



STEK Sharing is Not Caring: Bypassing TLS Authentication in Web Servers using Session Tickets

Sven Hebrok
Paderborn University

Tim Leonhard Storm
Paderborn University

Felix Matthias Cramer
Paderborn University

Maximilian Radoy
Paderborn University

Juraj Somorovsky
Paderborn University

Abstract

TLS session resumption with session tickets is a widely supported mechanism designed to accelerate TLS connections. It allows a server to use a symmetric Session Ticket Encryption Key (STEK) to encrypt a TLS context in a so-called session ticket, provide the ticket to the client, and later decrypt it during session resumption to obtain the context and seamlessly resume the session. Proper STEK handling is critical and may get complex in scenarios such as virtual hosting, where a single physical server accommodates multiple virtual hosts. Most importantly, these virtual hosts must remain securely isolated, even when they rely on the same TLS STEK for session protection.

We demonstrate how TLS session resumption in virtual hosting can introduce *session ticket confusion* vulnerabilities, potentially enabling the bypass of both server and client authentication. To validate the practicality of these attacks, we analyzed four open source implementations and conducted a large-scale evaluation. Our findings reveal that all four implementations – Apache, nginx, (Open)LiteSpeed, and Caddy – were vulnerable to client authentication bypasses. In our large-scale scans, we identified six clusters of vulnerable providers, including Fastly, which were susceptible to server authentication bypasses. Our results highlight inconsistent isolation of virtual hosts following TLS session resumption, exposing critical security gaps in modern virtual hosting environments.

1 Introduction

Transport Layer Security (TLS) [48, 49] plays a crucial role in securing modern internet communications by providing confidentiality, integrity, and authenticity for client-server applications. To achieve these security goals, both communication parties perform a TLS handshake at the beginning of the connection. This handshake follows a pre-defined protocol flow (cf. Figure 1a) and results in shared secrets that can be subsequently used to protect transmitted application data. As TLS operates below the application layer in the TCP/IP

protocol stack, it can secure a wide range of protocols, from HTTP and FTP to email exchange and mobile communication networks [52]. These scenarios impose different demands and security goals; some require client authentication (e.g., [52]), while others demand high performance.

Virtual Hosting. One of the most common use cases for TLS is securing websites. When deployed on a server, the TLS engine secures a website by protecting all HTTP data exchanged in the application layer. However, due to the limitations of IPv4 addresses and the need to optimize resources, web application servers typically host multiple domains on a single machine rather than dedicating one server per domain. This practice, known as virtual hosting, is widely implemented by open-source libraries and extensively deployed by Content Delivery Networks (CDNs). A single CDN server often hosts multiple distinct domains using virtual hosting, each with its own TLS certificate.

The shared environment in virtual hosting requires robust isolation mechanisms and involves multiple validation steps for both the server and the client. The server must evaluate the Server Name Indication (SNI), which the client includes in the first TLS handshake message (`ClientHello`). The SNI specifies the domain the client intends to connect to, allowing the server to present the appropriate certificate. Additionally, the server must verify that the SNI matches the HTTP `Host` header to prevent mismatches that could lead to security vulnerabilities. On the client side, it is crucial to validate the TLS certificate presented by the server, ensuring it corresponds to the intended domain. Any failure in maintaining this isolation could allow attackers to bypass authentication, undermining both server and client authentication guarantees.

Performance Improvements with Session Resumption. Next to security requirements, one of the key requirements for virtual hosting environments and application hosting in general is performance. However, TLS involves computationally intensive cryptographic public-key operations, particularly

during the handshake process (e.g., computing RSA signatures). To mitigate this overhead in subsequent connections, session resumption mechanisms were introduced [18, 48, 49]. These mechanisms allow servers to restore a previous session without repeating the full TLS handshake. One widely supported and standardized mechanism for improving performance in virtual hosting environments is session resumption with session tickets [18] (cf. Figure 1c). A session ticket is a cryptographic token protected by the server’s symmetric Session Ticket Encryption Key (STEK). After a successful handshake, the server issues this ticket to the client, allowing it to resume the session in subsequent connections. During resumption, the client presents the session ticket, which the server decrypts to retrieve the necessary session keys. This process enables the session to continue without costly public-key operations or certificate validation.

Security Implications of Session Resumption. While session resumption improves the TLS performance, it can also introduce new security risks. In 2015, Delignat-Lavaud and Bhargavan [14] highlighted isolation flaws in session resumption with session IDs [49], demonstrating how improper isolation between servers could lead to critical vulnerabilities bypassing server authentication. Since then, the landscape has shifted dramatically,¹ and session tickets have largely replaced session IDs as the primary mechanism for session resumption. This evolution necessitates a fresh investigation into the security of session resumption in modern TLS implementations, particularly with respect to server-side HTTPS multiplexing. In this work, we focus on an automated analysis of session ticket-based resumption.

It has already been demonstrated that session resumption with session tickets may undermine the security of TLS. In 2016, Springall, Durumeric, and Halderman [53] analyzed the potential impact of session resumption on forward secrecy. In 2017, Valsorda [60] pointed out that improper session ticket protection can lead to full session compromise. In 2023, Hebrok et al. [28] confirmed these dangers in large-scale analyses. They detected servers using weak STEKs or reused keystreams. These attacks primarily affected the confidentiality guarantees; an attacker exploiting a weak STEK could decrypt the session ticket and thus break current, previous, and future TLS sessions, effectively breaking perfect forward security guarantees. However, the complexity of internal routing mechanisms in virtual hosting environments and the complexity of session ticket resumption may introduce additional threats, including risks that affect the authenticity guarantees of TLS. This leads us to the first research question.

RQ1: What are the broader implications of attacks exploiting session resumption with session tickets, and how do they affect TLS authentication guarantees?

Motivated by the recent attack developments [14, 28, 60], we systematically studied the security implications of session ticket-based resumption and describe two *session ticket confusion* attacks undermining the authenticity of resumed TLS sessions (cf. Section 3). For both attacks, we define the victim behavior, attack prerequisites, and attack goals.

Server Authentication Attacks. In the first attack scenario, we extend the work of Delignat-Lavaud and Bhargavan [14] and consider an active Machine-in-the-Middle (MitM) attacker who hosts their domain in a virtual hosting environment along with their victim. We assume that both the victim and the attacker host their domain in a virtual hosting environment under different IPs. The attacker reroutes the victim to their IP during the session resumption handshake. If the server can resume the ticket and reroute the connection to the attacker’s domain, the attacker can obtain the victim’s request, including sensitive data like passwords or cookies.

Client Authentication Attacks. In our second attack, our focus extends beyond analyzing isolation in session resumption to exploring whether this mechanism could bypass TLS client authentication, presenting a novel attack not yet considered in the previous literature. TLS client authentication allows clients to authenticate to TLS servers, for example, by using certificates (cf. Figure 1b). In our scenario, we assume an attacker who uses TLS client authentication to authenticate at a server domain they can access. In the resumed handshake, they resume the ticket to a privileged domain to which they have no access. Such a scenario would result in privilege escalation within virtual hosting environments.

Note that detecting any of the attacks is challenging at the TLS level since the resumed handshake does not contain any certificate. This necessitates that the session ticket stores information about the authenticated communication parties, which leads us to the next research question.

RQ2: Do these session ticket confusion vulnerabilities translate to real-world server implementations and virtual hosting environments?

Evaluation of Open-Source Implementations. To answer this research question, we analyzed the security implications of session ticket-based resumption in four web servers in their recent versions: Apache, Caddy, nginx, and (Open)LiteSpeed. Based on the defined attack scenarios, we implemented test cases to assess their applicability by varying the SNI and `Host` header and evaluating potential vulnerabilities in different configurations. While none of the servers were vulnerable to server authentication vulnerabilities, we could bypass TLS

1. For example, Cloudflare started offering free TLS in 2014 [24], increasing TLS and CDN adoption. Let’s Encrypt started in 2015, which also increased TLS adoption [1].

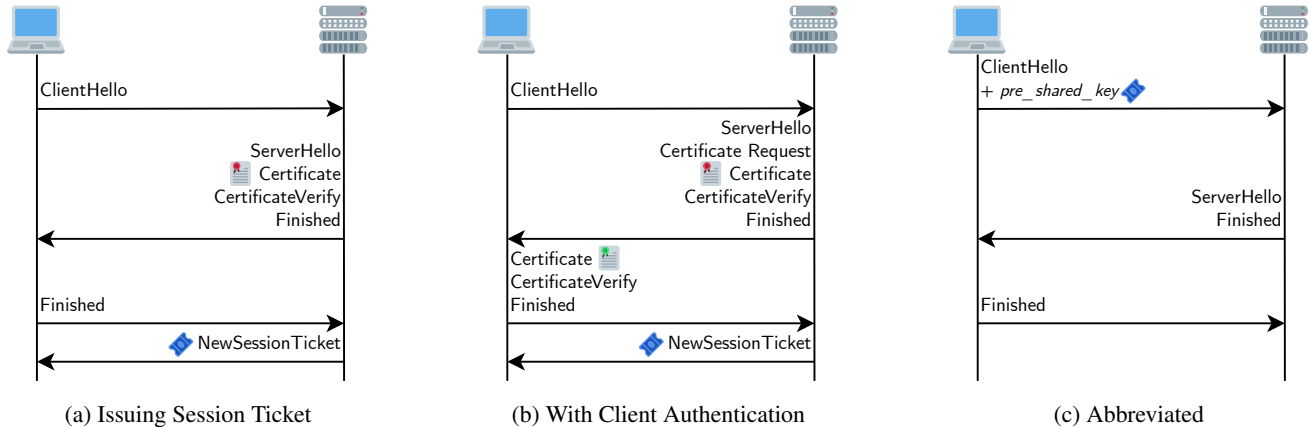


Figure 1: Illustration of TLS 1.3 handshakes: (a) a standard handshake issuing a session ticket, (b) a standard handshake with client authentication, and (c) an abbreviated handshake resuming the session using a session ticket. Note that (c) may depict a resumption of a session with or without client authentication. Neither communication party provides a certificate in the resumed TLS handshake.

client authentication on all the tested servers, relying on TLS session ticket-based resumption. Notably, in LiteSpeed, we could bypass TLS client authentication simply by modifying the `Host` header during the full handshake. This renders more complex attacks involving session resumption unnecessary.

