A Client-Side Seat to TLS Deployment

Moritz Birghan
Mozilla Corporation
mbirghan@mozilla.com

Thyla van der Merwe
ETH Zurich
tvdmerwe@ethz.ch

***Abstract*—The official release of the latest version of the Transport Layer Security (TLS) protocol, namely TLS 1.3, has been accompanied by rapid adoption across the Web. In 2019, Holz et al. set out to measure this adoption, i.e., deployment and uptake of the protocol (CoRR 2019). Whilst informative and undeniably useful for the TLS community, Holz et al. note that they were unable to measure some of the newer features of TLS 1.3, including zero round-trip time (0-RTT) and post-handshake authentication (PHA). The altered structure of TLS 1.3, with more encryption of the handshake, renders measurement of these features impossible via passive monitoring and Internet scanning. Access to client-side TLS telemetry enables our work to address these limitations, and presents a clearer view of the TLS 1.3 adoption landscape. Specifically, our work comments on the true acceptance rate of client-generated early data, and on the odd usage patterns surrounding client authentication that occurs post-handshake. Our work also presents an up-to-date measurement of TLS 1.3 deployment, both confirming and extending the predictions and results presented by Holz et al.**

***Keywords*—TLS, client-side measurement, zero round-trip time, post-handshake authentication**

I. Introduction

The Transport Layer Security (TLS) protocol has recently endured a major technical overhaul, both in terms of security and performance. In August of 2018, after a highly collaborative and iterative design process [1], the Internet Engineering Task Force (IETF) released TLS 1.3, the latest version of the protocol [2]. In order to avoid known protocol weaknesses and improve performance, the TLS Working Group reduced the round-trip time of the TLS handshake, and improved its security offering via the inclusion of state-of-the-art cryptographic mechanisms. Consequently, the latest version of the protocol has new features, including a zero round-trip time (0-RTT) handshake mode, a post-handshake authentication (PHA) feature, and a new, reduced set of cipher suites.

Given the iterative nature of the TLS 1.3 development process, early drafts of the protocol were implemented by some of the larger Internet corporations and organisations, mostly in an experimental fashion. For instance, the CDN provider Cloudflare enabled support for TLS 1.3 in September of 2016 [3], and Mozilla and Google enabled TLS 1.3 in Firefox [4] and Chrome, respectively, in 2017. This pre-emptive testing set the stage for early adoption of the revised protocol, and shortly after its official release in August of 2018, Holz et al. conducted a measurement study of TLS 1.3 usage across the Web [5]. Although perhaps early as an adoption study – the work was conducted in April and May of 2019 – the authors did find strong support for TLS 1.3, which was, however, clearly linked to large Internet companies pushing the protocol on the Web.

The data gathering process of Holz et al. needed to rely on large-scale Internet scans, passive traffic observations, and the Lumen Privacy Monitor [6] dataset (for Android ecosystem data). Although perfectly acceptable, and in fact commonplace in Internet measurement work, the nature of the revised TLS handshake rendered a complete measurement of the newer features of TLS 1.3 impossible. Specifically, encryption of handshake messages from early on in the handshake meant that the success rate of the 0-RTT feature could not be measured by the aforementioned methods, and also that use of the PHA feature was invisible and hence also not measurable. It is this gap that we aim to fill with our work – having a unique client-side perspective of TLS 1.3 connections allows us to step inside the handshake encryption barrier.

A. Contributions

In this work we build on the work presented by Holz et al. and we comment on (i) how the TLS 1.3 adoption landscape has changed since the publication of [5], and (ii) the features that could not be covered by Holz et al., specifically, 0-RTT and PHA. Notably, our work presents a client-side perspective of TLS 1.3 adoption and usage. The scanning techniques in [5] focus on domains and server-side behaviour, and the passive monitoring of TLS traffic via the ICSI SSL Notary [7] together with some supplemental observation in Australia, only gives a partial view of client-side behaviour. Holz et al. note in their work that it is not possible to quickly adapt the ICSI Notary infrastructure to collect data on new features in TLS, and as alluded to previously, only being able to passively monitor connections, without access to encrypted handshake messages, means that usage surrounding new features cannot be fully measured.

With access to Firefox telemetry and the means to measure client-side TLS behaviour from within the client, our work extends the results of Holz et al. Using the Firefox telemetry dataset,[1] we report on the following from the Firefox viewpoint: the TLS 1.3 adoption rate, TLS 1.3 cipher suite usage, 0-RTT usage, and PHA uptake. In order to measure the latter two features, we extend the Firefox telemetry functionality by landing new probes in Firefox. Over a 12-month measurement period extending from November, 2020 to November, 2021, we observe TLS 1.3 as the preferred version of choice with over 50% of Firefox TLS connections making use of TLS 1.3 in November 2021.[2] In the Holz et al. study, the authors found that 4.6% of connections negotiated TLS 1.3 and just under 40% of clients advertised support of TLS 1.3. More than two years on we see confirmation of the expected rapid adoption of TLS 1.3.

