2.4 Motivation – Using CT to Thwart Downgrade Attacks

As almost every TLS server now has its certificate logged, clients can learn in advance what certificate to expect from a server of interest. The central idea of this paper is to prevent downgrade attacks by leveraging CT logs. Not only web servers but also any servers with domain names have CT-logged certificates. For example, popular DNS resolvers (Quad9, Cloudflare, Google) and major email providers (Gmail, Outlook) have their certificates that appear in CT logs.

The accessibility of CT logs and monitors enables clients to proactively assess the security capabilities of a target server. Before initiating a connection, a client can query a CT monitor to retrieve the server's certificate. During the TLS handshake, the client then checks whether the server's behavior aligns with the security capabilities specified in the CT-logged certificate. If the server deviates from what is declared—such as failing to support advertised encryption modes—the client can suspect a downgrade attack and abort the connection. In this way, CT functions as a global source of truth for legitimate certificates, empowering clients to reject insecure connections by cross-checking server behaviors against publicly logged certificate data.

# 3 CT ADOPTION MEASUREMENT

To evaluate the practicality of leveraging CT for preemptive detection of insecure TLS connections, we conducted a measurement study of DNS and email services in October 2024 through Censys scans [3]. Our objective is to assess the current level of CT adoption across these protocols and measure how many of the corresponding certificates are publicly available in CT logs.

## 3.1 DNS

To assess CT adoption across encrypted DNS protocols, we collected TLS certificates from servers supporting DoT/DoH. Specifically, we identified 29,270 DoT servers and 632 curated DoH provider URLs. Of these, 18,237 DoT certificates (62.3%) and all 632 DoH certificates (100.0%) were found to be logged in CT logs. Table 1 summarizes the CT adoption rates and SCT count distributions across both protocols. Notably, the majority of CT-logged certificates in both datasets contained exactly two SCTs.

Figure 2 illustrates the distribution of SCTs among major CT logs. In both DoT and DoH datasets, the vast majority of

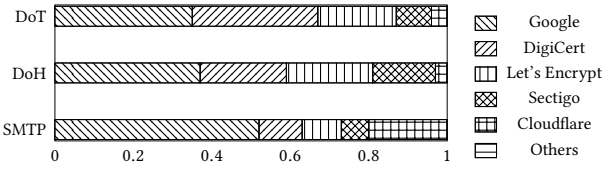


**Figure 2: Distribution of SCTs of certificates across CT logs is investigated for DoT, DoH, and SMTP servers.**

SCTs were issued by a small number of well-known logs operated by Google (Xenon2024), DigiCert (Yeti2024, Yeti2025), and Let's Encrypt (Oak2024H2).

## 3.2 Email

We also analyzed the email ecosystem to evaluate TLS support and CT participation among SMTP servers. Using a dataset of 3,290 mail server domains, we attempted TLS handshakes over the three standard email delivery ports: port 25 (STARTTLS optional), port 465 (implicit TLS), and port 587 (STARTTLS). STARTTLS on port 25 was the most widely supported, with 3,102 successful handshakes, followed by STARTTLS on port 587 (439) and implicit TLS on port 465 (402). Other configurations—such as direct TLS on ports 25 or 587—were rarely observed. In total, we collected 3,302 distinct TLS certificates, with 450 domains supporting multiple port/protocol combinations.

As summarized in Table 1, 2,621 certificates (79.4%) included at least one SCT, indicating strong CT adoption within the email ecosystem. We further examined the CT logs responsible for issuing SCTs to these certificates. As shown in Figure 2, the majority of SCTs originated from logs operated by Google and Cloudflare—specifically, Google's Xenon2024 and Argon2023, and Cloudflare's Nimbus2023.

| | DoT | DoH | SMTP |
|---|---|---|---|
| Total Certs | 29,270 | 632 | 3,302 |
| Logged | 18,237 (62.3%) | 632 (100.0%) | 2,621 (79.4%) |
| Not Logged | 11,033 (37.7%) | 0 (0.0%) | 681 (20.6%) |
| 1 SCT | – | – | 675 |
| 2 SCTs | 12,405 | 563 | 1,929 |
| 3 SCTs | 5,829 | 69 | – |
| >3 SCTs | 3 | – | 17 |