Large-Scale Server Evaluation. To retrieve a broader view of the real TLS server ecosystem, we performed a large-scale analysis of Tranco Top 1M domains [34] and their vulnerability to our attacks targeting TLS server authentication. Springall, Durumeric, and Halderman [53] estimated the prevalence of STEK-sharing based on prefixes to show the impact on forward secrecy. Delignat-Lavaud and Bhargavan [14] proved the existence of this vulnerability with manual testing, because, as they pointed out, evaluating these vulnerabilities on a large scale is challenging. We overcome these challenges by creating a list of candidate domains hosted on the same servers, a large-scale collection of session tickets, and evaluating responses to our resumed requests. With the developed methodology, we were able to detect vulnerabilities affecting six clusters of providers. Among the providers responsible for the vulnerable domains were Fastly and DDoS-Guard.

Contributions. Our main research contributions are:

1. *Attacks:* We systematically analyzed the impact of session tickets on the TLS connection and presented novel attacks that bypass client and server authentication.
2. *Open-Source Analysis:* We analyzed leading open-source TLS implementations to assess their handling of virtual host isolation in session resumption. Our analysis uncovered critical vulnerabilities in widely used server software, including Apache, nginx, (Open)LiteSpeed, and Caddy.

3. *Large-Scale Analysis:* We proposed an efficient methodology for large-scale scanning of server authentication bypass vulnerabilities. Using this approach, we identified multiple vulnerable CDNs, including Fastly. Additionally, we uncovered an unrelated vulnerability in Cloudflare SaaS.

Responsible Disclosure. We reported our findings to the developers of all affected open-source implementations, including Apache (CVE-2025-23048), nginx (CVE-2025-23419), (Open)LiteSpeed, and Caddy. We notified the operators of the vulnerable servers identified in our real-world evaluation.

2 Background

2.1 TLS

TLS is a cryptographic protocol designed to ensure secure communication [48, 49]. It is commonly used to protect web traffic but is also found in many other applications [23, 29, 30]. TLS ensures the confidentiality, integrity, and authenticity of transmitted data. The most recent version is TLS 1.3, with TLS 1.2 still being widely used [9, 39]. Upon starting a new connection, TLS performs a handshake between the client and server. We depict a TLS handshake in Figure 1a. The `ClientHello` and `ServerHello` messages negotiate cryptographic parameters and exchange key shares to establish shared secrets. To authenticate, the server sends its certificate to the client. This certificate includes essential components such as the server’s public key, the domain name(s), and a digital signature from a Certificate Authority. The client then validates this certificate. If the certificate is deemed valid and trusted, and the domain name matches the one the client is attempting to connect to, the client can confidently conclude

that the public key in the certificate genuinely belongs to that domain. Together with a signature provided by the server in the `CertificateVerify` message, the client verifies that the server has the private key. This confirms that the server belongs to the intended domain and is not being impersonated. Finally, both parties exchange `Finished` messages to conclude the handshake and start exchanging application data.

2.2 TLS Client Authentication

TLS also supports client authentication, which, like server authentication, relies on certificates. The process closely mirrors the server authentication flow, but it is less commonly used in the general World Wide Web. Instead, it finds applications in specialized domains, such as authenticating to wireless networks [52] or securing APIs [13]. [Figure 1b](#) illustrates the TLS handshake process when client authentication is enabled. After the `ServerHello` message, the server sends a `CertificateRequest` message to the client, specifying the expected client certificate's attributes, including a list of acceptable Certificate Authorities. The client responds by sending its certificate along with a `CertificateVerify` message, proving possession of the private key corresponding to the public key in the certificate. Aside from these additional steps, the handshake remains identical to the standard TLS handshake.

2.3 TLS Session Resumption

Session resumption aims to accelerate subsequent TLS connections by reusing previously negotiated parameters and secrets, reducing the computational load for both parties. As only these two parties have access to the previously negotiated secrets, they do not need to authenticate again. Therefore, they do not exchange a certificate or signature, reducing the computational load and accelerating the handshake.

One approach to implement session resumption is with *session tickets* [18, 50]. The server issues a session ticket containing all the necessary parameters and secrets to restore the session state at the next connection. This session ticket is encrypted and authenticated with a distinct key, the STEK, and sent to the client in the `NewSessionTicket` message. Upon resumption, the client includes the previously received ticket in its `ClientHello` message. The server will attempt to decrypt the ticket with its STEK and resume the session. Because the client handles the ticket as an *opaque* value, this enables session resumption as long as the server can understand the structure of a ticket. For the server, the resumption is stateless, meaning it does not need to store anything for the resumption, reducing overhead. The client must store the secrets negotiated during the session in which the ticket was issued, alongside the ticket itself. If a server can decrypt the session ticket and decides to accept the given ticket, it skips

the authentication with the certificate and proceeds to encrypted communication using the secrets restored from the ticket (optionally issuing a new ticket in the process). The client also does not authenticate itself and proceeds to the encrypted communication using the stored secrets.

Note that with TLS 1.3, the handshake has been overhauled, and session tickets have been technically replaced by the Pre-Shared Keys (PSK) mechanism [48], which enables other forms of abbreviated handshakes as well. Conceptually, tickets still work the same way as before and use the same structure [28].

2.4 Virtual Hosting and CDNs

Web servers often employ virtual hosting to host multiple websites on the same machine. For any incoming connection, the server delegates the request to the responsible virtual host, which serves the requested resource. HTTP uses the `Host` header, which specifies the domain and port of the requested service, allowing the virtual host responsible for the domain to be chosen [40].

A specific form of virtual hosting is reverse proxying. Here, a server acts as a middlebox between clients and one or more origin servers. Instead of directly managing different virtual hosts, the reverse proxy passes requests on to other servers and relays the response instead. Typical use cases include caching static resources, load balancing, filtering specific traffic, or rewriting specific requests before delegating any requests to the responsible origin server(s). To this end, they need to access the plaintext requests. That is, they need to act as the TLS server and provide a valid certificate for any virtual host. As the certificate is required before receiving the HTTP request, the server cannot use the `Host` header.

One approach to providing virtual hosting would be to use a certificate that covers all domains hosted on a server by using the Subject Alternative Name (SAN) field. However, using a single certificate for multiple (possibly unrelated) domains may cause security issues [14]. As an alternative, the client can include the SNI extension in their `TLS ClientHello` to indicate the desired domain name [5]. The server can then choose a valid certificate during the handshake, allowing one certificate to be issued per domain. The SNI extension is mandatory to implement since TLS 1.3 [48].

CDNs. CDNs are commonly used to speed up website access by deploying reverse proxies physically positioned near the user. These proxies are called *edge servers*. CDNs deploy many edge servers on distinct IPs and locations to improve redundancy and response times in different locations. By relying on this dedicated CDN infrastructure, companies can improve a website's performance in different locations and benefit from additional security measures, such as protection against Distributed denial-of-service (DDoS) attacks. CDNs typically employ virtual hosting to serve many customers

using limited hardware. For enterprise-level customers, more specialized setups are possible. For example, to support legacy clients, customers may request a dedicated IP address for their domain that does not rely on SNI.²

3 Vulnerabilities from STEK Sharing

CDNs benefit from sharing their STEK across their infrastructure; a client routed to a different node due to load balancing can still resume their session. Similarly, single servers may use the same STEK for all virtual hosts to simplify key management. In this work, we describe two *session ticket confusion* attacks caused by STEK sharing that circumvent either server or client authentication. Both attacks use tickets issued for one domain and resume them, confusing the server into serving a different domain. We consider two attack scenarios with varying capabilities of the attacker and impact. The first attack is similar to the scenario described by Delignat-Lavaud and Bhargavan [14] in which server authentication is bypassed, enabling TLS-MitM attacks. In our second attack, we circumvent client authentication, allowing an attacker to access restricted sites.

3.1 Server Authentication Bypass

The first attack we analyze in this work was previously outlined by Delignat-Lavaud and Bhargavan [14]. We exploit an issue in a CDN to attack a client visiting a website hosted by that CDN. The attacker’s goal is to retrieve the plaintext request and respond to it, effectively acting as a TLS MitM.

Victim Behavior. A victim connects to a website using TLS and is issued a session ticket. Within this connection, the victim validates the TLS certificate and ensures they are retrieving the correct website. The victim, within the session ticket’s lifetime, performs another request to the website and resumes the session ticket for this request. As this is a resumption, the server does not present its certificate again.

Attacker Goal and Impact. The attacker’s goal is to act as a MitM on the TLS layer for the second request. This exposes any information within this request to the attacker, including (session) cookies and any data the user entered on the website. The attacker can also respond to this request, allowing them to serve malicious content without breaking TLS authentication; for example, respond with a website containing malicious scripts, effectively achieving Cross-Site Scripting (XSS).

Attacker Capabilities. To achieve their goals, we consider an active attacker who can reroute the victim’s network traffic on the IP layer. This can, for example, be achieved by changing the addresses in the IP header, as this is not protected by TLS. We further assume that the attacker hosts a backend

server and registers this at the CDN. The attacker may use any service the CDN offers, including enterprise-level features (cf. Section 2.4), such as a dedicated IP address for their domain.

Attack Description. We provide an overview of the attack in Figure 2a. We assume the victim wants to visit `a.com`, which is hosted on IP 1. The attacker owns `e.com`, which the CDN hosts on IP 2. As outlined above, the victim performs two requests to `a.com`. The attacker reroutes the second request to IP 2. As both servers are operated by the CDN and share a STEK, the server at IP 2 can decrypt the ticket and resume the session. However, since the server at IP 2 is only configured to host `e.com`, it may ignore the SNI or `HTTP Host` headers that specify `a.com`, and instead forward the request to the attacker-controlled backend of `e.com`.

3.2 Client Authentication Bypass

Our second attack also exploits an isolation issue on the server side. In this case, the victim is a website hosted on the server. The goal of the attacker is to circumvent TLS client authentication.