[1]In this study we report on data across all Firefox channels, i.e., Nightly, Beta and Release. Of these, Release is by far the largest channel with over 200 million users. For reference, Beta users total in the region of 2 million and Nightly is at 5% of this figure.

[2]Encouragingly, recent telemetry indicates that just over 93% of all Firefox HTTP connections are run over TLS.

Of the three recommended TLS 1.3 cipher suites, we observe the greatest support for the AES variants, with 128-bit AES in GCM dominating in Firefox for desktop (at 82.69% of connections). This is in line with the findings by Holz et al. We note that our dataset does not extend to Firefox on mobile; cipher suite usage may indeed differ on mobile platforms.

In the case of the 0-RTT feature, we observe that in November 2021, 11.39% of Firefox TLS 1.3 connections signal the desire to send early data and that this early data is accepted in 98.20% of these cases. In [5], the authors were able to measure when early data was indicated but were not able to capture when early data was accepted.

When it comes to PHA, over our measurement period, daily PHA occurrences vary but are extremely low when compared to the total number of TLS 1.3 connections observed. Interestingly, however, for the connections observed, all client authentication is performed post-handshake and the feature exhibits "spike" behaviour, which we hypothesise corresponds to organisations and enterprises rolling out services that require client authentication.

The results presented in [5] paint a picture of TLS 1.3 adoption shortly after its official release. Our work updates this view to reflect the adoption status more than two years hence, and also uses a unique client-side position to report on the real-world usage of the newer, distinctive features of TLS, namely 0-RTT and PHA. To our knowledge, our work is the first to examine TLS 1.3 usage purely from a client-side perspective, and can be thought of as a companion piece to [5], both confirming and extending the results and insights put forward in that work. Having a clear and and current view of TLS 1.3 adoption and feature usage is not only of benefit to browser manufacturers and other industry actors but it is also of use to the broader TLS research community.

II. Related Work

As discussed above, Holz et al. [5] capture TLS 1.3 adoption shortly after official release of the standard. Specifically, they deploy Internet scanning techniques to look at TLS 1.3 usage at the domain level, finding that large front-end hosting services such as Cloudflare account for much of the TLS 1.3 adoption observed, and that many connections on mobile platforms can be attributed to organisations such a Google and Facebook deploying TLS 1.3. Through passive monitoring of TLS connections, the authors of [5] attempt to measure the newer features of TLS 1.3, not quite managing to comment on 0-RTT usage and the prevalence of PHA, due to more of the TLS 1.3 handshake being encrypted. Our work fills this gap. At the time of their work in April of 2019, the authors of [5] observed a TLS 1.3 adoption rate of 4.6% across the Web, with just under 40% of clients offering support for TLS 1.3 (note that is this includes earlier variants of TLS 1.3, and not only the finalised protocol).

In their follow up paper [8], Holz et al. extend their measurement period to the end of 2019, and more solemnly discuss the phenomenon of centralization of the Web as a driver for change, again noting that key developments on the Web, such as TLS 1.3 roll-out, are seemingly dictated by a few large and dominant players in the space. At the end of 2019, TLS 1.3 adoption is cited as coming in at just under 20% in [8].

Work by Lee et al. [9], examines TLS 1.3 adoption by observing connections to the top 1M Alexa websites, also incorporating measurement on TLS 1.3 handshake performance and TLS 1.3 implementation effectiveness across clients and servers. Lee et al. confirm the centralized adoption findings of Holz et al., and at the end of their measurement period in December 2020, they observe TLS 1.3 adoption at just over 48%, for their dataset.

To date, works examining the various aspects of TLS 1.3 adoption are few in number. Previous work by Kotzais et al. [10] presents a longitudinal study of TLS deployment over a five-year period. Only a very brief mention of TLS 1.3 is made – at the time of the study, April 2018, only 1% of observed connections were making use of (early variants of) TLS 1.3.

In contrast to the works mentioned above, our unique client-side position allows us to more closely examine client behaviour when it comes to TLS 1.3, and allows us to report on the usage of features that are no longer visible to researchers needing to rely on passive monitoring of TLS connections. A few studies in the area discuss TLS 1.3 adoption in the mobile setting [5], [6], [8]. At this stage, our work does not do this but can be thought of as complementing all of the aforementioned efforts.

III. Background

TLS 1.3 optimises for both security and speed by using state-of-the-art cryptographic primitives in its allowable cipher suites, and by offering a reduced round-trip time in all of its handshake modes. TLS 1.2 and below require two round-trips before the client can start sending application data [11]–[13]. In TLS 1.3, this occurs after one round-trip in an initial handshake, and as part of the client's first message flight in a 0-RTT handshake. Early data can be sent by the client if both the client and the server have agreed on a pre-shared key (PSK), either as part of an initial handshake, or via an out-of-band mechanism. The transmission of early data as part of a 0-RTT handshake does carry certain security risks: the data is not forward secure, i.e., the data is not protected against future compromise of the PSK used to protect it, and replays of the data across connections are possible. In terms of handshake mechanics, when the client sends early data, it sends this encrypted data along with its `ClientHello` message and an `EarlyDataIndication` value in the `ClientHello` extensions field, signalling to the server that early data is present. As the `ClientHello` extensions are not encrypted, this `EarlyDataIndication` value is visible to passive observers, and hence can be captured by infrastructure that passively monitors TLS connections, as was done by Holz et al. in [5]. If a server chooses to accept and process the early data, it will send its own indication value as part of its `EncryptedExtensions` message. As this is not part of the visible `ServerHello` extensions field, this indication of acceptance is not available to passive observers. Additionally, in an initial handshake between the client and the server, if a PSK is established for the purposes of session resumption, which is done via a `NewSessionTicket` message, the server will indicate whether or not this PSK may be used for early data purposes. However, the NewSessionTicket (NST) message is encrypted under the newly established application data keys and is also not visible to passive onlookers.

