Lucas Foppe, Jeremy Martin*, Travis Mayberry*, Erik C. Rye, and Lamont Brown

Exploiting TLS Client Authentication for Widespread User Tracking

Abstract: TLS, and SSL before it, has long supported the option for clients to authenticate to servers using their own certificates, but this capability has not been widely used. However, with the development of its Push Notification Service, Apple has deployed this technology on millions of devices for the first time. Wachs et al. [42] determined iOS client certificates could be used by passive network adversaries to track individual devices across the internet. Subsequently, Apple has patched their software to fix this vulnerability. We show these countermeasures are not effective by demonstrating three novel active attacks against TLS Client Certificate Authentication that are successful despite the defenses. Additionally, we show these attacks work against all known instances of TLS Client Certificate Authentication, including smart cards like those widely deployed by the Estonian government as part of their Digital ID program. Our attacks include *in-path* man-in-the-middle versions as well as a more powerful *on-path* attack that can be carried out without full network control.

Keywords: TLS, privacy, device tracking, client-certificates, device identifiers, anonymity

DOI 10.1515/popets-2018-0031
Received 2018-02-28; revised 2018-06-15; accepted 2018-06-16.

1 Introduction

Commodity devices like cell phones, tablets, and laptops with *always-on* connectivity increasingly provide consumers a robust, unconstrained mobile user experience. However, with this added convenience comes a range of privacy and tracking concerns due to the fact that those devices are constantly sending and receiving network traffic, often unprovoked by and unknown to the user themselves.

Many people naïvely believe they are anonymous on the Internet, but researchers have repeatedly uncovered scenarios in which mobile devices transmit unencrypted, unique identifiers across a network [11, 26, 42]. When these identifiers are sent over the Internet, or even a local network, it makes tracking of the associated devices and correlation of their traffic possible.

For example, researchers have found the way many devices utilize 802.11 Media Access Control (MAC) addresses and cellular International Mobile Equipment Identities (IMEI) and International Mobile Subscriber Identities (IMSI) can make them particularly vulnerable to tracking [15, 25, 29, 30, 34]. As such, Operating System (OS) vendors and network standards bodies have implemented protocols and policies to mitigate these vulnerabilities. Specifically, Android's Marshmallow OS and iOS 8 introduced MAC address randomization, albeit with mixed results [31, 40].

Exposing a device's unique identifiers to mobile applications has likewise raised significant privacy concerns [19, 21]. In response, Android and Apple have implemented stricter safeguards limiting an application's access to these persistent device identifiers [10]. Similarly, applications may construct their own unique and persistent software identifiers, unwittingly creating tracking identifiers outside the OSes purview and protection [20, 39].

The niche use of TLS Client Certificate Authentication (CCA), in addition to hardware and application identifiers, is another privacy concern. The use of CCA for mutual client-server web authentication while not widespread, has been increasingly utilized for a variety of country-wide digital ID systems [6, 33]. By design, CCA relies on client certificates which contain parameters that are unique to the user and can unintentionally create trackable identifiers. These attributes can be tied to a user's network activity and mobility behavior, which may cause a significant privacy concern. An attacker can further correlate a user's traffic by combining the observed certificate attributes with previously mentioned persistent, globally unique identifiers.

Lucas Foppe: U.S. Naval Academy
***Corresponding Author: Jeremy Martin:** The MITRE Corporation, U.S. Naval Academy, E-mail: jbmartin@mitre.org
***Corresponding Author: Travis Mayberry:** U.S. Naval Academy, E-mail: mayberry@usna.edu
Erik C. Rye: U.S. Naval Academy
Lamont Brown: U.S. Naval Academy

BY-NC-ND

CCA when implemented using the current Transport Layer Security (TLS) 1.2 standard, transmits the unique client certificate to the server prior to the point in the TLS handshake where encryption is established. As such, the certificate and unique client attributes are exposed as plaintext communication [24, 35].

Recent work by Wachs et al. [42] identified a significant privacy risk [2] in the use of CCA in conjunction with Apple's Apple Push Notification service (APNs). Since client certificates are sent unencrypted to the server, exposing unique information within the client certificate, they can allow adversaries to identify users and/or profile their online behavior [42]. This attack was particularly alarming because it applied across the entire Apple ecosystem, and Apple makes up a large share of mobile devices currently in use.

Wachs et al. [42] coordinated with Apple to fix the vulnerability they discovered and protect the privacy of APNs CCA by deferring the transmission of client certificates until after the TLS handshake. Additionally, Parsovs [35] and Johansson [24] have suggested extensions to TLS and modifications to client browsers to likewise protect the confidentiality of client certificates during CCA.

1.1 Contributions

In our paper we show that the existing techniques to protect client privacy during CCA are insufficient. We provide three novel attacks, two *in-path* and one *on-path*, circumventing all known CCA privacy protection techniques. For the distinction between in-path and on-path, see the work of Marczak et al. [28]. First, we employ these attacks to defeat Apple's initial mitigations for CCA privacy vulnerabilities. We worked with the Apple security team to ensure responsible disclosure of these vulnerabilities, and with Apple's release of iOS 11 and macOS High Sierra OSes they were effectively patched. Additionally, our disclosure directly resulted in the APNs CCA mitigation protection for Windows devices running Apple iTunes.

We extend the practicality of these attacks with a variety of TCP reset techniques, further highlighting the real-world impact of the vulnerabilities exposed. TCP reset attacks are not novel on their own, however, when used in tandem with our TLS attacks they provide a robust attack framework which illustrates a practical use case.

Although Apple has patched the vulnerability on their devices, our attack remains relevant due to slow adoption rates of new mobile OS software. We estimate from published data that nearly 50% of iOS devices remain vulnerable to our attacks [8]. Additional real-world tests, depicted in Table 1, were conducted on a large corporate network and are consistent with these reports. As such, our attacks remain effectively employable against a large population of the Apple ecosystem.