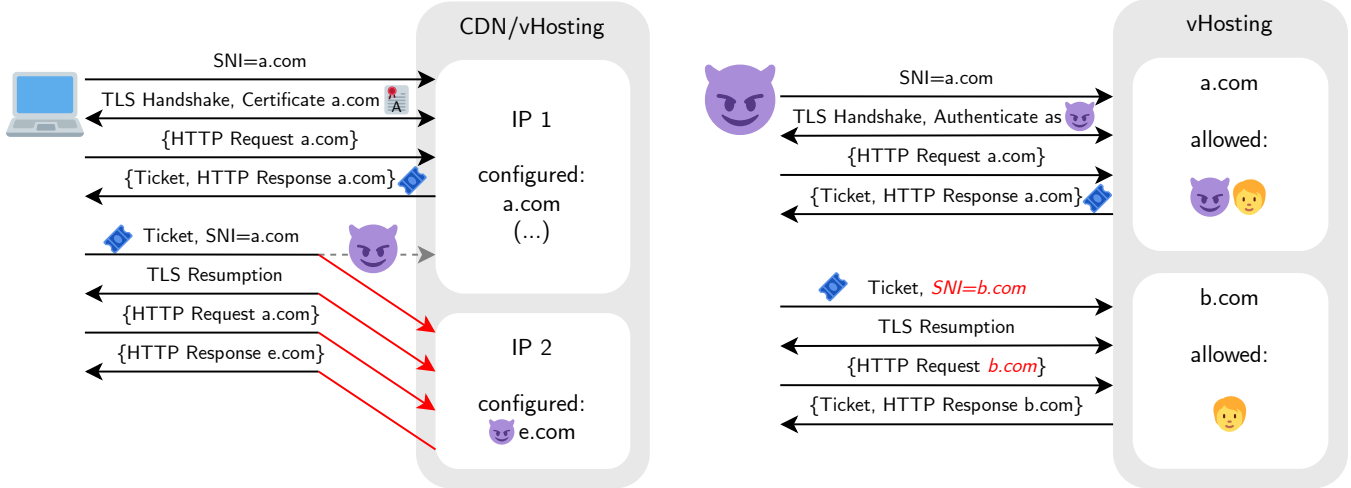
Victim Behavior. The victim’s host has two websites: one that the attacker can access and one that is restricted and inaccessible to the attacker. The attacker-accessible website may or may not use TLS client authentication. The inaccessible website is protected using TLS client authentication.

Attacker Goal and Impact. The attacker wants to visit the inaccessible website, which they should not be able to. This can be classified as a privilege escalation or authentication bypass.

Attacker Capabilities. The attacker can perform TLS connections to the server. If the accessible website is also protected by TLS client authentication, we assume the attacker possesses a certificate to access this website.

Attack Description. We provide an overview of the attack in Figure 2b. The attacker first visits the accessible website and is issued a session ticket by the server. They then attempt to resume the session ticket, and manipulate the SNI and `Host` header (in Figure 2b we show both as `b.com`). If the server accepts and resumes the session ticket, the client certificate authentication is skipped, and the server may serve the inaccessible website to the unauthenticated attacker.

2. As offered by Cloudflare [10].



(a) Server authentication bypass. The CDN/virtual hosting is configured to host `a.com` on IP 1 and the attacker-controlled `e.com` on IP 2. Upon resumption, the attacker reroutes the request to IP 2. As the CDN shares the STEK between all virtual hosts, the ticket can be decrypted on IP 2, causing the CDN to serve `e.com`.

(b) Client authentication bypass. The attacker can access `a.com` using its client certificate, but not `b.com`. The attacker performs a malicious handshake, modifying the SNI extension and HTTP request. This allows them to circumvent the authentication and access `b.com` in a resumption.

Figure 2: Overview of the session ticket confusion attacks.

4 Open-Source Analysis

We analyzed open source server implementations to determine whether they are affected by our proposed session ticket confusion attacks. We started with the server authentication bypass and analyzed the behavior when clients manipulate the SNI and `Host` header in resumed sessions. While this analysis exceeds the attacker’s capabilities in the server authentication bypass model (Section 3.1; the attacker cannot modify the `Host` header nor SNI), it still provides an overview of potential entry points for client authentication bypasses.

We configured each server to host at least two different domains as virtual hosts: the victim’s host and the attacker’s host. We covered cases where either of these two is the default host, as well as the case where a third host is the default host. Each virtual host used its certificate, which only covers their domain. Generally, we configured all virtual hosts to use the same STEK. In our first evaluation, the server authentication bypass, we also evaluated configuring the virtual hosts with different STEKs. During our analysis, we ensured that we covered all resumption combinations: using a ticket from a default host, resuming a ticket at a default host, and resuming a ticket from a non-default host at another non-default host. We evaluated each server in TLS 1.2 and TLS 1.3. We will outline further parameters in the respective sections.

Once we have deployed a server configuration, we retrieved a session ticket for a configured virtual host. We changed the SNI and `Host` header in the resumption attempts to be either the ticket-issuing host I or the resumption host R configured on the server. We also considered omitting the SNI exten-

sion, which we denote as `SNI=None`. We did not omit the `Host` header, as servers expect it and answer with a `400 invalid request` if it is missing.

We chose open source web server implementations with a market share over 10%: Apache, nginx, and OpenLiteSpeed [59]. In addition, we chose Caddy, as it claims easy configuration [63] and was referenced positively in previous works [1, 33, 53]. All implementations support TLS session tickets and virtual hosting. Only OpenLiteSpeed does not support client authentication, but its enterprise version LiteSpeed does.³ We used LiteSpeed when analyzing client authentication with the trial version included in the official Docker image.

We additionally considered *strict* versions of the following two implementations: For Apache, we enabled `SSLStrictSNIVHostCheck`, which we denote as Apache (strict). For nginx, we considered two cases where we defined a default virtual host that handles any hostname otherwise undefined. This host either returns an HTTP 404 error or has the option `ssl_reject_handshake` enabled and uses a unique certificate. We denote these configurations as nginx (strict HTTP) and nginx (strict SNI), respectively. We tested OpenLiteSpeed with and without the `sslstrictSni` option. This option did not impact our results, and therefore, we omit any distinction. OpenLiteSpeed provides an admin web interface that also uses TLS. We considered this an addi-

3. We found contradictory information regarding the support for client authentication in OpenLiteSpeed, and ultimately concluded it is not supported [2].

Table 1: Server behavior when resuming a ticket. All websites were hosted on a single server. I is the ticket issuing host, R is another virtual host. We show whether the server resumes the ticket and, if it is resumed, which virtual host is served. We highlight the column covering the server authentication bypass attacker model (SNI=I, Host=I).

| | | SNI=I | | SNI=R | | SNI=None | |
|---------------------|--------------------|--------|----------|---------------------------------|----------|----------------------------------|-----------------------|
| | | Host=I | Host=R | Host=I | Host=R | Host=I | Host=R |
| Apache | TLS 1.2 | I | 421 | <i>not resumed</i> ^F | | I [†] | 421 [†] |
| | TLS 1.3 | | | | | | R [†] |
| Apache (strict) | TLS 1.2 | I | 421 | <i>not resumed</i> ^F | | I [†] | 421 [†] |
| | TLS 1.3 | | | | | | 403 [†] |
| Caddy | TLS 1.2 TLS 1.3 | I | 421 | 421 | R | <i>not resumed</i> ^{BF} | |
| nginx | TLS 1.2 | I | R | I | R | I | R |
| | TLS 1.3 | | | | | | |
| nginx (strict HTTP) | TLS 1.2 | I | R | I | R | I | R |
| | TLS 1.3 | | | | | | |
| nginx (strict SNI) | TLS 1.2 | I | R | I | R | I | R |
| | TLS 1.3 | | | | | | |
| (Open)LiteSpeed | TLS 1.2 TLS 1.3 | I | R | <i>not resumed</i> ^F | | I [†] | R [†] |

I: Ticket issuer. **R**: (Other) Resumption host. 403/421: HTTP error code. [†]Only resumes the tickets issued by the default host in the config.
^ARaises TLS alert `unrecognized_name` (112). ^BRaises TLS alert `internal_error` (80) if no default host is defined.
^FServer falls back to full handshake, according to standard

tional virtual host for which we attempted to perform a server authentication bypass attack.⁴

4.1 Server Authentication

In addition to the general configuration options outlined previously, we evaluated each implementation in the following scenarios: Multiple domains may be hosted on a single server instance on the same port, on different ports, or on two different server instances. We also considered scenarios where two hosts are configured with unique STEKs.

Table 1 presents the results of our evaluation and shows the content served (if any) within such a manipulated resumption attempt. Note that although the server authentication bypass scenario only allows SNI=I and Host=I (highlighted column), other cases may become relevant when combined with additional attacks or more complex setups.

4.1.1 Results

We found no software vulnerable to circumventing server authentication simply by rerouting traffic to a different server. Hence, all implementations are secure against our server authentication bypass attacker model. However, our extensive tests still uncovered cases where the server resumed session tickets and served a different host. All resumptions occurred when all domains are hosted on one instance on the same port. We found that a single instance using multiple ports

behaves the same as two instances on different IPs. When two instances did not share a host with the same certificate, we could not observe any resumptions, even if the STEK is shared. The results do not change when configuring a unique STEK per virtual host.⁵

Apache properly detects misdirected requests. If the SNI does not match the ticket issuer, Apache rejects the ticket and falls back to a full handshake. Apache further checks that the Host header is set to the same value as the SNI and otherwise returns the HTTP error 421 `Misdirected Request`. If the SNI is omitted, Apache only resumes the tickets of the first configured host (i.e., the default host). However, for these session tickets, Apache behaves inconsistently across TLS versions. In TLS 1.2, it behaves as it does in the SNI=I case; it detects when the host header is set to another host and returns an HTTP error. In TLS 1.3, the result depends on the `SSLstrictSNIVHostCheck` setting. If enabled, requests without an SNI are rejected. If disabled (which it is by default), Apache may serve a different host depending on the Host header.

Caddy always requires the SNI to be present. It resumed all tickets. If the SNI and Host header match, it returns the

4. As the docker image uses a default certificate, we do not consider resumption attempts from one admin interface to another, but just resumptions between the admin interface and a virtual host.

5. (Open)LiteSpeed and Caddy do not support a STEK per virtual host but only one global STEK per server.

content based on these. Otherwise, an HTTP 421 error is returned.

When analyzing nginx, we noticed that nginx uses the same STEK for all virtual hosts. Further, nginx ignores the SNI and only relies on the `Host` header to determine which content to serve. The only case where nginx did not resume a session was in the nginx (strict SNI) configuration, where a missing SNI led to a TLS alert.

We found that OpenLiteSpeed only resumes tickets if the SNI during resumption matches the SNI from the ticket issuance. However, OpenLiteSpeed determines the served content based on the `Host` header. If the client specifies another virtual host in the `Host` header, OpenLiteSpeed returns the content for that virtual host. We could not resume tickets at or from the admin interface in this scenario. We observed LiteSpeed to behave identically to its open-source counterpart.