Besides allowing for client authentication within the handshake, as is the case in TLS 1.2 and below, TLS 1.3 also allows for client authentication post-handshake, i.e., at any time after an initial handshake has successfully completed. A server requests this form of client authentication by sending a CertificateRequest message to the client. To go ahead with the authentication, a client responds with Certificate, CertificateVerify and Finished messages. Willingness of a client to perform post-handshake client authentication can be viewed in the unencrypted ClientHello extensions field but again, as the server request for authentication, as well as the client's response, is encrypted, it is not possible for passive monitoring infrastructure to measure usage of this feature. Cremers et al. [14] point out that the PHA authentication feature fails to provide strong authentication guarantees – as the server is able to accept or reject client authentication done in this fashion "silently", it is possible that the client and the server are not in agreement regarding the client's authentication status. This may have consequences at the application layer, potentially resulting in sensitive information being transmitted by a client assuming authenticated status, and a server handling (possibly storing) this information as if this is not the case.

In order to ensure the use of state-of-the-art data protection mechanisms, the TLS 1.3 specification greatly reduces the number of cipher suites available to the client and the server. All symmetric encryption algorithms are of the authenticated encryption with associated data (AEAD) type, and all key exchange mechanisms are forward secure – static key exchange algorithms have been removed. The TLS 1.3 specification indicates that at a very minimum, TLS_AES_128_GCM_SHA256 be implemented, and TLS_AES_256_GCM_SHA384 and TLS_CHACHA20_POLY1305_SHA256 are highly recommended for implementation. The hash function indicated in each cipher suite is for use in TLS 1.3 key derivation. Cipher suite negotiation is handled via the exchange of ClientHello and ServerHello messages, and is visible to a passive observer.

IV. Data Collection

In this section we describe our data collection process and what differentiates it from the data collection processes of previous works.

A. Mozilla Probe Dictionary

To collect our data, we use the Mozilla probe dictionary [15]. This a collection of all telemetry probes for Firefox. It provides insight into real-world user behaviour and allows us look at the TLS handshake from a client-side perspective. As the client, Firefox has access to the unencrypted TLS 1.3 handshake and the probe dictionary can therefore collect information about it. This is an advantage over the standard data collection method of passive monitoring, which can only monitor the handshake as an outside observer. The probe dictionary makes use of an SQL tool called Redash which can be used to access the collected data at a granular level. Of course, such a dataset only provides detailed information from the client-side viewpoint of the TLS ecosystem. Using our data, we cannot make concrete assumptions about server-side behaviour.

Most of the data we use is accessible to the public. However, our research relationship with Mozilla puts us in a unique position to work with this data. Access to Mozilla's internal tooling allows us to filter the data and to examine it more closely, with more context. Additionally, we also have the luxury of adding new probes to Firefox, so as to measure the new features of TLS 1.3.

We note, however, that our resulting dataset is publicly available – the data can be accessed via the Mozilla probe dictionary by querying the probes that we land in Firefox. Also, we believe that this study is easily reproducible by any researchers wishing to work with Firefox data, given Mozilla's willingness to work with external researchers.

B. New Probes

The probe dictionary collects telemetry on almost every aspect of the TLS handshake, however, some of the newer features of TLS 1.3 were not accounted for at the commencement of our study. We specifically wanted to focus on the 0-RTT and PHA features since previous studies were unable to concretely measure these features. This is due to the early handshake encryption barrier established by TLS 1.3 – many of the feature indicators are protected by either handshake or application data traffic keys and hence are not observable to passive monitors, as is the case in the Holz et al. work [5], [8]. Our unique client-side position, however, allows us to inspect these features. In order to measure 0-RTT and PHA feature usage in the wild, we land two new probes in Firefox. These probes send a ping back to the Mozilla telemetry dataset when events associated with the relevant feature occur within a TLS 1.3 connection. All our probes do is record and increase a count – we are not able to observe the contents of encrypted data, and nor are we capable of discerning browsing destination or identifiable information pertaining to the client.