**Table 1: CT adoption rates and the distributions of the number of SCTs in their certificates are measured for DoT, DoH, and SMTP servers.**

| Protocol | SSLMate | | crt.sh | |
|---|---|---|---|---|
| | Hit (%) | Miss (%) | Hit (%) | Miss (%) |
| **Total (21,490)** | 19,534 (90.9%) | 1,956 (9.1%) | 21,453 (99.8%) | 37 (0.2%) |
| DoT (18,237) | 17,605 (96.5%) | 632 (3.5%) | 18,201 (99.8%) | 36 (0.2%) |
| DoH (632) | 631 (99.8%) | 1 (0.2%) | 632 (100.0%) | 0 (0.0%) |
| SMTP (2,621) | 1,298 (49.5%) | 1,323 (50.5%) | 2,620 (99.9%) | 1 (0.1%) |

**Table 2: Hit and miss counts of certificates from two CT monitors (SSLMate and crt.sh) are shown.**

## 3.3 CT Monitor

To evaluate the coverage of public CT monitors, we queried two widely used monitors, SSLMate and crt.sh, for certificates containing at least one SCT. These monitors were chosen due to their public accessibility via free API interfaces [5]. As summarized in Table 2, crt.sh consistently achieved near-perfect coverage across all datasets: DoH (100.0%), DoT (99.8%) and SMTP (99.9%). In contrast, SSLMate showed moderate coverage for DoH and DoT (99.8% and 96.5%, respectively), but performed poorly on the SMTP dataset, with a hit rate of only 49.5%.

These discrepancies are largely due to factors such as certificate expiration, unmonitored CT logs, delays in monitor processing, and filtering or search limitations based on certificate subject fields [17]. It is important to note that all certificates in this evaluation included SCTs, confirming their submission to CT logs and, in principle, their retrievability.

Nonetheless, the substantial number of missing certificates—especially in the DoT and SMTP datasets—highlights the incompleteness and inconsistency of public CT monitor coverage. To account for this limitation, our study assumes a CT Monitor Oracle capable of retrieving all certificates.

# 4 MITIGATING DOWNGRADE ATTACKS

We argue that the CT oracle be leveraged to prevent downgrade attacks between clients and servers with domain names. The key idea is to upload a server's certificate to CT logs so as to allow potential clients to learn the "protocol capability" of the server, and to enforce a client to use the most secure mode/version of the protocol of the server. This strategy requires two main components: (1) a new X.509 extension that specifies the protocol capability in the certificate, and (2) a CT oracle-aware process by which a client queries the CT oracle for the certificate of the server before establishing a connection. By combining these, a client can detect if an attacker attempts to downgrade a connection or to inject a fraud certificate. We assume that the CT oracle is reliable and up-to-date[2].

[2]Actually, the CT oracle should retrieve certificates from CT logs periodically, to be discussed later.

```
                ...
X509v3 Basic Constraints: critical
        CA: FALSE
1.3.6.1.4.1.55555.1:
        Protocol: DoT
        Port: 853
        IP Address: 147.46.80.1
        Version: TLS1.3
X509v3 Extended Key Usage:
        TLS Web Server Authentication
                ...
```

**Figure 3: An example of the proposed X.509 v3 `Protocol Capability` extension is illustrated in the blue box.**

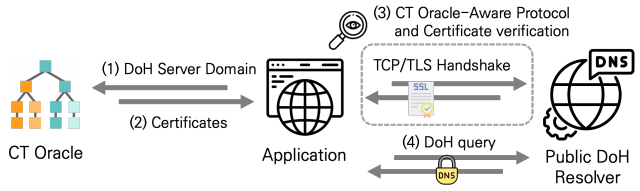


**Figure 4: The CT oracle-aware certificate checking process is illustrated. If the server's behaviors (during the connection setup) do not align with the protocol capability in the CT-logged certificate, the client will abort the connection.**