Next, we evaluate our attacks against CCA-enabled secure websites. Prior to conducting our active attacks, we inspect the status of CCA privacy related mitigation strategies. Surprisingly, our analysis of Estonia's digital ID infrastructure revealed a large percentage of web servers still failing to protect the exposure of the client certificate from existing passive attacks. Approximately ∼40% of the 78 tested Estonian websites are configured to transmit `CertificateRequest` messages prior to encryption establishment during the TLS handshake, thereby exposing the client certificate. Disturbingly, this figure actually represents an increase from ∼33% reported by a 2014 study [35].

While the mitigation strategies proposed by Parsovs [35] and Johansson [24] exhibit low adoption rates, we evaluate the effectiveness of our attacks against these strategies. Our tests indicate two attacks are viable, including our *in-path* replay attack which is universally effective regardless of TLS server configurations.

Finally, we discuss possible mitigation strategies for our attacks. In the case of Apple, having control of the entire APNs CCA client-server infrastructure affords the luxury of implementing a proprietary, application layer solution. CCA systems supporting traditional web login services, however, are unable to enforce client-side privacy actions, as they cannot compel specific client behaviors in response to our attacks.

As such, we posit that widespread, successful mitigation of our attacks requires full-scale adoption of TLS 1.3. TLS 1.3 introduces a new handshake protocol that mandates encrypted transmission of client certificates. Unfortunately, eliminating the privacy and tracking concerns enumerated in our work entirely necessitates revoking support for TLS 1.2 on both clients and servers in order to prevent downgrade attacks. Since TLS 1.3 is still years from widespread adoption, we echo the recommendation of [35] that browser manufacturers include notifications when TLS client certificates are sent unencrypted, which would indicate either a poorly configured server or a potential Man in The Middle (MiTM) attack. To that end we have disclosed to major browser vendors (Microsoft, Apple, Mozilla, and Google) and are currently coordinating with Mozilla and Apple towards such a solution.

2 Background

TLS is used ubiquitously across the Internet to provide authentication and confidentiality to communications between clients and servers [17, 38]. In the most prevalent use case, the server authenticates to the client using an X.509 certificate containing a valid chain of trust. After the server authenticates itself, a secure channel is established based on a session key derived by the client and server during the handshake. At this point, if the client needs to authenticate to the server (say, to login to their account) then they send a password over this secure channel at the application layer and it is verified by the server.

However, TLS also supports the ability for clients to verify their identity to the server using their own X.509 certificate. This option, called CCA, is initiated by the server, who sends a `CertificateRequest` message along with its `ServerHello` to indicate the client should send a certificate. The client then responds to the server with a `ClientCertificate` message which includes their certificate. Figure 2a illustrates this process.

While it is possible for a client certificate to be manually installed into a web browser, in practice CCA often uses certificates stored on a smart card inserted into a device's smart card reader. When the browser receives a `CertificateRequest` message from the server, it displays a modal dialog box to the user to confirm which certificate the user would like to send for authentication. After the client interacts with the dialog box, the certificate is sent to the server. Figure 1 shows an example of this dialog box in a modern browser.

Once a client certificate is sent across the network, an adversary that observes this traffic can uniquely identify the user that sent it. Oftentimes, this certificate includes the user's full name, email address, and/or a national- or corporate-specific identifier [6, 24, 35]; at the very least, it contains a certificate serial number that can be used to correlate traffic from the same user. An attacker seeking to take advantage of this privacy vulnerability may capture client certificates anywhere along the network path from the client to the server. For example, CCA taking place on an unencrypted wireless access point at a coffee shop exposes users' certificates to adversaries within 802.11 transmission range. Internet Service Providers, from residential to Internet backbone Autonomous Systems, transit the client certificates of their customers; recent work suggests eavesdropping capabilities on less than a dozen networks globally allows the tracking of ∼80% of APNs messages. [42]

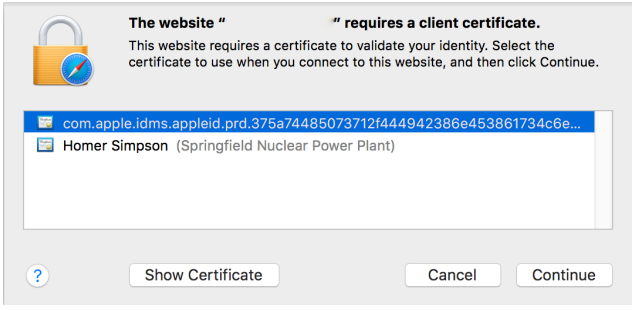


Fig. 1. Example certificate selection dialog on macOS X (Safari)