1) 0-RTT: For the purposes of our 0-RTT probe, we distinguish four different types of 0-RTT status: *not possible*, *possible*, *used*, and *accepted*. Each TLS 1.3 connection observed gets classified into one or more of these states. The first state, *not possible*, is used for TLS 1.3 connections that are not able to make use of the 0-RTT feature. This includes all initial connections that do not involve a PSK handshake. It also includes PSK handshakes for which the PSK has not been signalled for use with early data. In Firefox, it is also possible for users to disable the 0-RTT feature by adjusting their browser preferences. Connections for which this has been done will also be captured in this group. The second group, *possible*, includes all TLS 1.3 connections where it is possible to send early data. This state includes all connections for which the PSK in question has been signalled for use with early data, as indicated in the associated NST message.[3] The third group, *used*, includes connections in which early data is actually sent; just because 0-RTT is possible for a connection does not mean that the feature is actually used. The fourth

[3]This can also be done via a PSK shared out-of-band but in Firefox this functionality does not appear to operational just yet.

group, *accepted*, collects a count of all the TLS 1.3 connections in which the server accepts the early data. This last group is the most interesting as it represents the 0-RTT acceptance rate, something which previous studies have been unable to measure. Note that if a connection is recorded in the *accepted* group, then this means that it is also *used* and was therefore *possible*.

2) PHA: As described in Section III, the PHA feature is an extension of the TLS client authentication mechanism. This feature adds the option for the server to request and subsequently receive client authentication messages at any point after an initial TLS 1.3 handshake has completed. Again, as the PHA messages are encrypted under application data traffic keys, they are not visible to passive onlookers and thus this feature has not been measured by previous works.

Our unique client-side position allows us to measure usage of this feature. In order to set up our probe, we needed to carefully consider where to place the relevant code: Firefox makes use of a third party library, Network Security Services (NSS) [16], to handle many aspects of the TLS handshake. This includes traditional client authentication as well as PHA (in fact, the same code is called to complete both types of authentication). Since NSS is not compiled with Firefox, we were not able to directly land a probe at the applicable invocation point of the PHA feature. However, within Firefox, we are able to observe when (i) client authentication has completed and (ii) when an initial, full handshake has completed. Therefore, to measure occurrences of PHA, for each connection we measure the relative occurrence of these two events. When we notice that client authentication has occurred, we retrospectively examine the connection for completion of an initial handshake; our probe is set up to allow for this.

C. Ethical Considerations

Our probes reveal nothing about the contents of the encrypted data that we monitor – they merely serve to increase a count value regarding when certain events are triggered on a TLS 1.3 connection. Our data can in no way be used to discern individual user browsing behaviour or fingerprint individual clients. We say more about our ethical data collection practices in Appendix A.

D. A Different Data Slice

In comparison to previous works, our data concerns TLS 1.3 usage from the perspective of a single browser, Firefox. As noted above, being able to monitor handshake events from within the client allows for us to gain further insights into the prevalence and usage of the newer features of TLS 1.3 on the Web but it does not explicitly allow us to comment on server-side behaviour. We can possibly infer certain ecosystem behaviours and trends but we cannot do so from the perspective of differing clients connecting to many servers, as is the case in previous works [5], [8], [9]. However, our perspective is not limited to the narrow population segment that is covered by the SSL Notary, which is predominantly institutions of higher learning. Our data covers a much larger slice of the population, capturing a wider class of use cases and user behaviours. We believe that our work complements these prior works and their associated methodologies – in combination with these efforts, our work helps to broaden the understanding of TLS 1.3 deployment on the Web.

V. Results

In this section we outline our findings. We start by looking at the adoption data collected for TLS 1.3, and then go into detail regarding the 0-RTT and PHA features. For this analysis we cover only Firefox on desktop. Our dataset is restricted in this way due to the limited telemetry gathering machinery available for Firefox on mobile platforms.

A. TLS 1.3 Adoption

Due to an iterative development process, spanning four years, many large vendors already had TLS 1.3 implementations before the official release of the protocol in August 2018. Comparing the usage numbers for the differing TLS versions on Firefox, we notice that TLS 1.3 currently displays the highest usage share: just prior to the commencement of our official measurement period, in October of 2020, TLS 1.3 surpassed the 50% mark on Firefox desktop, and over our official 12-month measurement period extending from November 2020 to November 2021, we observe TLS 1.3 as the preferred version of choice with 57.23% of Firefox TLS connections making use of TLS 1.3 in November, 2021. Figure 1 depicts TLS version usage in Firefox.

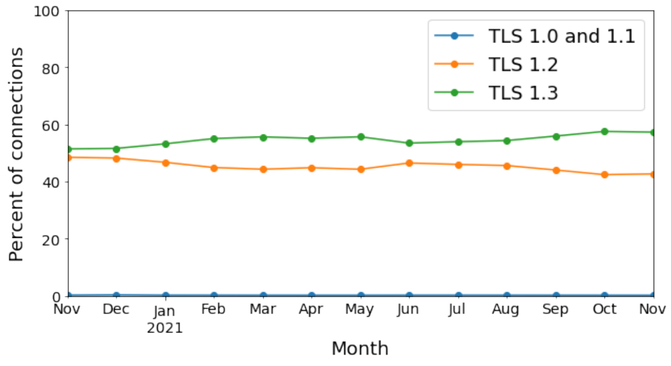

Fig. 1. TLS version usage in Firefox

Also in October of 2020, TLS 1.3 surpassed TLS 1.2 as the most used version of TLS on Firefox. Perhaps unsurprisingly, given the deprecation effort by various browser vendors surrounding TLS 1.0 and 1.1, we see almost no usage at all of these versions, as is visible in Figure 1. The deprecation effort started in mid-2020, with Mozilla initiating deprecation on June 30th, 2020 [17]. All other major browser vendors followed shortly thereafter, with Google doing so for Chrome on July 14th, 2020 [18] and Microsoft for Edge on July 16th, 2020 [19]. This deprecation came after a long period of decline for both versions; at the point of deprecation both versions combined only made up 0.3% of connections in Firefox.