## 4.1 X.509 Certificate Extension for *Protocol Capability*

We should extend a certificate with metadata about the protocol capability. We design a new custom X.509 v3 extension *Protocol Capability* that specifies the following details of protocols and connections. This non-critical extension (so that unaware clients simply ignore it) will declare what application protocol the certificate is for, which port (or port range) is to be used. Optionally we can add the relevant protocol version (say, TLS version) and IP address(es) to the certificate. All sub-fields are optional and can be included as needed to precisely describe the service.

For instance, as shown in Figure 3, a public DNS resolver's certificate may include (i) Protocol: DoT (DNS over TLS) and (ii) Port: 853 (the standard port for DoT). Adding the TLS protocol version can also be useful if the DoT server runs the latest version TLS (e.g., enforcing TLS 1.3 to prevent downgrade to older TLS versions).

Technically, adding a custom extension is fully supported within the X.509 standard, which permits the use of arbitrary object identifiers (OIDs) for new extensions. Organizations can obtain a Private Enterprise Number (PEN) from IANA and define sub-OIDs under their PEN for custom purposes. For our prototype, we registered a dedicated OID to encode the extension. Using OpenSSL or standard CA software, this extension can be incorporated by specifying the OID and the corresponding values—for instance, through an OpenSSL configuration file that defines the extension fields using an ASN1:SEQUENCE structure.

## 4.2 CT Oracle-Aware Certificate Checking

The second part of our mitigation strategy involves having the client verify the server's behavior against the protocol capabilities declared in its CT-logged certificate before establishing a secure connection. This validation step effectively prevents downgrade attacks and helps detect fraudulent certificates.

The CT oracle is assumed to periodically retrieve certificates from CT logs—for example, every 24 hours. When a CT log receives a pre-certificate from a CA, it typically has up to 24 hours, known as the Maximum Merge Delay (MMD), to incorporate the certificate into its Merkle tree. In practice, servers often renew their certificates with overlapping validity periods, ensuring that newly issued certificates are logged and available well before they are actively deployed. This overlap provides a natural buffer of several days, allowing clients to retrieve and verify CT-logged certificates in time.

Figure 4 illustrates the certificate checking process using a DNS-over-HTTPS scenario as an example.

(1) The client application (e.g., a web browser) queries the CT oracle to obtain the server's certificate before initiating a connection.

(2) The CT oracle returns the most recently logged certificate[3], which includes the server's declared protocol capabilities (e.g., DoT on port 853, DoH on port 443, TLS 1.3).

(3) The client establishes a TCP/TLS connection according to the protocol capabilities. During the TLS handshake, the server presents its certificate, which the client compares against the one obtained from the CT oracle:

- If they match, the client proceeds normally.
- If there is a mismatch, the client re-queries the CT oracle for an updated certificate. If the new response matches the server's certificate, this is interpreted as a legitimate certificate rotation. The client updates its cache and proceeds. If the mismatch persists, the connection is deemed unsafe, and the client hard-fails the connection.

(4) Once the certificate is validated and the server's behavior aligns with the declared protocol capabilities, the client considers the connection secure and proceeds (e.g., sends a DNS query over HTTPS).

[3]In practice, it will return all the valid (say, not expired) certificates for a given domain name.

To improve efficiency, the client maintains a local cache of certificates retrieved from the CT oracle. Before each connection attempt, the client checks its cache; only in the event of a cache miss does it query the CT oracle.

# 5 DISCUSSIONS

## 5.1 Reliability of CT Monitors in Practice

The effectiveness of the proposed mitigation strategy hinges on the assumption of trustworthy and complete data about certificates from CT monitors (or more fundamentally from CT logs). However, recent studies have shown significant reliability issues with CT monitors. Li et al. [17] demonstrated that popular CT monitors such as crt.sh and SSLMate frequently fail to provide a complete set of certificates, not to mention delays or synchronization problems. Sun et al. [21] confirmed these issues later, noting that even well-known monitors regularly fail to return timely and accurate certificate data. Given the reports on the unreliability of CT monitors, relying only on the monitors is insufficient for trustworthy operations. A CT oracle should be able to download data from the CT logs directly as well as retrieving data from the CT monitors.