4.2 Client Authentication

As we did not observe session ticket resumptions across different virtual hosts when they were hosted on multiple instances in [Section 4.1](#), we chose to limit the setups to a single server instance hosting all virtual hosts on a single port on the same instance. We assumed one virtual host to be accessible to the attacker, the initial host *I*, which they use to obtain session tickets. We evaluated scenarios where this host does not require client authentication (“*I* is freely accessible”) and where it does (“*I* requires client authentication”). The attacker has a valid client certificate for this host, if required. The other virtual host *R* is protected with client authentication, and the attacker has no access to a certificate to authenticate against this host. If both hosts require client authentication, both hosts use distinct Certificate Authorities for client authentication.

4.2.1 Results

We found multiple cases where we could circumvent client authentication. Most required the ticket-issuing host to use client authentication, thus classifying the vulnerability as a privilege escalation. Only in LiteSpeed, we could bypass authentication altogether without needing any prior authentication. We summarize the results in [Table 2](#).

As expected, all web servers accept the resumption when the SNI matches the initial request. When the `Host` header also matched the initial domain, all servers replied with the content of the initial website. When the `Host` header was set to the domain of the secured resumption host, all servers except LiteSpeed replied with the HTTP status code 421. This indicates that web servers have checks implemented that connect the authentication in the TLS layer to the request on the HTTP layer.

Unprivileged Access. LiteSpeed is the only server in our test that allowed us to access the restricted content without

any prior authentication. After investigation, we found that LiteSpeed is vulnerable without using tickets. Changing the `Host` header in the initial connection yields the protected host as well.

Privilege Escalation. For all other servers, we found vulnerabilities requiring the attacker to use client authentication at the accessible host. Note that the attacker does not possess a certificate valid for the inaccessible host. The combination of SNI and `Host` header necessary for the attack to work varies between the servers.

Apache incorrectly allowed access to the second site when the client omitted the SNI in the resumption handshake and then requested the target site’s domain in the `Host` header. This only worked if the accessible host was the default host (the first in the config) and only in TLS 1.3. In TLS 1.2, the server returned an HTTP 421 status code. The strict config variant prevented this, and the server responded with an HTTP 403 instead.

Caddy is vulnerable if both the SNI and `Host` header within the resumption are set to the inaccessible domain. Note that Caddy has a default setting to prevent changing the `Host` header to bypass client authentication [51], but it seemingly does not consider session resumption.

nginx is vulnerable if the SNI in the resumption is set to the inaccessible domain or omitted entirely, and the `Host` header is set to the inaccessible domain. The attack only worked for TLS 1.3 resumptions. Using nginx (strict SNI) only prevented the resumption if the SNI was missing. Since the vulnerability also occurred if the SNI is set to the domain of the target site, this did not prevent the attack completely.

4.3 Inconsistent Behavior

Apart from the immediate vulnerabilities, we observed inconsistent behavior between and within web servers.

Between Web Servers. No two web servers behaved the same within our tests. We found that web servers did not agree on when to resume session tickets. When changing the SNI to the inaccessible host, Apache and LiteSpeed rejected the ticket. Caddy rejected the session ticket in some cases, depending on whether the ticket was issued from another host using client authentication. Meanwhile, nginx resumed nearly all tickets except in one configuration with TLS 1.3.

Within Web Servers. More notable are the inconsistencies within single web servers. We expected servers to behave the same, independent of TLS version or whether the accessible host uses client authentication.

However, we observed inconsistencies between TLS 1.2 and 1.3. Interestingly, we found that all servers are vulnerable in TLS 1.3, but not all are vulnerable in TLS 1.2. We further found all web servers except LiteSpeed to show different

Table 2: Server behavior when attempting to resume a ticket from I with different combinations of SNI and Host header. All websites were hosted on a single server on a distinct domain. R always requires client authentication. The configuration should allow the attacker to access only I but not R.

| | | I is freely accessible | | | | | | I requires client authentication | | | | | |
|---------------------|--------------------|------------------------|----------|---------------------------------|------------|-------------------------|--|----------------------------------|----------|---------------------------------|-----------------|--|---|
| | | SNI=I | | SNI=R | | SNI=None | | SNI=I | | SNI=R | | SNI=None | |
| | | H=I | H=R | H=I | H=R | H=I | H=R | H=I | H=R | H=I | H=R | H=I | H=R |
| Apache | TLS 1.2 TLS 1.3 | I | 421 | <i>not resumed</i> ^F | | I [†] | 421 [†] 403 [†] | I | 421 | <i>not resumed</i> ^F | | I [†] | 421 [†] R [†] |
| Apache (strict) | TLS 1.2 TLS 1.3 | I | 421 | <i>not resumed</i> ^F | | I [†] | 421 [†] 403 [†] | I | 421 | <i>not resumed</i> ^F | | I [†] | 421 [†] 403 [†] |
| Caddy | TLS 1.2 TLS 1.3 | I | 421 | <i>not resumed</i> ^F | | 421 ^B 421 | 421 ^B 421 | I | 421 | 421 | R | <i>not resumed</i> ^B ^F | |
| nginx | TLS 1.2 TLS 1.3 | I | 421 | I | 421 400 | I | 421 400 | I | 421 | I | 421 R | I | 421 R |
| nginx (strict HTTP) | TLS 1.2 TLS 1.3 | I | 421 | I | 421 400 | I | 421 400 | I | 421 | I | 421 R | I | 421 R |
| nginx (strict SNI) | TLS 1.2 TLS 1.3 | I | 421 | I | 421 400 | I | 421 <i>not resumed</i> ^A | I | 421 | I | 421 R | I | 421 <i>not resumed</i> ^A |
| LiteSpeed* | TLS 1.2 TLS 1.3 | I | R | <i>not resumed</i> ^F | | I [†] | R [†] | I | R | <i>not resumed</i> ^F | | I [†] | R [†] |

I: Ticket issuer.

R: Other resumption host requiring client authentication.

400/403/421: HTTP error code.

[†]Only resumes tickets issued by the default host in the config.

*Behavior is not session resumption specific, changing the `Host` header in a normal connection also leaks R.

^ARaises TLS alert `unrecognized_name` (112).

^BRaises TLS alert `internal_error` (80) if no default host is defined.

^FServer falls back to full handshake, according to standard

behavior in different TLS versions. Caddy seemingly uses separate SNI logic in TLS 1.2 and will raise a TLS alert `internal_error` if no fallback domain was provided. Further, nginx (strict SNI) did not resume tickets in TLS 1.3 when the SNI was omitted, but did so in TLS 1.2.

All servers except LiteSpeed behaved differently depending on whether the accessible host required client authentication. Apache, Caddy, and nginx only seemed to verify whether a client was authenticated during the initial connection, but not which certificate was used.

5 Real-World Evaluation

To evaluate the impact of session ticket confusion, we performed a large-scale scan of deployed servers. We only evaluated the server authentication bypass on a large scale. Scanning for client authentication bypasses is not feasible on a large scale, as popular public servers rarely use client authentication. Further, our open source analysis revealed that findings are more likely if the attacker has prior privileges, which we do not possess. Nonetheless, we evaluated specific CDN services manually.

5.1 Server Authentication Scan

5.1.1 Scan Methodology

To recreate the attacker scenario, we would need to buy enterprise-level features with dedicated IPs at each virtual hosting provider. However, as this is infeasible on a large scale, we aim to find sites that could already perform the server authentication bypass. That is, we want to find servers that resume tickets but serve content different from that of the ticket issuer. Concretely, we want to find a domain hosted on one IP address and resume the ticket at a different IP address. To find such servers, we first create groups of servers we presume are likely to resume tickets. We then request a domain from the first server, retrieve a ticket, and attempt to resume it on the second server. We evaluate whether the second response matches the requested domain or whether the server responded with the content for another domain, indicating that the request was misrouted to a third party. We perform the whole process in parallel with TLS 1.2 and TLS 1.3 because of the inconsistencies we observed in Section 4.1.

Creating Candidates. We aim to evaluate popular websites, which often use CDNs to accelerate page load times. To this end, we use the Tranco Top 1M list [34] as a basis. We obtain the IP addresses for the domains on that list and probe whether

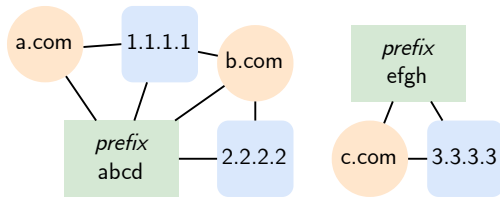


Figure 3: Exemplary grouping of domains. `a.com` and `b.com` use a common session ticket prefix, and both are hosted on `1.1.1.1`. `b.com` is additionally hosted on `2.2.2.2`. `c.com` uses a distinct prefix and IP. We would only attempt to resume a ticket from `a.com@1.1.1.1` at `2.2.2.2`, but no other combinations in this example.

the servers support TLS and session tickets by sending an HTTPS request. Additionally, we deepened the exploration by also considering related domains listed in the SANs of the certificates encountered during our scan. This increases the number of domains potentially sharing technical infrastructure and possibly their STEK. We iteratively repeated this process until no further domains were found. During this process, we store the tickets we observed in all requests.

We group the domains and IP addresses based on the retrieved tickets. Most implementations include an identifier of the used STEK at the beginning of the ticket. This keyname is commonly 16 or 4 bytes long [28]. We use the first four bytes to group servers that may use the same STEK and might allow resumption.

This still results in too many possible candidates to evaluate exhaustively.⁶ Therefore, we chose to use a sampling-based approach. For each IP a domain resolves to, we propose up to 10 IPv4 and up to 10 IPv6 addresses to resume a ticket. These resumption IP addresses must share the ticket prefix in the TLS version, and the domain in question must not be hosted on these IP addresses. We instruct our database to return the IP addresses randomly to decrease the chance of using only similar IP addresses.

We provide an example of selecting target addresses in Figure 3. In the example, we would attempt to resume a ticket from `a.com@1.1.1.1` at `2.2.2.2`, as it might fallback to serving a different content. We would not attempt to resume a ticket from `b.com` at `1.1.1.1` nor `2.2.2.2`, as we would expect both IP addresses to properly handle `b.com` in the resumption. As `c.com` does not share the prefix with another domain, we would not attempt to resume a ticket at another address.

Attempting Resumption. Each candidate in the sampling approach consists of a domain, an IP that issues the ticket, and a set of IPs at which we attempt to resume the ticket. We emulate the attack by performing an HTTP GET / request with the domain included as the SNI and Host header to the server to retrieve the ticket and HTTP response. This equals

the user visiting the website. For each resumption IP, we attempt to resume the ticket and send the same HTTP GET / request. This equals the attacker redirecting a subsequent connection of the client. We store whether the ticket was resumed and the HTTP response.