Looking at the usage results, we note that TLS 1.3 has enjoyed a much faster adoption rate than TLS 1.2. This immediate predecessor took six years to reach a 50% usage share on Firefox, which is three times as long as TLS 1.3 took to hit the same target on the browser. This is undoubtedly due to the prolonged development process of TLS 1.3, as also argued in [8] and [9]; most browsers, large websites, and hosting providers already had TLS 1.3 draft implementations at the point of official release. Over our measurement period, as displayed in Figure 1, we notice a gentle increase in TLS 1.3 usage as TLS 1.2 looses usage share. Although still enjoying an evident increase in adoption, we posit that it is possible that TLS 1.3 adoption may stabilise or "level-off" over the coming months. In April 2019, the Holz et al. study reports 4.6% of connections negotiating TLS 1.3 and just under 40% of clients advertising support of TLS 1.3. Their updated work reported a TLS 1.3 usage number of just under 20% in November 2019. A year on from that, in November 2020, Lee et al. [9] report a TLS 1.3 adoption figure of just over 48%. Our figure of 57.23%, yet another year on, confirms the expected early, rapid adoption of TLS 1.3, but in the light of all of the aforementioned data points suggests a slowing down of the TLS 1.3 adoption rate across the Web. It is possible that the larger organisations credited with driving the fast pace of TLS 1.3 adoption have done all they can for TLS 1.3 within the TLS ecosystem, and as with previous versions of the protocol the "tail-end" of the Web will take a while to catch up.

B. TLS Cipher Suites

Cipher suites are essential to the security offering of TLS. With regards to TLS 1.3 cipher suites, which only allow for the use of state-of-the-art cryptographic mechanisms (AEAD algorithms and forward-secure key exchange), Figure 2 captures our observations for the measurement period in question. Of the three recommended TLS 1.3 cipher suites, we see the greatest support for the AES variants, with 82.69% and 17.24% of TLS 1.3 connections making use of `TLS_AES_128_GCM_SHA256` and `TLS_AES_256_GCM_SHA384`, respectively, in November 2021. Use of `TLS_CHACHA20_POLY1305_SHA256` is very low in Firefox at 0.07% of connections. This is perhaps surprising, however, we note that `TLS_CHACHA20_POLY1305_SHA256` is not a mandatory TLS 1.3 cipher suite, and that our measurements only capture Firefox for desktop; usage figures may differ for Firefox on mobile. Also, use of ChaCha20-Poly1305 was, and still is, very low in TLS 1.2 for Firefox desktop (1.13% of TLS 1.2 connections), which may also contribute to its lack of use in TLS 1.3; ecosystem inertia may potentially have a role to play.

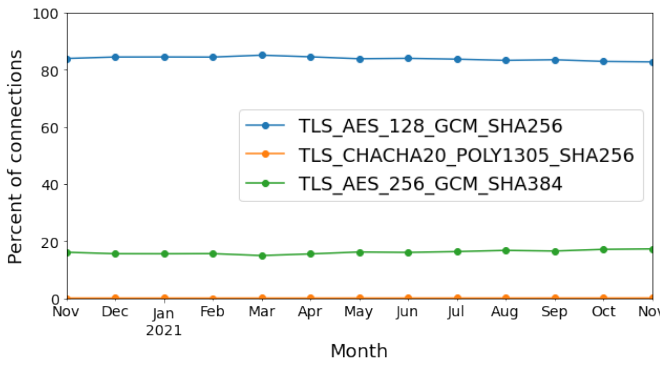

Fig. 2. TLS 1.3 cipher suite usage in Firefox

In terms of TLS 1.3 cipher suites, Holz et al. [5], [8] found that `TLS_AES_128_GCM_SHA256` was used in 79.2% of connections, `TLS_AES_256_GCM_SHA384` in 14.4% of connections, and `TLS_CHACHA20_POLY1305_SHA256` in 6.4% of connections. These measurements are very similar to ours but much higher usage of `TLS_CHACHA20_POLY1305_SHA256` was observed, likely due to the fact that their dataset covered a spread of varying clients, and not only Firefox.

C. 0-RTT

The new 0-RTT feature of TLS 1.3 allows the client to send encrypted application data as part of its first message flight. This is part of the session resumption handshake and uses a cryptographic key, a PSK, from the previous session to encrypt the early data. Previous works were not able to fully observe this feature in the wild since it is hidden behind the handshake encryption barrier of TLS 1.3, and hence is not visible to passive observers. Our unique client-side perspective allow us to observe usage of this feature more readily (and as stated in Section IV, we are not able to, and nor do we wish to, know the contents of the encrypted data).