## 5.2 Relation to RFC 9460 Service Bindings

RFC 9460 (SVCB and HTTPS Resource Records) [20] and RFC 9461 (Service Binding Mapping for DNS Servers) [19] similarly aim to allow clients to discover a server's protocol capabilities during DNS resolution. In this respect, they offer a mechanism comparable to our proposed approach, with the added advantage that DNS lookups are a required step prior to connection establishment, making capability advertisement more naturally integrated into the connection process. This integration may also enhance deployability, as clients can obtain all necessary connection hints upfront. However, DNS-based capability records are susceptible to tampering and suppression—such as cache poisoning or on-path manipulation—unless DNSSEC is fully deployed. Unfortunately, DNSSEC adoption remains limited in practice, with only about 5% of domains signed as of 2023 [6]. In contrast, our approach leverages CT, which provides cryptographically verifiable and tamper-resistant logs. As a result, it is significantly more difficult for attackers to forge or manipulate CT data.

# 6 CONCLUSION

In this work, we proposed a method to mitigate downgrade attacks, which remain widespread across many Internet protocols. Although mechanisms like DoH, DoT, and STARTTLS are increasingly adopted, their reliance on soft-fail and fallback behavior leaves them vulnerable to active interference. Our approach shifts from opportunistic to validated security by leveraging Certificate Transparency (CT) and extending certificates to advertise a server's supported protocol capabilities. We demonstrated that reusing the existing CT infrastructure is both practical and deployable. CT logs already serve as a global repository of PKI certificates, and CT monitors provide domain-based certificate visibility. With our approach, clients can detect downgrade attempts for any server—such as DNS resolvers or email agents—that presents a valid certificate, thereby preventing fallback to insecure communication channels.

# ACKNOWLEDGEMENTS

This work was supported by the Ministry of Science and ICT (MSIT), Korea, under the Information Technology Research Center (ITRC) support program (IITP-2025-2021-0-02048) supervised by the Institute for Information & Communications Technology Planning & Evaluation (IITP). It was also supported by the IITP grant funded by the Korea government (MSIT) (No. RS-2021-II211343, Artificial Intelligence Graduate School Program (Seoul National University)) and by the Next-generation Cloud-native Cellular Network Leadership Program (IITP-2025-RS-2024-00418784). In addition, this work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2022R1A2C2011221, RS-2023-00220985).

# REFERENCES

[1] Eman Salem Alashwali and Kasper Rasmussen. 2018. What's in a downgrade? A taxonomy of downgrade attacks in the TLS protocol and application protocols using TLS. In *Security and Privacy in Communication Networks: 14th International Conference, SecureComm 2018, Singapore, Singapore, August 8-10, 2018, Proceedings, Part II*. Springer, 468–487.

[2] Taejoong Chung, Roland van Rijswijk-Deij, Balakrishnan Chandrasekaran, David Choffnes, Dave Levin, Bruce M Maggs, Alan Mislove, and Christo Wilson. 2017. A Longitudinal,{End-to-End} View of the {DNSSEC} Ecosystem. In *26th USENIX Security Symposium (USENIX Security 17)*. 1307–1322.

[3] Zakir Durumeric, David Adrian, Ariana Mirian, Michael Bailey, and J. Alex Halderman. 2015. A Search Engine Backed by Internet-Wide Scanning. In *22nd ACM Conference on Computer and Communications Security*.

[4] Ian Foster, Jenny Larson, Max Masich, Alex C. Snoeren, Stefan Savage, and Kirill Levchenko. 2015. Security by Any Other Name: On the Effectiveness of Provider Based Email Security. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 450–464. https://doi.org/10.1145/2810103.2813628

[5] Google. [n. d.]. Certificate Transparency Monitors. https://certificate.transparency.dev/monitors/. Accessed: 2024-06-27.