Evaluating the Responses. We classify whether a server is vulnerable based on the response to our resumption attempt. If a ticket is rejected, the attempt is considered secure. Similarly, if the HTTP request is rejected, for example, with a status code like 421 *Misdirected Request*, we consider it secure. We also assume a resumption attempt to be secure when the content served on resumption remains exactly identical, or if both contain redirects to the same destination. We do not analyze attempts where either the initial or the resumption connection did not retrieve an HTTP body (e.g., only one is a redirect, a 5xx error was returned).

For the remaining attempts, where the session ticket was accepted, and the server performed an abbreviated handshake, we compare the returned HTTP bodies. In case the resumption body is similar to the initial body, we can consider the resumption as likely secure. If the body differs, we cannot automatically consider this behavior vulnerable, as some websites may serve different content on each load (e.g., CSRF tokens in HTML, dynamic content). Therefore, we aim to find the origin of the returned content. To this end, we compare the returned body with other bodies observed in domains hosted on the resumption IP. To compare the bodies, we chose the normalized Levenshtein ratio once over the whole HTML document and once only over the HTML header. We look for low similarity between the initial and resumed page and high similarity between the resumed and another proposed origin. If the proposed origin uses the same certificate as in the initial connection, we consider the resumption secure. For other origins, we manually review the resumption. During this, we cluster the resulting candidates by CDN or Autonomous System (AS) to aid the process.

Implementation. We implemented our scanning methodology using tools from the ZMap Project. Concretely, we used ZDNS [31] to resolve DNS entries to IPv4 and IPv6 addresses. We use ZMap [16] with IPv6 support⁷ to evaluate whether the default TLS port is open on a web server. We use ZGrab2 [15] with TLS 1.3 support⁸ to perform TLS handshakes and send HTTP requests to the servers.

6. We also checked this for 16B keynames, and this resulted in a similar number of candidates. We chose 4B to cover more implementations.

7. We used a forked ZMap for IPv6 support, see <https://github.com/UPB-SysSec/zmap/>

8. We used a forked ZGrab2 for TLS 1.3 and better session ticket support, see <https://github.com/UPB-SysSec/zgrab2/>

Table 3: Number of domains and evaluated sample pairs. In the end, we manually reviewed 4,370 resumptions.

| Proposed Domains | 2,063,760 | |
|------------------------------------|------------|-------|
| Resolved to IP(s) | 1,844,289 | 89.4% |
| Open :443 | 1,463,917 | 70.9% |
| ... TLS | 1,169,527 | 79.9% |
| ... TLS and issues session ticket | 1,108,322 | 94.8% |
| Total pairs sampled | 59,752,483 | |
| Classified Safe | 31,709,491 | 53.1% |
| ... Identical Redirect | 14,535,697 | 45.8% |
| ... No Resumption | 8,707,542 | 27.5% |
| ... Identical Body | 8,449,048 | 26.6% |
| ... Redirect Detected / Not Routed | 17,204 | 0.1% |
| Classified Not Applicable | 20,824,364 | 34.9% |
| ... Initial Request Error | 12,519,768 | 60.1% |
| ... Resumption Access Denied (403) | 7,580,628 | 36.4% |
| Classified for Similarity Analysis | 7,218,628 | 12.1% |
| ... Manual Review | 4,370 | 0.1% |
| – Cloudflare SaaS | 4,033 | |
| – inconclusive | 161 | |
| – vulnerable | 176 | |

5.1.2 Scan Results

We performed our large-scale evaluation from December 2024 to January 2025 using the Tranco list [34] generated on December 5th.⁹ The numbers of gathered hosts, sampled pairs as well as their classification are shown in Table 3. The threshold for manual review was a maximum similarity of 0.6 between the initial HTML and the resumed HTML and a minimum of 0.9 between the resumed HTML and the HTML of the assumed origin. We clustered the results to review by AS of the involved IP addresses.

We uncovered several instances of vulnerable session resumptions spanning multiple ASes. In one notable cluster, ten ASes, operated by different companies, were found to allow resumptions across ASes. Despite the organizational differences, all these companies utilized DDoS-Guard’s¹⁰ DDoS protection, strongly suggesting that the root cause lies within DDoS-Guard’s service. Additionally, further vulnerable resumptions were observed within this cluster, which were not considered clearly vulnerable. One additional AS that initially appeared unconnected was also determined on further investigation to use DDoS-Guard’s services, reinforcing the hypothesis of a centralized issue. A second, smaller cluster involving four ASes across three companies revealed resumptions with varying degrees of vulnerability. Two of these ASes appeared to leverage DDoS protection services from Variti,¹¹

the third company, suggesting another potentially misconfigured provider. Beyond these clusters, we also discovered isolated resumptions within three independent ASes, with no apparent ties to known CDN providers.

Furthermore, one instance of a technically vulnerable resumption was observed within West Texas A&M University’s network, although all sessions remained confined to the same organization. Several cases remained inconclusive due to insufficient information. This included 106 resumptions across 27 ASes, most located in China. One apparent false positive was identified in the Verizon Business AS, and additional false positives were observed in the networks of Google and Akamai, likely stemming from temporary error pages.

In a preliminary scan [54], we additionally discovered that Fastly’s CDN service would resume tickets regardless of SNI, but serve content based on SNI. By abusing their dedicated IP offering [20], meant for legacy non-SNI traffic, we were able to fully enact the server authentication bypass attack. Due to the critical impact of this finding, we reported this to Fastly directly in our pre-scan phase in September 2023, and they fixed the issue by binding tickets to the issuing certificate (see Section 7).

Findings Unrelated to Tickets. Most of our findings were attributed to Cloudflare’s network. Upon closer review, we determined that this issue was unrelated to session resumption and had a broader authentication impact. Cloudflare allows specific website providers, so-called *SaaS providers*, to use dedicated IPs [11]. Any traffic to these IPs is TLS-terminated by Cloudflare, and the request is forwarded to the backend for the SaaS providers. This is intended to be used, such that customers of the SaaS provider can register their domain to point to this IP address. It is then the SaaS provider’s job to serve the correct content.

In our scan results, we found that when requesting any domain registered at Cloudflare on a SaaS IP, we received the correct certificate for the domain. However, the server responded with an error page from the SaaS provider, suggesting that the request had been internally forwarded to the SaaS provider. This allows the SaaS provider to read the full request and answer with any content (e.g., an XSS attack payload), allowing them to perform a full TLS-MitM. We disclosed this vulnerability to Cloudflare. According to Cloudflare, this is part of a deprecated version that is only available to existing customers with an enterprise contract. They confirmed that the request was routed to the SaaS provider, theoretically enabling the SaaS provider to perform a TLS-MitM attack. Although this vulnerability was unrelated to our attack model and falls outside the scope of session resumption, it highlights broader security challenges and suggests that our methodology may have applicability beyond its intended purpose.

9. Available at <https://tranco-list.eu/list/V9V2N>.

10. <https://ddos-guard.net/>

11. <https://variti.io>

5.2 Client Authentication Bypass

We performed an evaluation of three free CDN services to determine whether they are susceptible to client authentication bypasses using session resumption. We analyzed Google’s Load Balancer [26], the Azure App Gateway [36], and Cloudflare’s API Shield [12].

For the Google Cloud Load Balancer, we found that session tickets are disabled when enabling client authentication. The Azure App Gateway supports session resumption with client authentication. However, we could not resume tickets across two configured gateways.

At Cloudflare, we used two accounts to set up client authentication (“mTLS”) for two distinct domains. Using the same attacker model as in Section 4.2, we authenticated against the first domain (I) and were issued a ticket. We ran a set of tests from Table 2, intending to access the second domain (R). Our evaluation revealed that we could resume the ticket specifying SNI=R and Host=R, and were served the supposedly protected content (R). We disclosed this to Cloudflare developers using their bug bounty program, who disabled session resumption when using mTLS [7].

5.3 Limitations

Our large-scale evaluation of server authentication is constrained by several factors that may result in us underestimating the true scale of the vulnerabilities. First, we employed random sampling, which may have caused vulnerable pairs to be excluded by chance. Second, we assume that the keyname exists, is located at the beginning of a ticket, and that the same STEK is always accompanied by the same keyname. This may not always hold, potentially leading to further overlooked cases. Finally, our similarity score methodology may produce false negatives, further limiting the comprehensiveness of our findings. Together, these factors suggest that our analysis presents only a lower bound on the actual prevalence of the vulnerabilities. However, note that we only need to identify some and not all cases to determine if a cluster is vulnerable.

The evaluation of client authentication in CDNs was not meant to be exhaustive. We only considered a few CDNs where we could easily find documentation on client authentication. We did not consider CDNs where client authentication is a paid feature (e.g., Fastly [19]), or the documentation indicates secure behavior (e.g., AWS ALB [3]: session resumption is not supported in combination with client authentication). We performed our analysis manually and only evaluated a single mTLS configuration for each CDN. However, CDNs are complex and allow for a variety of configurations, and may offer multiple services that use client authentication. Due to the complexity of CDNs (cf. [4]), we reckon a deeper analysis may uncover vulnerable configurations.

6 Discussion

The interaction between TLS session resumption and hostnames has been a topic of some consideration in existing standards, but our findings highlight significant gaps that compromise security in modern virtual hosting and CDN environments. RFC 6066 for TLS 1.2 mandates that when an SNI is provided, servers must not accept session resumption under a different hostname than the initial SNI [17]. In TLS 1.3, the explicit linkage between SNIs and sessions is removed. Instead, resumption is tied to the certificate presented during the initial handshake, though a “performance optimization” proposed only to match SNIs [48, Section 4.6.1]. However, this requirement is explicitly placed on the client, overlooking scenarios in which an attacker redirects requests without the client’s knowledge or the client being malicious itself. Additionally, the standard completely relieves the server from any responsibility in validating session resumption, as “there is no need for the server to associate an SNI value with the ticket.” [48, Section 4.2.11] Furthermore, the standard claims that “normally, there is no reason to expect that different servers covered by a single certificate would be able to accept each other’s tickets.” [48, Section 4.6.1] However, in modern CDN and virtual hosting environments, servers may accept each other’s tickets even when multiple certificates are involved, potentially creating a gap between theoretical expectations and practical realities. All in all, the attacks we presented are not considered in the standard. The differences in the TLS 1.2 and TLS 1.3 standards also partly explain the observed behavior inconsistencies described in Section 4.3.