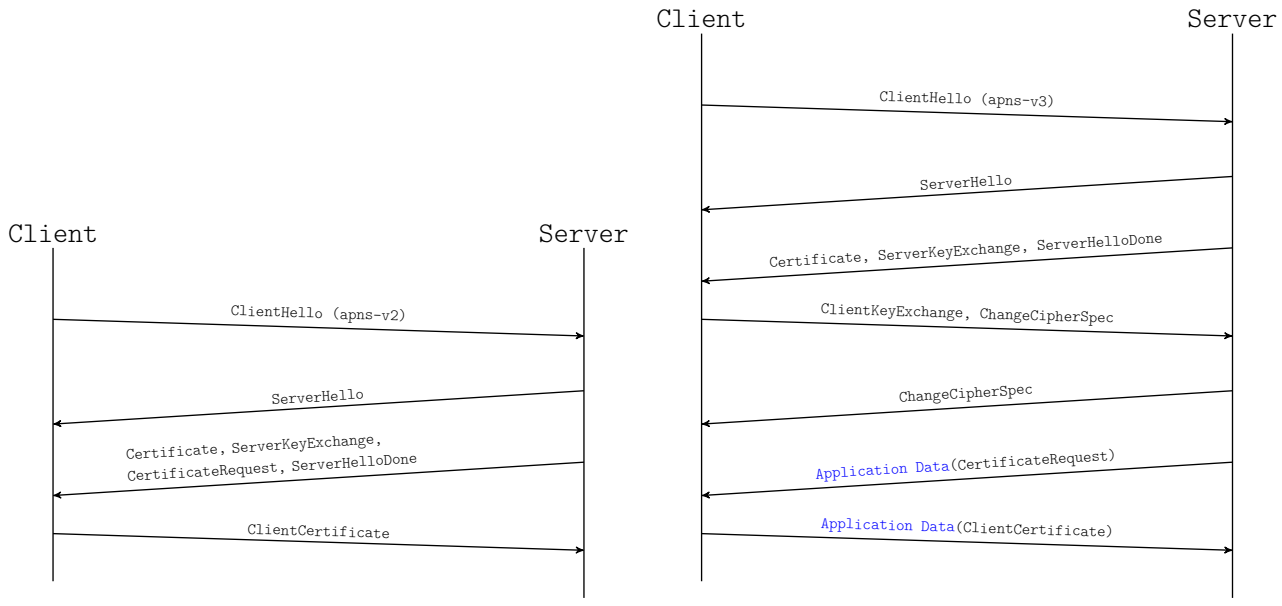
3 Methodology

We evaluate the use of CCA in two ecosystems: first, the Apple APNs infrastructure and second, secure website digital-ID-based authentication. Our observations reveal a remarkable inconsistency in adoption of privacy mitigation strategies. Furthermore, even when effort has been employed to protect the CCA-based exposure of Personally Identifiable Information (PII) our analysis reveals inherent TLS flaws preventing such mitigation attempts.

3.1 Ethical Considerations

In order to evaluate the novel attacks we present in this work, we conduct a variety of experiments on lab devices owned by the authors and the authors' institutions. These devices were allowed to communicate with legitimate network services. Each of the attacks presented here were against our devices, and when successful, resulted in the exposure of our lab related certificates and associated PII. Given the nature of our data experiments, we consulted with our Institutional Review Board (IRB).

The primary concerns of the IRB centered on: i) the information that we collected, and ii) whether the experiment collects data "about whom" or "about what." Because we limit our analysis to TLS packets of our own devices, we do not observe sensitive PII. Our experiment was therefore determined to not be human subject research. Finally, in consideration of beneficence and respect for persons, our work presents no expectation of harm, while the concomitant opportunity for network measurement and security provides a societal benefit.



(a) Pre iOS 10.2, macOS 10.12.2

(b) iOS 10.2.1^{+} and macOS 10.12.3^{+} with APNs v3

Fig. 2. APNs Plaintext Certificate Exposure – Pre/Post Mitigation

3.2 Infrastructure Setup

We construct our research network using personally and institutionally owned equipment in order to observe normal, unaltered TLS communications between our client devices and TLS CCA-enabled services on the Internet. Our design allows us to easily extended our experiments to inspect client and server transmitted packets, modify any observed transmissions, and inject our modified traffic back into our network.

Additionally, we conducted test cases where we assume the role of a weaker active adversary we call an *on-path attacker*. Unlike an *in-path* attacker, an *on-path* attacker can observe client transmitted packets and inject their own packets to the client or server, but is unable to modify any traffic sent by the legitimate client or server. This models a scenario like an open WiFi network, where other clients do not control the network but can spoof messages.

This simple design is implemented using a commodity-grade laptop running Arch Linux 2017 configured with two 802.11 wireless interfaces. The first interface connects to our Internet gateway, the second, using *create_ap*, acts as a bridged interface. Using *Wireshark* we capture all client and server-based communications, as well as any data we inject into our network. Our initial observations evaluate the parameters, attributes, and client-server messages transmitted during CCA. Furthermore, we evaluate and document the observed modifications to the TLS handshake when privacy mitigation strategies have been employed for CCA.

After completing the evaluation of current CCA implementations we proceed to test a variety of *in-path* MiTM attacks using *Scapy* to conduct packet inspection and injection. Lastly, an additional laptop, acting as an *on-path* attacker, injects crafted packets with *Scapy* directly onto the local network of the targeted client device.

4 Analysis

4.1 Apple Push Notification Client Service

Wachs et al. [42] suggest the use of TLS 1.3 as a solution to APNs client certificate privacy leakage by moving the `CertificateRequest` message until after the TLS encryption handshake has been completed, thereby obscuring fields that can be used to create cryptographically unique fingerprints of devices utilizing APNs. As a result, Apple implemented this client certificate obscuration feature of TLS 1.3 in TLS 1.2 using an application layer protocol extension. Figure 2a depicts the behavior in Apple devices prior to iOS 10.2.1 and macOS 10.12.3; Figure 2b shows the client certificate sent to the server as encrypted application data in devices running iOS version 10.2.1 or macOS 10.12.3 and later. The updates were implemented using an upgraded Application Layer

Protocol Negotiation (ALPN), specifically a transition from `apns-security-v2` to `apns-security-v3` within the TLS 1.2 standard.

We explore the effectiveness of this mitigation technique and highlight three novel attacks that circumvent this practice. First, we evaluate the CCA TLS handshake process prior to Apple's mitigation deployment. Next, we examine the CCA behavior using Apple's protected CCA framework derived from [42]. Several key observations were identified revealing significant implementation flaws within the TLS protocol. These flaws, which we illustrate in practice, allow for the systematic retrieval of CCA derived PII.

4.1.1 In-Path ALPN Downgrade Attack

An initial active attack on APNs use of CCA requires an attacker be positioned in the path between the client device and the APNs server the client is attempting to authenticate with. Unlike the previous vulnerability, in which the client PII was trivially exposed, this requires an active attacker who can observe and modify transmitted packets between the server and client.

This attacker, upon intercepting the `ClientHello` message sent by the client device, modifies the ALPN extension attribute from `apns-security-v3` to `apns-security-v2`, thereby indicating to the APNs server that the client does not support the recently improved privacy safeguards. The server responds with the previously vulnerable sequence of messages; namely, the server sends the `CertificateRequest` message prior to the TLS encryption handshake establishment, causing the client to respond with its certificate with cryptographically unique identifiers exposed. Figure 3a demonstrates how an *in-path* attacker might intercept the `ClientHello` message to downgrade the apns-security version used.