In order to understand the usage of 0-RTT, we collect different kinds of data, as shown in Figures 3 and 4. We first collect the number of TLS 1.3 connections in which it is possible to make use of the 0-RTT feature ("0-RTT possible" in Figure 3). This means that the server has indicated that the PSK in question can be used for early data purposes (but no guarantee is given that the server will accept this data). This stands at 11.82% in November 2021. We speculate that the dip in 0-RTT usage starting in August 2021 is likely a result of increased TLS 1.3 adoption (from August 2021) *without* the associated increase in 0-RTT support. Perhaps more servers switched to supporting TLS 1.3 but did not enable 0-RTT functionality for a few months; from October 2021 we see an uptick in the usage numbers. However, as we did not complete a dedicated server-side study, this remains speculation on our part. Our second measurement captures the percentage of TLS 1.3 connections where 0-RTT is used, meaning that Firefox sends early data ("0-RTT used"). This occurs in 96.40% of 0-RTT-possible connections, and in 11.39% of all TLS 1.3 connections. Finally, we record the number of connections where 0-RTT is accepted by the server ("0-RTT accepted"). This extends the work done by Holz et al. , which was not able to measure the acceptance rate of 0-RTT data. We find that in 98.20% of connections where early data is sent, it is also accepted. This means that in 11.19% of all TLS 1.3 connections made by Firefox, early data is sent and also accepted.

Over our measurement period, we notice a doubling of the cases in which 0-RTT is possible. This potentially indicates a change in server behaviour when it comes to allowing for early data, however, it may also indicate an increase in the number of connections to servers which already allow for early data. In [5], Holz et al. note a measurement of 6.8% of TLS 1.3 connections in which clients send early data. Two years on, our data for one browser indicates an increase of the number of early data connections on the Web.

D. PHA

The second TLS 1.3 feature that we examine more closely is the PHA feature. This feature allows a client to authenticate

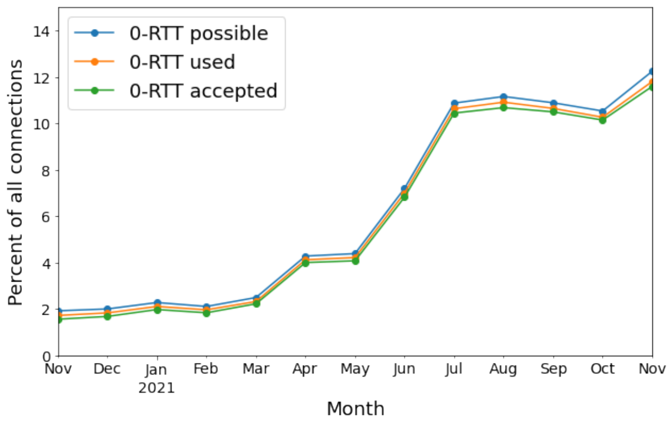

Fig. 3. 0-RTT usage in Firefox

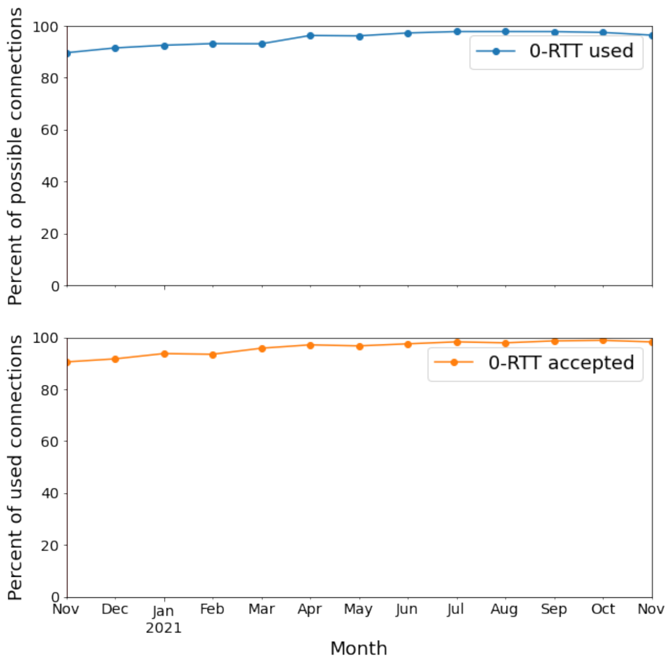

Fig. 4. 0-RTT "used" and "accepted" in Firefox

at any point after an initial TLS 1.3 handshake has occurred. Again, this feature is not observable via passive monitoring as the associated messages are encrypted under the application data traffic keys established during the handshake.

Interestingly, for the Firefox connections observed, almost all client authentication is performed post-handshake (we noted only three connections in which this was not the case). Over our measurement period, daily PHA occurrences vary, never exceeding 6000. Compared to the total number of daily TLS 1.3 connections, which is in the region of 32 - 66 billion, usage of this feature is extremely low. However, the feature does exhibit "spike" behaviour. In October 2020, just prior to the commencement of our official measurement period, we observed the largest of these spikes with 5610 PHA occurrences in one day. We hypothesise that this behaviour coincides with an organisation rolling out PHA to its entire staff and requiring at least one connection to a service requiring client authentication. Figure 5 displays PHA connection numbers starting in September, 2020.