Another contributing factor is the intersection of TLS and HTTP standards, which operate at different layers but are intrinsically linked in practice. TLS uses the SNI for hostname indication, while HTTP relies on the `Host` header. Our open source analysis revealed that servers sometimes fail to validate that these two hostname indications match, even though TLS only provides authentication for the SNI hostname. This discrepancy can result in broken authentication, allowing attackers to exploit the mismatch. Additionally, RFC 8446 mandates that only the resumption SNI be exposed to application layers [48, Section 4.6.1]. This underscores the need for TLS to ensure that authentication is maintained during session resumption. Still, this resumption SNI should be matched against the HTTP `Host` header so authenticity is achieved throughout the whole request, not only within TLS.

Lastly, our attacks exploit misconfigurations in virtual hosting environments, which, while preventable through proper configuration, are a persistent and well-documented issue. Web server misconfiguration has been a factor in previous security breaches (e.g., [33]), and our server authentication bypass analysis shows that it continues to exist in the wild. Specifically, the concept of a “default host” in virtual hosting environments, where any unconfigured hostname defaults to a generic handler, is particularly critical. Large CDN providers

must carefully evaluate how misdirected or improperly routed requests are managed to avoid enabling authentication bypass or other vulnerabilities.

7 Countermeasures

The vulnerabilities identified in this paper stem from the inconsistent isolation of virtual hosts following TLS session resumption. Addressing these issues requires guarantees that the identities asserted during the initial handshake remain consistent throughout resumed sessions. A robust solution involves binding session tickets to all exchanged certificates and asserted identities from the initial handshake. Upon resumption, the server must ensure that these certificates and identities remain valid. Concretely, we recommend distinct countermeasures for the server and client authentication bypasses.

Server Authentication. We recommend binding the ticket to the server certificate chain. This ensures that the identity the client assumed during the initial handshake remains valid. If the certificate changes at the server, the server should not resume the session; instead, it should perform a full handshake, presenting the new certificate to the client. This allows the client to inspect the new certificate and decide whether they accept it.

Client Authentication. We recommend the server to store the certificate presented by the client in the session ticket, as recommended by the standard [18]. This way, the server can revalidate the certificate in a resumption handshake, ensuring the authentication is still valid for the resumption.

Alternatively, the server could bind the ticket to the rules that the certificate passed initially. If the server expects different rules to pass upon resumption, the server can fall back to a full handshake. This approach may be easier to implement, but has worse performance for resumptions where the rules differ.

Considered Alternatives. We also considered whether binding the ticket to the SNI is a viable alternative. While it can prevent some client authentication bypasses we observed (Table 2), fallback mechanisms in multi-server scenarios may cause authentication to be bypassed, as observed in the server authentication bypasses during the large-scale evaluation. Further, session resumption across hostnames may be used intentionally within the same certificate [56].

We also considered how binding to the server certificate interacts with certificates spanning multiple domains (A and B). In this case, our attack is not necessary; the attacker can reroute the client (requesting A) to a different server using the same certificate. If that second server is misconfigured to fallback to another virtual host (B), it will serve the certificate valid for both domains (A and B), which the client accepts.

Implementation. While the countermeasures seem simple to implement, they require a good understanding of the TLS library used. We explored how to implement our recommendations using OpenSSL 3.5 [41]. OpenSSL offers the concept of a session context [43]. A session context is an arbitrary 32-byte string. Tickets are bound to this context. That is, a ticket issued in one context can only be resumed in the same context and will be rejected otherwise. Servers must already set this if they wish to use client authentication and session resumption. We recommend using this to also fix the server authentication bypasses. To this end, we recommend including the server certificate in the session context.

For client authentication, OpenSSL provides functions to specify a list of certificates against which the client certificate is validated, and to specify a callback function [45]. Upon resumption, the certificate is not checked again, even if a different list of certificates to validate against is specified. Therefore, we recommend including the list of certificates in the session context too. Similarly, the callback is not called in resumptions. Therefore, implementations cannot rely on the callback to reject clients from accessing certain virtual hosts.

Implementing this has some caveats. The session context must be set before OpenSSL decides whether to resume the session. This means the `ClientHello` callback [42] must be used, and not the SNI callback [44]. Within the `ClientHello` callback, the context should be set as outlined above. As the context is limited in size, we recommend hashing all data and using the hash as the context. Furthermore, we recommend setting the context on the `SSL` object rather than the `SSL_CTX` object, as multiple contexts on a single socket may not function as expected.

8 Related Work

Confusion Attacks. Although we specifically focus on session tickets, host confusion attacks involving both HTTP and TLS are nothing new: Delignat-Lavaud and Bhargavan [14] originally formalized virtual host confusion attacks in 2015, where they present a number of attacks abusing insecure fallback hosts, shared TLS caches, or HTTP connection reuse. In particular, they highlight the difficulties of systematic evaluation and perform most analyses manually. In 2021, Brinkmann et al. [6] presented cross-protocol attacks that exploited certificates shared between services. Shared certificates have also been exploited between distinct subdomains using HTTP headers: Zhang et al. [64] described how insecure servers with a shared certificate can downgrade HTTPS connections or circumvent HTTP Strict Transport Security.

CDNs. CDNs require a lot of trust, as they act as a MitM between the content owner and client and often require the private key of the content owner. Liang et al. analyzed 20 CDNs HTTPS implementations and propose solutions to sharing private keys. Ghaznavi et al. [25] categorized attacks against

CDNs into attacks on edge servers, routing, and origin servers. According to them, previous work on attacking CDN routing has largely focused on identifying routing infrastructure and denial of service. Although virtual host confusion has been explored within CDNs, it was based on weaknesses in HTTP request and header parsing [8]. Similar attacks that, through misconfiguration, lead to CDNs serving malicious content or leaking user traffic, include web cache poisoning [32] or deception [37, 38].

Session Resumption. Session tickets weaken TLS by reusing session secrets. This undermines Perfect Forward Secrecy (PFS), leading to repeated criticism in the past [57, 60] with PSKs being favored instead [58]. In 2016, Springall, Durumeric, and Halderman [53] showed that the concerns about PFS were dangerous in practice, finding that 10% of domains in the Alexa Top Million would re-use STEKs for at least 30 days. They also examined STEK sharing across different domains, identifying a group of 62k domains belonging to Cloudflare CDN. Although they discuss best practices, they did not attempt to exploit STEK sharing. In 2018, Sy et al. [55] showed that with a session duration of seven days (as recommended by the TLS 1.3 draft at the time), it was possible to track 65% of users in their dataset across different websites by observing the corresponding. In 2020, Sy et al. [56] proposed to encourage and advertise STEK sharing across different SNIs, using a newly designed extension. Although this proposes a way to safely implement STEK sharing, potentially preventing the attacks we described, it has not been adapted to any standard.

Session tickets have also been used as new attack vectors: In 2016, Filippo Valsorda [21] presented a vulnerability dubbed *Ticketbleed* in F5’s TLS implementation, which allowed attackers to extract uninitialized memory from a server, akin to the infamous *Heartbleed* bug [27]. In 2023, Hebrok et al. [28] examined the security of session tickets, identifying cryptographic implementation issues within ticket handling. Notably, they discovered a bug within Amazon AWS, causing all-zero STEKs to be used. In 2024, Radoy, Hebrok, and Somorovsky [47] used session tickets as a side channel for a Partitioning Oracle Attack against AES-GCM to obtain the underlying STEK and therefore decrypt all current session tickets.

Client Authentication. Research on TLS client authentication is sparse: Parsovs [46] presented usability and security related implementation issues in 2014. In recent years, Wachs, Scheitle, and Carle [61] and Foppe et al. [22] showcased the potential for tracking users with client certificates. A large-scale study performed by Xia et al. [62] showed that common issues within X.509 certificates themselves were also prevalent in client authentication. To our knowledge, we are the first to attempt to actively circumvent client authentication using application-level inconsistencies.

9 Conclusions

Our investigation into session resumption and its implications on TLS authentication guarantees revealed significant vulnerabilities. Addressing our first research question, we explored attack models that exploit TLS session resumption to undermine authentication guarantees. Specifically, we introduced novel attacks capable of bypassing client authentication and revisited attacks on server authentication in the context of session tickets and shared STEKs. Our findings underscore that eliminating certificate exchanges during session resumption significantly complicates authentication by requiring authentication contexts to persist across connections. This reveals a fundamental challenge in maintaining robust security guarantees without a full handshake. In response to our second research question, we confirmed that STEK-based vulnerabilities translate to real-world implementations and virtual environments. Our analysis uncovered authentication vulnerabilities in popular servers such as nginx and Apache, among others, and exposed faulty implementations in CDNs.

We discussed potential causes within the standards that fail to account for the attack scenarios we identified. Notably, our evaluation revealed that servers implementing TLS 1.3 exhibited higher susceptibility to these issues compared to TLS 1.2. This discrepancy may stem from explicit assurances in the TLS 1.3 standard that does not mandate binding session tickets to the SNI and partly reflects in our analysis results. Given that the TLS 1.3 standard is finalized and widely deployed, retroactive changes to the protocol are unlikely to be feasible. However, this knowledge informs future standards to address these overlooked vulnerabilities.

We looked at the OpenSSL APIs and evaluated how to implement virtual hosting securely. In our testing, we found the APIs to be easy to misuse, exhibiting unintuitive behavior when sessions are resumed, and the wrong usage of the API does not throw errors. We urge library maintainers to implement misuse-resistant APIs.

Future research could extend this work by investigating the broader authentication guarantees of TLS, particularly in scenarios involving resumption across hostnames. A longitudinal study of misconfigurations and their evolution over time would also provide valuable insights into mitigating these vulnerabilities. Further, an analysis of STEK sharing based on location or service providers may allow insights into network infrastructure. We further propose researching the virtual hosting implementations of CDNs and the interaction with TLS.

Acknowledgments

Sven Hebrok was supported by the research project “North-Rhine Westphalian Experts in Research on Digitalization (NERD II)”, sponsored by the state of North Rhine-Westphalia – NERD II 005-2201-0014. Felix Cramer and

Tim Storm were supported by the “PRISMA” program of the Computer Science Institute at Paderborn University. **Figures 1** and **2** contain graphics from Twemoji¹², licensed under CC BY 4.0, and from latex-twemojis¹³.

Ethical considerations

Ethical Scan Guidelines. For our network-wide research scans, we strictly adhere to established ethical guidelines to ensure that our research is responsible and respectful of the broader internet community [16]. We maintain a block list of IP ranges that have requested to be excluded from our research activities. We set up reverse DNS entries and provide an information website on our scanning VM including contact details and information about opting out, allowing network operators to easily identify our scanning activities and understand their purpose. Our scans were registered with our ISP, any abuse reports were forwarded to and handled by us in a timely manner. To minimize the impact on networks, our scanning activities are designed to be of low intensity and are spread over time. This prevents network disruption or congestion. Additionally, only valid TLS traffic is generated. By adhering to these ethical guidelines, we aim to conduct our research in a manner that is both scientifically valuable and considerate of the rights and responsibilities of network operators.