4.1.2 In-Path `CertificateRequest` Replay Attack

A second *in-path* attack, depicted in Figure 3b, requires that a previous `CertificateRequest` has been saved by the attacker from a less secure `apns-security-v2` session – any `CertificateRequest` will suffice, as it is client device agnostic. Presumably, attackers in all three threat models described in [42] – local network, regional, or global adversaries – will have access to a myriad of unencrypted `CertificateRequest` messages.

The attack is carried out when a client attempts to establish a connection with an APNs server. The attacker first observes the `ServerHello` and corresponding `ServerCertificate`, `ServerKeyExchange`, and `ServerHelloDone` messages. The attacker simply modifies these messages to include the previously stored `CertificateRequest` and updates relevant TCP and IP header values accordingly. As before, the client responds with the client certificate, exposing its unique identifiers.

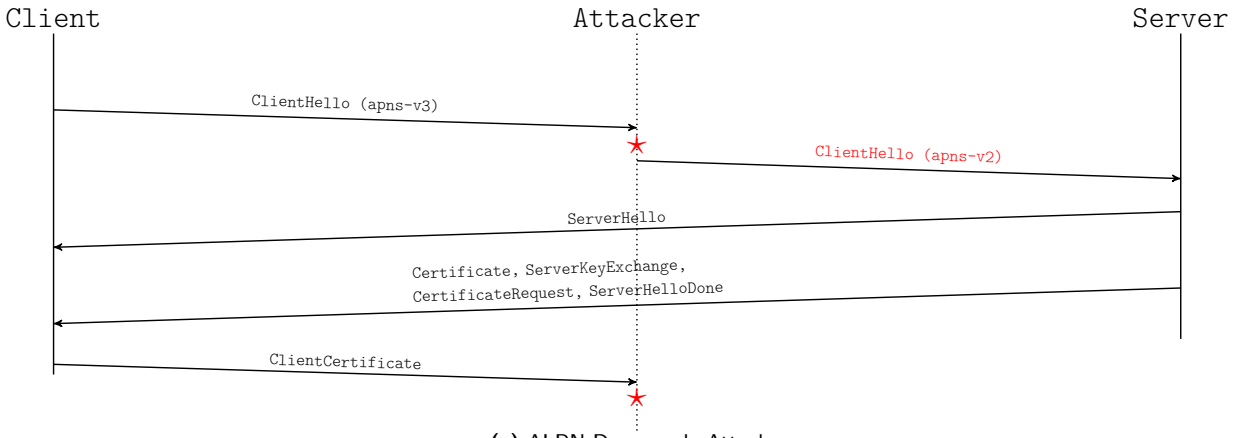
We believe this behavior is most likely due to an oversight when implementing `apns-security-v3`. Ostensibly, if the client is connecting with version 3 of the protocol, it should not respond to `CertificateRequest` messages, but the API in iOS requires that the potential client certificates be specified before the connection is initiated. This means the client does not know whether it will ultimately negotiate to use version 2 or 3 at the point that it has to specify client certificates to the TLS layer. iOS 11 appears to fix this problem by disallowing TLS client certificates entirely, making it impossible to use `apns-security-v2`.

4.1.3 On-Path APNs Server Spoofing

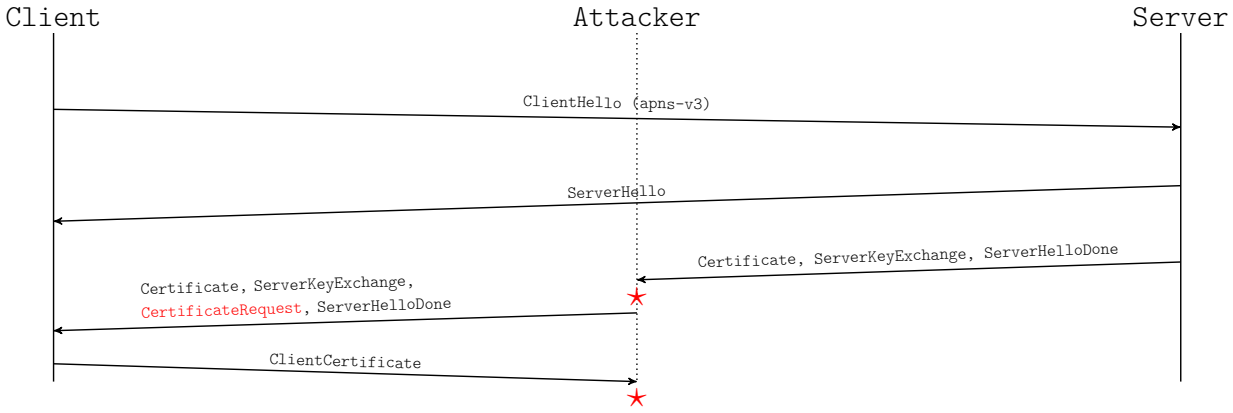
The last attack, an *on-path* attack, does not require the attacker to modify either client or server packets in cases when the attacker is merely an observer on the same transmission medium; an attacker observing a client on an open 802.11 network is a likely and relevant threat. This method requires the attacker to prepare a modified version of the expected server response in advance as described below.

Prior to an attack the adversary must first gather the required server-based messages. The attacker must force an APNs server to respond with a cipher suite that does not include ephemeral Diffie-Hellman key exchange. This is necessary as an ephemeral key exchange includes an additional `ServerKeyExchange` message which contains a signature that protects against replay attacks. The APNs appear to support only elliptic-curve ephemeral Diffie-Hellman (ECDHE) and RSA, so this amounts to removing elliptic-curve cipher suites as an option.