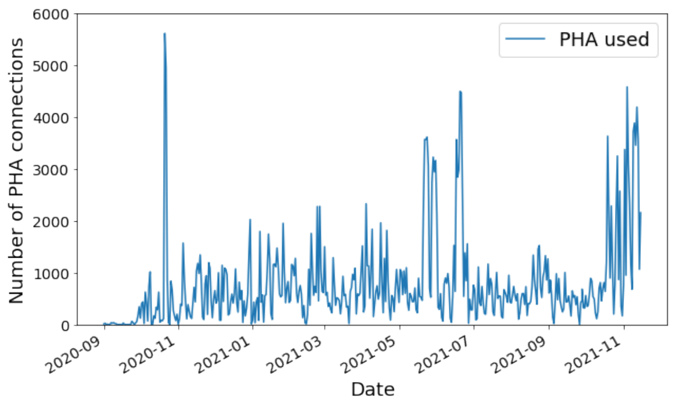

Fig. 5. PHA usage in Firefox

The spike behaviour continues over our entire measurement period, with larger spikes again occurring between May and July of this year. As far as we are aware, our work is the first to measure usage of this feature in the wild. Our low numbers also confirm that server authentication is the dominant authentication use case for TLS on the Web.

VI. Conclusion

Our work extends previous TLS 1.3 measurement studies by commenting on the usage of newer TLS 1.3 features in the wild, something which previous efforts were not able to do because of the altered structure of the TLS 1.3 handshake – more encryption means less visibility to passive observers. Being able to examine usage of these features from within the client, a rather unique vantage point, paints a more colourful picture of TLS 1.3 adoption across the Web. We believe that having more information regarding the prevalence of 0-RTT and PHA in the real-world is important given the security risks associated with these features. Current efforts are under way to eliminate the replay and non-forward secrecy risks inherent to the 0-RTT mechanism [20]–[22], and not only do our results inform these works but they also suggest that whilst slowly climbing, the prevalence of 0-RTT on the Web is still fairly low, which may be for the best given the aforementioned security risks. At this stage, it is also encouraging to see low usage of the PHA feature, specifically given the authentication weakness highlighted by Cremers et al. [14], which may have the potential to contribute to serious attacks against TLS 1.3.

Our results also indicate a potential stabilising of the TLS 1.3 landscape. Over our measurement period of 12 months, our results stayed fairly constant across TLS version, cipher suite, and 0-RTT usage. This may be because the centralisation of the Web by large players, as posited by Holz et al. [5], [8] and confirmed by Lee et al. [9], has done all that can for the adoption of TLS 1.3 on the Web – as with years gone by, less popular sites and servers will lag behind.

To our knowledge, ours is the first work to comment on TLS 1.3 adoption and feature usage from within the client-side encryption barrier. We believe that our work complements previous works and also feel that a coordinated, longitudinal study by additional browser vendors and researchers alike would make for interesting future work.

References

[1] K. G. Paterson and T. van der Merwe, "Reactive and Proactive Standardisation of TLS," in *Security Standardisation Research*, L. Chen, D. McGrew, and C. Mitchell, Eds. Cham: Springer International Publishing, 2016, pp. 160–186.

[2] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, Aug. 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc8446.txt

[3] N. Sulivan, "Introducing TLS 1.3," Cloudflare Blog, 2016, available at https://blog.cloudflare.com/introducing-tls-1-3/. [Online]. Available: https://blog.cloudflare.com/introducing-tls-1-3/

[4] C. Brook, "Mozilla turning tls 1.3 on by default with firefox 52," Threat-Post Blog Post, Oct. 2016, available at: https://threatpost.com/mozilla-turning-tls-1-3-on-by-default-with-firefox-52/121461/.

[5] R. Holz, J. Amann, A. Razaghpanah, and N. Vallina-Rodriguez, "The Era of TLS 1.3: Measuring Deployment and Use with Active and Passive Methods," *CoRR*, vol. abs/1907.12762, 2019.

[6] A. Razaghpanah, A. A. Niaki, N. Vallina-Rodriguez, S. Sundaresan, J. Amann, and P. Gill, "Studying TLS usage in android apps," in *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies, CoNEXT 2017, Incheon, Republic of Korea, December 12 - 15, 2017*. ACM, 2017, pp. 350–362.

[7] J. Amann, M. Vallentin, S. Hall, and R. Sommer, "Extracting Certificates from Live Traffic: A Near Real-Time SSL Notary Service," Technical Report TR-12-014, 2012.

[8] R. Holz, J. Hiller, J. Amann, A. Razaghpanah, T. Jost, N. Vallina-Rodriguez, and O. Hohlfeld, "Tracking the Deployment of TLS 1.3 on the Web: a Story of Experimentation and Centralization," *Comput. Commun. Rev.*, vol. 50, no. 3, pp. 3–15, 2020.