Responsible Disclosure. We have disclosed all identified vulnerabilities to the respective developers and/or organizations. CVEs were assigned for the findings in Apache (CVE-2025-23048) and nginx (CVE-2025-23419). LiteSpeed has fixed the issue, and Caddy has disabled session tickets when using client authentication. Fastly binds the tickets to the certificate, and Cloudflare, as a preliminary fix, disabled session tickets when using client authentication. We are in the disclosure process with the remaining identified vendors. We have communicated our intention to publish these findings, including the planned publication date, ensuring that all stakeholders are informed and have sufficient time to implement fixes.

Availability and Open Science

We are committed to advancing open science by sharing the tools and resources used in this research. To this end, we make our toolset available through GitHub repositories and provide a snapshot of the tools, as used in this paper, on Zenodo.¹⁴ The shared toolset includes:

- Tools to reproduce our open source software analysis (**Section 4**).^{15,16}
- Tools necessary to perform our large-scale evaluation (**Section 5**).¹⁷ This includes our forks of ZMap,¹⁸ which

adds IPv6 support, and ZGrab2,^{19,20} which adds TLS 1.3 and better session ticket support.

We do not publicly share the full database due to several constraints:

- The database may contain additional false negatives that are potentially vulnerable and require ethically responsible handling.
- Portions of the data may include copyrighted content, which we are ethically bound not to republish.
- The database’s size prevents us from using long-term archiving services like Zenodo. The proprietary database files are approximately 240 GiB for MongoDB and 332 GiB for Neo4J.

Finally, we will actively support the AEC in evaluating the functionality and reproducibility of our artifacts.

References

- [1] Josh Aas, Richard Barnes, Benton Case, Zakir Durumeric, Peter Eckersley, Alan Flores-López, J. Alex Halderman, Jacob Hoffman-Andrews, James Kasten, Eric Rescorla, Seth Schoen, and Brad Warren. “Let’s Encrypt: An Automated Certificate Authority to Encrypt the Entire Web”. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’19. New York, NY, USA: Association for Computing Machinery, Nov. 2019, pp. 2473–2487. ISBN: 978-1-4503-6747-9. DOI: [10.1145/3319535.3363192](https://doi.org/10.1145/3319535.3363192).
- [2] Michael Alegre. *lswsdocs_config_vhost_ssl – Client Verification*. URL: <https://www.litespeedtech.com/docs/web-server/config/virtual-host-ssl#clientVerify> (visited on 05/08/2025).
- [3] Amazon. *Mutual authentication with TLS in Application Load Balancer - Elastic Load Balancing*. URL: <https://docs.aws.amazon.com/elasticloadbalancing/latest/application/mutual-authentication.html> (visited on 05/08/2025).
- [4] Vaisha Bernard. *From Simulation to Tenant Takeover*. Dec. 2024. URL: <https://media.ccc.de/v/38c3-from-simulation-to-tenant-takeover> (visited on 04/29/2025).
- [5] Simon Blake-Wilson, Jan Mikkelsen, Magnus Nyström, David Hopwood, and Tim Wright. *Transport Layer Security (TLS) Extensions*. Request for Comments. June 2003. DOI: [10.17487/RFC3546](https://doi.org/10.17487/RFC3546).

12. <https://github.com/twitter/twemoji>

13. <https://gitlab.com/rossel.jost/latex-twemojis/>

14. <https://doi.org/10.5281/zenodo.15474656>

15. <https://github.com/UPB-SysSec/stekruebe-serve-r-tests>

16. <https://github.com/UPB-SysSec/stekruebe-auth-dotsche>

17. <https://github.com/UPB-SysSec/stekruebe-pipeline>

18. <https://github.com/UPB-SysSec/zmap>

19. <https://github.com/UPB-SysSec/zgrab2>

20. <https://github.com/UPB-SysSec/zcrypto>

- [6] Marcus Brinkmann, Christian Dresen, Robert Merget, Damian Poddebniak, Jens Müller, Juraj Somorovsky, Jörg Schwenk, and Sebastian Schinzel. “ALPACA: Application Layer Protocol Confusion - Analyzing and Mitigating Cracks in TLS Authentication”. In: *30th USENIX Security Symposium (USENIX Security 21)*. 2021, pp. 4293–4310. ISBN: 978-1-939133-24-3. URL: <https://www.usenix.org/conference/usenixsecurity21/presentation/brinkmann>.
- [7] Matt Bullock, Rushil Mehra, and Alessandro Ghedini. *Resolving a Mutual TLS session resumption vulnerability*. Feb. 2025. URL: <https://blog.cloudflare.com/resolving-a-mutual-tls-session-resumption-vulnerability/> (visited on 04/29/2025).
- [8] Jianjun Chen, Jian Jiang, Haixin Duan, Nicholas Weaver, Tao Wan, and Vern Paxson. “Host of Troubles: Multiple Host Ambiguities in HTTP Implementations”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. Vienna Austria: ACM, Oct. 2016, pp. 1516–1527. ISBN: 978-1-4503-4139-4. DOI: 10.1145/2976749.2978394.
- [9] Cloudflare. *Adoption & Usage Worldwide | Cloudflare Radar*. 2025. URL: <https://radar.cloudflare.com/adoption-and-usage?dateRange=28d#tls-12-vs-tls-13-vs-qtls> (visited on 01/20/2025).
- [10] Cloudflare. *Browser compatibility - Non-SNI support*. Feb. 2025. URL: <https://developers.cloudflare.com/ssl/reference/browser-compatibility/#non-sni-support> (visited on 05/08/2025).
- [11] Cloudflare. *Cloudflare for SaaS*. Sept. 2024. URL: <https://developers.cloudflare.com/cloudflare-for-platforms/cloudflare-for-saas/> (visited on 05/08/2025).
- [12] Cloudflare. *Enable mTLS*. Aug. 2024. URL: <https://developers.cloudflare.com/ssl/client-certificates/enable-mtls/> (visited on 05/08/2025).
- [13] Cloudflare. *Mutual TLS (mTLS) · Cloudflare API Shield docs*. Aug. 2024. URL: <https://developers.cloudflare.com/api-shield/security/mtls/> (visited on 01/20/2025).
- [14] Antoine Delignat-Lavaud and Karthikeyan Bhargavan. “Network-based Origin Confusion Attacks against HTTPS Virtual Hosting”. In: *Proceedings of the 24th International Conference on World Wide Web*. WWW ’15. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, May 2015, pp. 227–237. ISBN: 978-1-4503-3469-3. DOI: 10.1145/2736277.2741089.
- [15] Zakir Durumeric and David Adrian. *zgrab2*. URL: <https://github.com/zmap/zgrab2>.
- [16] Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. “ZMap: Fast Internet-wide Scanning and Its Security Applications”. In: *22nd USENIX Security Symposium (USENIX Security 13)*. 2013, pp. 605–620. ISBN: 978-1-931971-03-4. URL: <https://www.usenix.org/conference/usenixsecurity13/technical-sessions/paper/durumeric>.
- [17] D. Eastlake. *Transport Layer Security (TLS) Extensions: Extension Definitions*. Jan. 2011. DOI: 10.17487/rfc6066.
- [18] Pasi Eronen, Hannes Tschofenig, Hao Zhou, and Joseph A. Salowey. *Transport Layer Security (TLS) Session Resumption without Server-Side State*. Request for Comments. Jan. 2008. DOI: 10.17487/RFC5077.
- [19] Fastly. *Setting up Mutual TLS authentication*. URL: <https://docs.fastly.com/en/guides/setting-up-mutual-tls-authentication> (visited on 05/08/2025).
- [20] Fastly. *TLS service options – Dedicated IP addresses*. URL: <https://docs.fastly.com/products/tls-service-options#dedicated-ip-addresses> (visited on 05/08/2025).
- [21] Filippo Valsorda. *Ticketbleed (CVE-2016-9244)*. URL: <https://filippo.io/Ticketbleed/> (visited on 08/05/2024).
- [22] Lucas Foppe, Jeremy Martin, Travis Mayberry, Erik C. Rye, and Lamont Brown. “Exploiting TLS Client Authentication for Widespread User Tracking”. In: *Proceedings on Privacy Enhancing Technologies 2018.4* (Oct. 2018), pp. 51–63. ISSN: 2299-0984. DOI: 10.1515/popets-2018-0031.
- [23] Paul Ford-Hutchinson. *Securing FTP with TLS*. Request for Comments. Oct. 2005. DOI: 10.17487/RFC4217.
- [24] Sean Gallagher. *Cloudflare gives Internet a present: free, no-hassle “Universal” SSL*. Sept. 2014. URL: <https://arstechnica.com/information-technology/2014/09/cloudflare-gives-internet-a-present-free-no-hassle-universal-ssl/> (visited on 05/19/2025).
- [25] Milad Ghaznavi, Elaheh Jalalpour, Mohammad A. Salahuddin, Raouf Boutaba, Daniel Migault, and Stere Preda. “Content Delivery Network Security: A Survey”. In: *IEEE Communications Surveys & Tutorials 23.4* (2021). Conference Name: IEEE Communications Surveys & Tutorials, pp. 2166–2190. ISSN: 1553-877X. DOI: 10.1109/COMST.2021.3093492.
- [26] Google Cloud. *Mutual TLS overview | Load Balancing*. URL: <https://cloud.google.com/load-balancing/docs/mtls> (visited on 05/08/2025).
- [27] *Heartbleed Bug*. URL: <https://heartbleed.com/> (visited on 08/28/2024).
- [28] Sven Hebrok, Simon Nachtigall, Marcel Maehren, Nurullah Erinola, Robert Merget, Juraj Somorovsky, and Jörg Schwenk. “We Really Need to Talk About Session Tickets: A Large-Scale Analysis of Cryptographic Dangers with TLS Session Tickets”. In: *32nd USENIX Security Symposium (USENIX Security 23)*. 2023, pp. 4877–4894. ISBN: 978-1-939133-37-3. URL: <https://www.usenix.org/conference/usenixsecurity23/presentation/hebrok>.
- [29] Paul E. Hoffman. *SMTP Service Extension for Secure SMTP over Transport Layer Security*. Request for Comments. Feb. 2002. DOI: 10.17487/RFC3207.
- [30] Zi Hu, Liang Zhu, John Heidemann, Allison Mankin, Duane Wessels, and Paul E. Hoffman. *Specification for DNS over Transport Layer Security (TLS)*. Request for Comments. May 2016. DOI: 10.17487/RFC7858.
- [31] Liz Izhikevich, Gautam Akiwate, Briana Berger, Spencer Drakonitaidis, Anna Ascherman, Paul Pearce, David Adrian, and Zakir Durumeric. “ZDNS: a fast DNS toolkit for internet measurement”. In: *Proceedings of the 22nd ACM Internet Measurement Conference*. IMC ’22. New York, NY, USA: Association for Computing Machinery, Oct. 2022, pp. 33–43. ISBN: 978-1-4503-9259-4. DOI: 10.1145/3517745.3561434.
- [32] James Kettle. *Practical Web Cache Poisoning*. Aug. 2018. URL: <https://portswigger.net/research/practical-web-cache-poisoning> (visited on 01/13/2025).
- [33] Katharina Krombholz, Wilfried Mayer, Martin Schmiedecker, and Edgar Weippl. ““I have no idea what i’m doing” - on the usability of deploying HTTPS”. In: *26th USENIX security symposium (USENIX security 17)*. Vancouver, BC: USENIX Association, Aug. 2017, pp. 1339–1356. ISBN: 978-1-931971-40-9. URL: <https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/krombholz>.