The attacker must prepare a server certificate, as would normally be sent during any TLS connection, but this certificate must include an RSA public key and not an elliptic curve one. This is because when elliptic curve public keys are used for key exchange the TLS protocol also includes an additional signature field that



(a) ALPN Downgrade Attack



(b) CertifcateRequest Replay Attack

Fig. 3. In-Path Attacks

prevents replay attacks. Since we are basically executing a modified replay attack, and we do not actually have the private key, we cannot produce that signature and the attack would fail. Therefore we must have an RSA-based certificate which will not cause the client to expect any signature. The default certificate that APN servers transmit does use an elliptic curve public key, however by sending a carefully crafted `ClientHello` message we can force the APN to send an RSA certificate instead, which can be used to carry out an *on-path* attack. More generally, this means that a similar attack on other servers can only work if those servers have published an RSA certificate. Fortunately RSA certificates are the most common TLS certificates by far [22].

To obtain a non-ephemeral response, we generate modified `ClientHello` messages advertising no support for ECDHE cipher suites or elliptic curve extensions, as well as removing support for most RSA suites, as some RSA suites were identified through trial and error to fail to establish a TLS client response from the server. Specifically, the attacker removes all elliptic curve cipher suites, elliptic curve signature algorithms, ALPN, `status request`, `signed certificate timestamp`, and `extended master secret` fields from its forged `ClientHello` message. The server responds with non-elliptic curve `ServerHello`, `Certificate`, `CertificateRequest`, and `ServerHelloDone` messages.

Additionally, when presented with these cipher suites as options the server responds by choosing TLS 1.0 instead of TLS 1.2. To obtain a proper client response to this message, we further modify the server messages by changing the TLS version from 1.0 back to 1.2, adding the ALPN apns-security parameters into the `SeverHello`, and modify the cipher suite chosen by the server from `TLS_RSA_WITH_AES_128_CBC_SHA` to `TLS_RSA_WITH_AES_128_CBC_SHA256`. The resulting data is saved for later use.

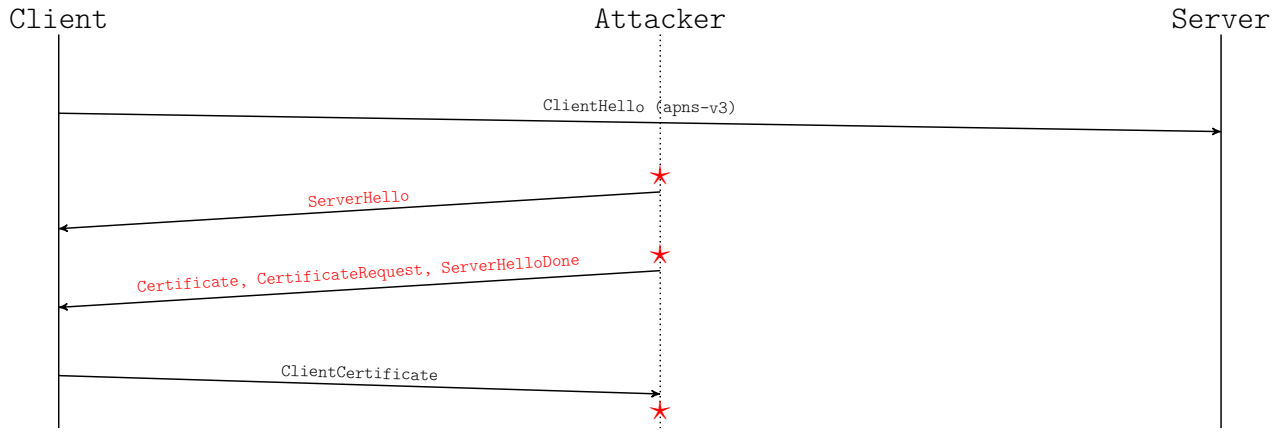


Fig. 4. Eavesdropper Attack (On-Path)

The actual attack, depicted in Figure 4, is carried out when an eavesdropper observes a `ClientHello` message on the network. The attacker spoofs a response from the server using the aforementioned saved server response messages, considering carefully the TCP and IP header field values in its forged server response. The adversary must, for example, be aware of both the client's and server's next expected sequence numbers in order to be in the TCP receive window of client device.

Furthermore, all Apple client devices were observed using the TCP Timestamp Option defined in [23], which requires additional state to be tracked by an attacker in order to successfully impersonate the APNs server. In order for a forged TCP segment to be accepted by the victim device's TCP implementation, the attacker must echo the Timestamp Value (`TSval`) field of the victim device's last segment in its spoofed packet's Timestamp Echo Reply (`TSecr`) field. Although we experimentally determined that the IP Time-To-Live field need not be within any range of the actual APNs server's packets' IP TTL when it reaches the victim, previous work [14] has suggested monitoring received packet TTL values as a defense mechanism against accepting forged data from attackers (though other research shows that legitimate IP TTL values vary wildly in practice, and suggest that IP TTL-based spoofing countermeasures are an unreliable defense at best [44].)

Nevertheless, as a best practice, we suggest an adversary attempting to execute this attack inject server messages with an IP TTL that closely matches that of the legitimate traffic from the APNs server. Upon properly adjusting our spoofed traffic's IP and TCP header values and injected our forged server response, the client responds with the client certificate, completing the attack by exposing its certificate.

4.1.4 Attack Extensions via TCP Reset

An important limitation of these attacks is the adversary must be present and active during the initial TLS handshake in order to execute them. This is somewhat limiting, especially in the Apple scenario, because TLS sessions can be built on top of long-lived TCP connections that last for hours.

In order to exploit these vulnerabilities at will, an attacker also then needs a method to force a new TLS handshake. One way to do this is to force the underlying TCP connection to close. Malicious actors, both *in* and *on the path* between communicating end systems, and those *off-path* or blind to the end systems' communications, have long attempted to use specially crafted TCP messages to terminate established connections [43, 46]. *Off-path* adversaries face the difficulty of guessing the active sequence number space being used by both endpoints of the communication, and much work has been done to quantify the difficulty of *off-path* connection resets, decrease the likelihood of an attacker's success, and survey the state of adoption of mechanisms designed to thwart *off-path* attackers in the wild [27, 36, 37].