[6] Elias Heftrig, Haya Shulman, and Michael Waidner. 2023. Downgrading {DNSSEC}: How to Exploit Crypto Agility for Hijacking Signed Zones. In *32nd USENIX Security Symposium (USENIX Security 23)*. 7429–7444.

[7] Paul E. Hoffman. 2002. SMTP Service Extension for Secure SMTP over Transport Layer Security. RFC 3207. https://doi.org/10.17487/RFC3207

[8] Paul E. Hoffman. 2023. DNS Security Extensions (DNSSEC). RFC 9364. https://doi.org/10.17487/RFC9364

[9] Paul E. Hoffman and Patrick McManus. 2018. DNS Queries over HTTPS (DoH). RFC 8484. https://doi.org/10.17487/RFC8484

[10] Zi Hu, Liang Zhu, John Heidemann, Allison Mankin, Duane Wessels, and Paul E. Hoffman. 2016. Specification for DNS over Transport Layer Security (TLS). RFC 7858. https://doi.org/10.17487/RFC7858

[11] Qing Huang, Deliang Chang, and Zhou Li. 2020. A comprehensive study of {DNS-over-HTTPS} downgrade attack. In *10th USENIX Workshop on Free and Open Communications on the Internet (FOCI 20)*.

[12] Karel Hynek. 2021. The Prevalence of DNS over HTTPS. https://blog.apnic.net/2021/02/16/the-prevalence-of-dns-over-https. Accessed: 2024-06-27.

[13] Or Katz. 2022. DNS Threat Report — Q3 2022. https://www.akamai.com/blog/security/dns-threat-report-q3-2022. Accessed: 2024-06-27.

[14] Brian Kondracki, Johnny So, and Nick Nikiforakis. 2022. Uninvited guests: Analyzing the identity and behavior of certificate transparency bots. In *31st USENIX Security Symposium (USENIX Security 22)*. 53–70.

[15] Ben Laurie, Adam Langley, and Emilia Kasper. 2013. Certificate Transparency. RFC 6962. https://doi.org/10.17487/RFC6962

[16] Sangtae Lee, Youngjoo Shin, and Junbeom Hur. 2020. Return of version downgrade attack in the era of TLS 1.3. In *Proceedings of the 16th International Conference on Emerging Networking Experiments and Technologies*. 157–168.

[17] Bingyu Li, Jingqiang Lin, Fengjun Li, Qiongxiao Wang, Qi Li, Jiwu Jing, and Congli Wang. 2019. Certificate transparency in the wild: Exploring the reliability of monitors. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2505–2520.

[18] Damian Poddebniak, Fabian Ising, Hanno Böck, and Sebastian Schinzel. 2021. Why {tls} is better without {starttls}: A security analysis of {starttls} in the email context. In *30th USENIX Security Symposium (USENIX Security 21)*. 4365–4382.

[19] Benjamin M. Schwartz. 2023. Service Binding Mapping for DNS Servers. RFC 9461. https://doi.org/10.17487/RFC9461

[20] Benjamin M. Schwartz, Mike Bishop, and Erik Nygren. 2023. Service Binding and Parameter Specification via the DNS (SVCB and HTTPS Resource Records). RFC 9460. https://doi.org/10.17487/RFC9460

[21] Aozhuo Sun, Jingqiang Lin, Wei Wang, Zeyan Liu, Bingyu Li, Shushang Wen, and Qiongxiao Wang Fengjun Li. 2024. Certificate Transparency Revisited: The Public Inspections on Third-party Monitors. In *NDSS*.

[22] Maxie Wang. 2020. Winding Down STARTTLS Everywhere. https://www.eff.org/deeplinks/2020/12/winding-down-starttls-everywhere. Electronic Frontier Foundation, Accessed: 2024-06-27.