[9] H. Lee, D. Kim, and Y. Kwon, "TLS 1.3 in practice: How TLS 1.3 contributes to the internet," in *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, J. Leskovec, M. Grobelnik, M. Najork, J. Tang, and L. Zia, Eds. ACM / IW3C2, 2021, pp. 70–79.

[10] P. Kotzias, A. Razaghpanah, J. Amann, K. G. Paterson, N. Vallina-Rodriguez, and J. Caballero, "Coming of age: A longitudinal study of TLS deployment," in *Proceedings of the Internet Measurement Conference 2018, IMC 2018, Boston, MA, USA, October 31 - November 02, 2018*. ACM, 2018, pp. 415–428. [Online]. Available: https://dl.acm.org/citation.cfm?id=3278568

[11] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246 (Proposed Standard), Internet Engineering Task Force, Aug. 2008, updated by RFCs 5746, 5878, 6176. [Online]. Available: http://www.ietf.org/rfc/rfc5246.txt

[12] ——, "The Transport Layer Security (TLS) Protocol Version 1.1," RFC 4346, Apr. 2006. [Online]. Available: https://rfc-editor.org/rfc/rfc4346.txt

[13] C. Allen and T. Dierks, "The TLS Protocol Version 1.0," RFC 2246, Jan. 1999. [Online]. Available: https://rfc-editor.org/rfc/rfc2246.txt

[14] C. Cremers, M. Horvat, J. Hoyland, S. Scott, and T. van der Merwe, "A comprehensive symbolic analysis of TLS 1.3," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, Eds. ACM, 2017, pp. 1773–1788. [Online]. Available: https://doi.org/10.1145/3133956.3134063

[15] "Mozilla probe dictionary," Mozilla, available at: https://telemetry.mozilla.org/.

[16] "Network security services," Mozilla, available at: https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS.

[17] T. van der Merwe, "It's the boot for tls 1.0 and tls 1.1," Mozilla Hacks Blog Post, Feb. 2020, available at: https://hacks.mozilla.org/2020/02/its-the-boot-for-tls-1-0-and-tls-1-1/.

[18] C. Thompson, "Chrome ui for deprecating legacy tls versions," Chromium Blog Post, Apr. 2020, available at: https://blog.chromium.org/2019/10/chrome-ui-for-deprecating-legacy-tls.html.

[19] K. Pflug, "Plan for change: Tls 1.0 and tls 1.1 soon to be disabled by default," Microsoft Blog Post, Mar. 2020, available at: https://blogs.windows.com/msedgedev/2020/03/31/tls-1-0-tls-1-1-schedule-update-edge-ie11/.

[20] F. Günther, B. Hale, T. Jager, and S. Lauer, "0-rtt key exchange with full forward secrecy," in *Advances in Cryptology – EUROCRYPT 2017*, J.-S. Coron and J. B. Nielsen, Eds. Cham: Springer International Publishing, 2017, pp. 519–548.

[21] N. Aviram, K. Gellert, and T. Jager, "Session resumption protocols and efficient forward security for TLS 1.3 0-rtt," *J. Cryptol.*, vol. 34, no. 3, p. 20, 2021. [Online]. Available: https://doi.org/10.1007/s00145-021-09385-0

[22] D. Derler, K. Gellert, T. Jager, D. Slamanig, and C. Striecks, "Bloom filter encryption and applications to efficient forward-secret 0-rtt key exchange," *J. Cryptol.*, vol. 34, no. 2, p. 13, 2021. [Online]. Available: https://doi.org/10.1007/s00145-021-09374-3

Appendix A
Ethical Data Collection

Mozilla is a privacy-focused organisation. Therefore, telemetry and data collection are governed by strict guidelines. These guidelines ensure that if users agree to data collection, that no personally identifiable information or other privacy-sensitive data is recorded. Mozilla's internal data collection policy is built upon four major principles: *necessity, privacy, transparency* and *accountability*. *Necessity* implies that only the amount of data that is needed is collected. For every bit of data collected there also has to be a clear business, or in this case research, need. It also means that many telemetry probes get deactivated after a certain time period. In our case, as we performed a long running study, our probes are still active, however, they are due to expire in early 2022. *Privacy* means that users have the ability to choose what data, if any, is collected. All Firefox telemetry probes are "opt-out" enabled, and therefore the user has control over his or her data. A user may opt out of telemetry probes via the browser preference settings. The *transparency* principle forces Mozilla to make all data collection decisions public so as to create an environment where nothing is hidden from the user. This also includes making the telemetry probes accessible to everyone via the telemetry dashboard [15]. Lastly, *accountability* means that for each telemetry probe there is at least one Mozilla employee that is responsible for, and aware of, the data.

To ensure that the above goals are met, Mozilla requires a data review to be performed for all new telemetry probes to be added to Firefox. Internal data stewards then approve, deny or request changes to the planned probes. For the purposes of this work, each of our probes underwent a data review and were approved in August 2020. Our probes can in no way be used to fingerprint clients or discern individual client behaviour – we merely collect counts for the observed 0-RTT and PHA events.