By contrast, our threat model focuses on two distinct scenarios: *i.* attackers that are *in the path* between the communicating devices (for *e.g.*, Autonomous System- or nation-state-level adversaries), and *ii.* adversaries who are *on the path* sharing a broadcast communication channel or are within range of the channel in order to passively observe traffic between the victim and the end system with whom it is communicating (for *e.g.*, an 802.11 wireless link.) The most infamous example of nation-state government censorship, the so-called "Great Firewall of China", interferes with established TCP connections by sending a series of forged TCP Re-

Table 1. An overview of the diversity and vulnerabiltiy of Apple operating systems in the network that we monitored [13]

Operating System	Day_0	Day_1	Day_2	Day_3	Day_4	Day_5	Daily Avg (%)	Exposed	In-path	On-path	Reset
OS X 10.13.1	43	62	36	47	63	97	15	✗	✗	✗	✓
OS X 10.13.0	5	4	3	2	4	7					
OS X 10.12.6	224	169	88	140	220	226	46	✗	✓	✓	✓
OS X 10.12.5	21	15	14	9	22	20					
OS X 10.12.4	2	6	0	1	3	0					
OS X 10.12.1	0	0	0	1	0	0	39	✓	n/a	n/a	✓
OS X 10.12.0	4	1	0	3	2	1					
OS X 10.11.6	218	147	72	117	153	204					
OS X 10.10.5	22	17	9	12	18	24					
iOS 11.1.1	0	4	3	8	26	28	78	✗	✗	✗	✓
iOS 11.1	10	13	14	9	11	8					
iOS 11.0.3	44	38	29	21	25	42					
iOS 11.0.2	1	1	0	1	2	3					
iOS 11.0.1	3	0	2	1	1	1					
iOS 11.0	1	1	0	0	0	0					
iOS 10.3.3	13	10	12	9	16	5	20	✗	✓	✓	✓
iOS 10.3.2	7	5	3	1	2	2					
iOS 10.2.1	0	0	0	1	2	0					
iOS 10.2	2	0	0	0	1	2	2	✓	n/a	n/a	✓
iOS 10.1	0	0	0	0	1	0					
iOS 10.0.2	0	1	1	2	0	0					

set (RST) packets to the endpoint within China, causing the TCP connection to abort [14, 44, 45]. Further, Comcast, a large American Internet Service Provider, was reported to use forged TCP RST packets in order to bar the use of the BitTorrent file-sharing system on its network [16, 18]. In our work, we leverage forged TCP RSTs to force the close of a (relatively) long-lived TCP connection to passively observe CCA sent during the initial TLS handshake, or actively modify TLS connection establishment messages in order to force identifiers to be sent prior to message encryption.

In our laboratory environment, we observed the APNs TLS sessions established were long-lived, persistent TCP connections, which limits the scope of our attacks due to the need to modify or spoof TLS handshake messages in order to obtain the client certificate. In order to overcome this limitation, we found simple TCP RST or SYN-based reset attacks forced all iOS devices to renegotiate a new TCP/TLS connection, thereby allowing us to attack any iOS devices on the network. A variety of triggers are plausible to initiate such reset attacks, as the APNs servers are on well-known /8 networks, the APNs TLS ports are well-known, and clients on a LAN will have Apple OUI-based MAC addresses. Further, attackers on the local network of the victim device or in the path between the victim and the APNs server will have access to the TCP and IP header field values necessary to populate the forged TCP messages to close these connections, force a new TLS handshake to occur, and provide an opportunity to mount one of the three attacks.

4.1.5 APNs Vulnerability Scope

All versions of iOS and macOS, as well as Windows devices running iTunes, are, with the exception of iOS 11+ and macOS High Sierra, vulnerable to each of the three attacks [3]. Our responsible disclosure efforts resulted in forthcoming security patches, mitigating attacks against Windows iTunes clients [4].

Apple's published operating system adoption rate statistics, derived from the Apple Store on November 6th 2017 indicate that 48% of iOS devices are vulnerable to passive exposure as well as our novel attacks [8]. Additionally, using the techniques described by Anderson and McGrew [13] we conduct real-world tests on a large enterprise network November 9th-14th. Due to the resulting observed low-adoption rates, these ground-truth results illustrated in Table 1 highlight the significant

number of still vulnerable devices. In particular, laptops running OS X exhibit a significantly low adoption rate. The observed daily average of High Sierra OS X devices was 15%, leaving 46% and 39% of observed devices vulnerable to our active attacks and the previously disclosed passive exposure. We also note these numbers likely show a higher than average adoption rate for iOS 11 due to the fact that the test was conducted on the enterprise network of a technology company, where users are likely to be more security conscious compared to the population at large.

Table 1 derived via collaboration with the authors of Anderson and McGrew [13] is based strictly on the authors ground truth knowledge of the client device's operating system. The table represents the totality of the iOS and macOS users observed within the testing window. At no time were our attacks attempted on this enterprise network. Actual tests of our in-path and on-path attacks were levied across Apple's device and OSs. All tests were conducted solely on our laboratory devices, and as such contain no PII. Apple confirmed our OS version vulnerability conclusions as part of our dialog initiated during responsible disclosure.

4.2 HTTPS CCA via Digital-ID

Gemalto, a global digital security company advertises support to over 40 national eID programs [6]. While the overall use of HTTPS CCA is low representative to more common authentication methods, the increasing implementation of national digital identification systems illustrates a growing privacy concern.

Estonia claiming to have the most highly-developed national ID card system in the world [7], serves as a CCA privacy case study. Over 98% of Estonia's population have been issued a digital government ID allowing for CCA to banking, medical, government, and many other national service websites. A 2014 study by Parsovs [35] evaluated the privacy settings of Estonia's digital-ID CCA web servers. Specifically, the author evaluates the mitigation strategy in which the Apache `mod_ssl` `SSLVerifyClient` parameter is configured to protect the client certificate from trivial adversarial exposure.