- [34] Victor Le Pochat, Tom van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczynski, and Wouter Joosen. “Tranco: A research-oriented top sites ranking hardened against manipulation”. In: *26th annual network and distributed system security symposium san diego, california, USA, february 24-27, 2019*. NDSS 2019. San Diego, CA, USA: The Internet Society, 2019. URL: <https://www.ndss-symposium.org/ndss-paper/tranco-a-research-oriented-top-sites-ranking-hardened-against-manipulation/>.
- [35] Jinjin Liang, Jian Jiang, Haixin Duan, Kang Li, Tao Wan, and Jianping Wu. “When HTTPS Meets CDN: A Case of Authentication in Delegated Service”. In: *2014 IEEE Symposium on Security and Privacy*. ISSN: 2375-1207. May 2014, pp. 67–82. DOI: 10.1109/SP.2014.12.
- [36] Microsoft. *Tutorial: Configure an Application Gateway with TLS termination using the Azure portal*. Apr. 2023. URL: <https://learn.microsoft.com/en-us/azure/application-gateway/create-ssl-portal> (visited on 05/08/2025).
- [37] Seyed Ali Mirheidari, Sajjad Arshad, Kaan Onarlioglu, Bruno Crispo, Engin Kirda, and William Robertson. “Cached and Confused: Web Cache Deception in the Wild”. In: *29th USENIX Security Symposium (USENIX Security 20)*. 2020, pp. 665–682. ISBN: 978-1-939133-17-5. URL: <https://www.usenix.org/conference/usenixsecurity20/presentation/mirheidari>.
- [38] Seyed Ali Mirheidari, Matteo Golinelli, Kaan Onarlioglu, Engin Kirda, and Bruno Crispo. “Web Cache Deception Escalates!” In: 2022, pp. 179–196. ISBN: 978-1-939133-31-1. URL: <https://www.usenix.org/conference/usenixsecurity22/presentation/mirheidari>.
- [39] Mozilla. *ssl_handshake_version | GLAM*. 2025. URL: https://glam.telemetry.mozilla.org/firefox/probe/ssl_handshake_version/explore (visited on 01/20/2025).
- [40] Henrik Nielsen, Jeffrey Mogul, Larry M. Masinter, Roy T. Fielding, Jim Gettys, Paul J. Leach, and Tim Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.1*. Request for Comments. June 1999. DOI: 10.17487/RFC2616.
- [41] OpenSSL. *OpenSSL Library*. URL: <https://openssl-library.org/> (visited on 05/08/2025).
- [42] OpenSSL. *SSL_CTX_set_client_hello_cb - OpenSSL Documentation*. URL: https://docs.openssl.org/3.5/man3/SSL_CTX_set_client_hello_cb/ (visited on 05/08/2025).
- [43] OpenSSL. *SSL_CTX_set_session_id_context - OpenSSL Documentation*. URL: https://docs.openssl.org/3.5/man3/SSL_CTX_set_session_id_context/ (visited on 05/08/2025).
- [44] OpenSSL. *SSL_CTX_set_tlsext_servername_callback - OpenSSL Documentation*. URL: https://docs.openssl.org/3.5/man3/SSL_CTX_set_tlsext_servername_callback/ (visited on 05/08/2025).
- [45] OpenSSL. *SSL_CTX_set_verify - OpenSSL Documentation*. URL: https://docs.openssl.org/3.5/man3/SSL_CTX_set_verify/ (visited on 05/08/2025).
- [46] Arnis Parsovs. “Practical Issues with TLS Client Certificate Authentication”. In: *Proceedings 2014 Network and Distributed System Security Symposium (2014)*. Conference Name: Network and Distributed System Security Symposium ISBN: 9781891562358 Place: San Diego, CA Publisher: Internet Society. DOI: 10.14722/ndss.2014.23036.
- [47] Maximilian Radoy, Sven Hebrok, and Juraj Somorovsky. “In Search of Partitioning Oracle Attacks Against TLS Session Tickets”. In: *29th European Symposium on Research in Computer Security*. To appear. Bydgoszcz, Poland, 2024. ISBN: 978-3-031-70896-1. DOI: 10.1007/978-3-031-70896-1_16.
- [48] Eric Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. Request for Comments. Aug. 2018. DOI: 10.17487/RFC8446.
- [49] Eric Rescorla and Tim Dierks. *The Transport Layer Security (TLS) Protocol Version 1.2*. Request for Comments. Aug. 2008. DOI: 10.17487/RFC5246.
- [50] Joseph A. Salowey, Hao Zhou, Hannes Tschofenig, and Pasi Eronen. *Transport Layer Security (TLS) Session Resumption without Server-Side State*. Request for Comments. May 2006. DOI: 10.17487/RFC4507.
- [51] Caddy Web Server. *Global options (Caddyfile) – strict_sni_host*. URL: <https://caddyserver.com/docs/caddyfile/options#strict-sni-host> (visited on 05/08/2025).
- [52] D. Simon, B. Aboba, and R. Hurst. *The EAP-TLS authentication protocol*. RFC 5216. IETF, Mar. 2008. URL: <http://tools.ietf.org/rfc/rfc5216.txt>.
- [53] Drew Springall, Zakir Durumeric, and J. Alex Halderman. “Measuring the Security Harm of TLS Crypto Shortcuts”. In: *Proceedings of the 2016 Internet Measurement Conference*. IMC ’16. New York, NY, USA: Association for Computing Machinery, Nov. 2016, pp. 33–47. ISBN: 978-1-4503-4526-2. DOI: 10.1145/2987443.2987480.
- [54] Tim Leonhard Storm. “Large Scale Scanning of TLS Session Ticket Confusion”. PhD thesis. 2023. DOI: 10.17619/UNIPB/1-1770.
- [55] Erik Sy, Christian Burkert, Hannes Federrath, and Mathias Fischer. “Tracking Users across the Web via TLS Session Resumption”. In: *Proceedings of the 34th Annual Computer Security Applications Conference*. ACSAC ’18. New York, NY, USA: Association for Computing Machinery, Dec. 2018, pp. 289–299. ISBN: 978-1-4503-6569-7. DOI: 10.1145/3274694.3274708.
- [56] Erik Sy, Moritz Moennich, Tobias Mueller, Hannes Federrath, and Mathias Fischer. “Enhanced performance for the encrypted web through TLS resumption across hostnames”. In: *Proceedings of the 15th International Conference on Availability, Reliability and Security*. ARES ’20. New York, NY, USA: Association for Computing Machinery, Aug. 2020, pp. 1–10. ISBN: 978-1-4503-8833-7. DOI: 10.1145/3407023.3407067.
- [57] Tim Taubert. *Botching Forward Secrecy - The sad state of server-side TLS Session Resumption implementations*. URL: <https://timtaubert.de/blog/2014/11/the-sad-state-of-server-side-tls-session-resumption-implementations/> (visited on 08/05/2024).
- [58] Tim Taubert. *The future of session resumption - Forward secure PSK key agreement in TLS 1.3*. URL: <https://timtaubert.de/blog/2017/02/the-future-of-session-resumption/> (visited on 08/05/2024).
- [59] *Usage Statistics and Market Share of Web Servers, January 2025*. URL: https://w3techs.com/technologies/overview/web_server (visited on 01/06/2025).
- [60] Filippo Valsorda. *We need to talk about Session Tickets*. Sept. 2017. URL: <https://words.filippo.io/we-need-to-talk-about-session-tickets/> (visited on 07/03/2024).
- [61] Matthias Wachs, Quirin Scheitle, and Georg Carle. “Push away your privacy: Precise user tracking based on TLS client certificate authentication”. In: *2017 Network Traffic Measurement and Analysis Conference (TMA)*. June 2017, pp. 1–9. DOI: 10.23919/TMA.2017.8002897.
- [62] Wei Xia, Wei Wang, Xin He, Gang Xiong, Gaopeng Gou, Zhenzhen Li, and Zhen Li. “Old Habits Die Hard: A Sober Look at TLS Client Certificates in the Real World”. In: *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. ISSN: 2324-9013. Oct. 2021, pp. 83–90. DOI: 10.1109/TrustCom53373.2021.00029.

- [63] ZeroSSL. *Caddy - The Ultimate Server with Automatic HTTPS*. URL: <https://caddyserver.com/> (visited on 01/06/2025).
- [64] Mingming Zhang, Xiaofeng Zheng, Kaiwen Shen, Ziqiao Kong, Chaoyi Lu, Yu Wang, Haixin Duan, Shuang Hao, Baojun Liu, and Min Yang. “Talking with Familiar Strangers: An Empirical Study on HTTPS Context Confusion Attacks”. In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’20. New York, NY, USA: Association for Computing Machinery, Nov. 2020, pp. 1939–1952. ISBN: 978-1-4503-7089-9. DOI: [10.1145/3372297.3417252](https://doi.org/10.1145/3372297.3417252).

A Versions

For the experiments in [Section 4](#), we used the publicly available Docker versions with the respective tags, which all were the latest stable images during our experiments:

- httpd:2.4.62
- caddy:2.8.4
- litespeedtech/openlitespeed:1.8.2-lsphp82
- litespeedtech/litespeed:6.3.1-lsphp81
- nginx:1.27.2