We recreate this test, observing surprising results, namely a decreased implementation of this privacy preserving technique. Our results indicate that ∼40% of the 78 tested Estonian websites are currently configured to transmit `CertificateRequest` messages outside a protected channel. Disturbingly, this represents an increase from ∼33% of insufficiently configured web servers. We conclude a representative portion of Estonia's digital-ID CCA infrastructure remains insufficiently protected, exposing the client certificate, allowing for trivial retrieval. Next, we evaluate our active attacks against CCA-enabled websites, including web servers configured to protect trivial exposure of the client certificate.

In Section 4.1 we introduced three attacks, the first an *in-path* attack targeting the ALPN attribute utilized by APNs to distinguish the security version, whereby the server and client negotiate a secure client certificate exchange. As website-based CCA uses no such parameter to protect the exchange of the client certificate this downgrade attack is not applicable against HTTPS CCA schemes.

Implementing our next technique, the `CertificateRequest` *in-path* replay attack, we modify an observed `ServerHello` message, injecting a previously captured `CertificateRequest` into the message sequence. All HTTPS CCA utilizing TLS 1.2 or lower, begins with the RFC defined [38] sequence of messages, whereby we inject our replayed messages as described in Section 4.1.2. The resulting observations of our attacks across the breadth of representative web servers reinforce the fundamental TLS flaw depicted in our APNs attack evaluations, specifically TLS 1.2 offers no warning or protection against attacker modified packets prior to the completion of the TLS handshake. As such we find 100% of HTTPS CCA remains vulnerable to our novel attack.

Lastly, we assess the feasibility of the *on-path* attack where an adversary spoofs the servers initial TLS handshake messages, including the `CertificateRequest` message required to elicit a certificate from the client. This attack relies on a race-condition scenario where an attacker is unable to directly modify a server message but has the ability to inject pre-crafted messages destined for the client network.

This attack works against all known browsers, with the only practical limitation being that a valid certificate must be available for the targeted server that includes an RSA public key. Since the server certificate is validated before the client responds with its own certificate, an attacker must send a valid certificate. Furthermore, since elliptic curve cipher suites require the server to send an additional signature in the ServerHello message that prevents replay attacks, this *on-path* attack can only be successfully executed with RSA-based key exchange, and hence and RSA public key in the certificate. Fortunately, RSA certificates are available for the vast majority of websites [22].

Using Cisco Umbrella's website popularity list [9] we confirm the findings of [22]. We run the cipherscan tool [1] using the top 200 thousand websites identified by Cisco Umbrella, extracting the supported handshake protocols from the resulting TLS certificate transactions. Our results show ∼93% of the resulting 182,656 successful connections support RSA-based key exchanges.

When an RSA certificate is available this attack is significantly more powerful than the generic *in-path* attack. It works even when an *on-path* attacker can only observe client traffic in which they don't have any actual control over the network.

During our analysis of websites implementing CCA, we discovered several that exhibit the curious behavior of establishing a single, long-lived TLS connection upon the initial user logon, but then spawn new TLS connections with CCA when the user interacts with the website after authentication. These "interaction-based" TLS CCA connections persist for a relatively short amount of time, closing after the user-requested object had been transmitted from the server to the client. While resetting the long-lived TLS connection with a TCP RST is a potential avenue for obtaining a client certificate, we observe that in these situations, the configuration of the server causing multiple TLS connections offers numerous opportunities to conduct our *in-path* and *on-path* attacks. Further, resetting the persistent, long-lived TLS connection causes the browser to require user interaction to reselect the desired certificate to authenticate with the server again; whereas waiting for user interaction to spawn a short-lived, non-persistent TCP/TLS connection creates an opportunity for the attacker to attempt an attack without disrupting the user experience. In our initial Apple APNs case study, resetting the long-lived TLS TCP connection is unlikely to ever be noticed by a user, as push notifications are not expected to arrive at a particular time.

5 Mitigating Attacks

Strategies for defeating client certificate surveillance attacks fall into two categories: defense against passive attacks and defense against active attacks.

5.1 Passive Attacks

The most secure defense against these attacks is to move client authentication out of TLS and into the application layer. This allows for the client certificate to be encrypted at all times and successfully prevents passive attacks. However, for many applications using TLS (like HTTPS) it is not possible for the application developer to make changes to the underlying protocol. There are some commercial solutions that perform client authentication using, for example, Java applets [5], which are outside of the normal TLS negotiation, but there are no standards that we are aware of for performing this type of authentication.

The best standards-compliant defense against the passive observation of client certificates is to allow clients to connect to the server without specifying a client certificate, and then perform client authentication during an immediate TLS renegotiation. This has the benefit of sending the client certificate encrypted with the already established session key, obscuring it from eavesdroppers. Such a strategy is possible with Apache web server, for example, by configuring `mod_ssl` with the `SSLVerifyClient` directive in a directory context [35].

5.2 Active Attacks

Unlike the passive version of this attack, our new active attack seems to be very difficult to mitigate. Again, if the client authentication is moved entirely into the application layer then theoretically the active attack would also be prevented. We see this done by Apple in iOS 11 where they incorporate client authentication into their APNS protocol and configure both the client and server not to use TLS client certificates. However, this is only possible for Apple because they have complete control over both the server and client. For most uses of client certificates, the client is a generic unmodified browser and cannot run a custom application layer protocol to perform authentication.

Defenses that work against passive attacks, like Java applets or session renegotiation, unfortunately cannot help against an active attacker because they all come into effect too late in the protocol. The active attacker inserts the `CertificateRequest` into the TLS negotiation at the first server message, before any authentication or integrity information is exchanged. At this point there is no feasible way for the client to know whether it legitimately comes from the server or was

inserted fraudulently. Even cipher suites which provide some extra integrity check during the handshake, like ECDHE, do not attempt to provide integrity for the `CertificateRequest` message and so cannot prevent this attack.

In TLS 1.3, client authentication is moved to a later stage of the protocol, after the session key has been established. However, not even full implementation of TLS 1.3 (which is still some time away) will entirely mitigate an active attack as long as clients still support TLS 1.2 as a fallback. An attacker can simply intercept traffic between the server and client and force them to use TLS 1.2 or even carry out the handshake entirely with the client in an *on-path* attack. Downgrade attacks are an unfortunate consequence of backwards compatibility, and have already been shown to be very dangerous in TLS [12, 32]. We see only two complete defenses against an active attack:

– Browsers can be modified to provide a setting that prevents client certificates from being sent unencrypted during a TLS negotiation. This would require manual opt-in by all potentially effected users, or else risk not allowing connections to non-complying servers.
– Browsers and servers can fully implement TLS 1.3 and refuse to do client authentication with servers that do not support it.

Neither of these are currently feasible given the broad range of devices and users that need to be supported by most web applications.

5.3 TCP Reset

An orthogonal question is whether our TCP reset technique can be prevented. Ostensibly, once a TLS session has been established it should be possible to disallow resets from unencrypted TCP messages, although this is not currently how the protocol is specified. Völker and Schöller [41] have shown that one can repurpose the MD5 signature option within TCP to prevent reset attacks against established TLS connections. However, to our knowledge this remains a proof-of-concept work and is not implemented in any major systems.

5.4 Conclusion

We have demonstrated three new active attacks against Client Certificate Authentication in TLS 1.2 that can allow an attacker to identify and track individual users across the internet. Specifically, we have shown these attacks can be used against Apple's Push Notification Service to track any user with an iOS device prior to iOS 11, a macOS X device prior to High Sierra or a Windows machine with any version of iTunes installed.

Our attacks also apply to all technologies that use TLS client certificates for HTTPS authentication through a web browser, such as smart cards. We present several mitigation strategies for these attacks, but because the core vulnerability we exploit is built into TLS 1.2, full protection from these attacks is unlikely until TLS 1.3 becomes widely deployed.

Acknowledgment

We thank Blake Anderson and David McGrew for providing significant time and resources in assisting us with the APNs vulnerability scope analysis. Additionally, we would like to thank the Apple privacy team who provided prompt feedback and guidance. Views and conclusions are those of the authors and should not be interpreted as representing the official policies or position of the U.S. Government. The author's affiliation with The MITRE Corporation is provided for identification purposes only, and is not intended to convey or imply MITRE's concurrence with, or support for, the positions, opinions or viewpoints expressed by the author.

References

[1] cipherscan. https://github.com/mozilla/cipherscan. Accessed: 2017-12-28.
[2] CVE-2017-2383. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-2383, . Accessed: 2017-10-17.
[3] CVE-2017-13863. https://support.apple.com/en-us/HT208112, . Accessed: 2018-02-24.
[4] CVE-2017-13864. https://nvd.nist.gov/vuln/detail/CVE-2017-13864, . Accessed: 2018-02-24.
[5] URL http://dbsign.com/products/dbsign/uws.
[6] eID card - eID programs. https://www.gemalto.com/govt/identity. Accessed: 2017-11-28.
[7] e-estonia - e-identity. https://e-estonia.com/solutions/e-identity/id-card/. Accessed: 2017-11-28.
[8] App Store - As measured by the App Store on November 6, 2017. https://developer.apple.com/support/app-store/. Accessed: 2017-11-28.
[9] Cisco - Umbrella Popularity List. http://s3-us-west-1.amazonaws.com/umbrella-static/index.html. Accessed: 2017-12-28.

[10] What Is A UDID And Why Is Apple Killing Apps That Track Them? https://www.cultofmac.com/160248/what-the-hell-is-a-udid-and-why-is-apple-worried-about-them-feature/. Accessed: 2017-11-28.

[11] China Deputizes Smart Phones to Spy on Beijing Residents' Real-Time Location. https://www.eff.org/deeplinks/2011/03/china-deputizes-smart-phones-spy-beijing-residents, Oct 2011.

[12] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta, et al. Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 5–17. ACM, 2015.

[13] B. Anderson and D. McGrew. OS Fingerprinting: New Techniques and a Study of Information Gain and Obfuscation. *IEEE Conference on Communications and Network Security*, 2017.

[14] R. Clayton, S. Murdoch, and R. Watson. Ignoring the Great Firewall of China. In *Privacy Enhancing Technologies*, pages 20–35. Springer, 2006.

[15] M. Cunche. I Know Your MAC Address: Targeted Tracking of Individual Using Wi-Fi. *Journal of Computer Virology and Hacking Techniques*, 2014.

[16] M. Dischinger, A. Mislove, A. Haeberlen, and K. P. Gummadi. Detecting Bittorrent Blocking. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pages 3–8. ACM, 2008.

[17] E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3 draft.

[18] P. Eckersley, F. von Lohmann, and S. Schoen. Packet Forgery by ISPs: A Report on the Comcast Affair. *Electronic Frontier Foundation*, 2007.

[19] M. Egele, C. Kruegel, E. Kirda, and G. Vigna. PiOS: Detecting Privacy Leaks in iOS Applications. In *NDSS*, pages 177–183, 2011.

[20] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. TaintDroid: an Information-flow Tracking System for Realtime Privacy Monitoring on Smartphones. *ACM Transactions on Computer Systems (TOCS)*, 2014.

[21] C. Gibler, J. Crussell, J. Erickson, and H. Chen. AndroidLeaks: Automatically Detecting Potential Privacy Leaks in Android Applications on a Large Scale. *Trust*, 12:291–307, 2012.

[22] L.-S. Huang, S. Adhikarla, D. Boneh, and C. Jackson. An Experimental Study of TLS Forward Secrecy Deployments. *IEEE Internet Computing*, 18(6):43–51, 2014.

[23] V. Jacobson, R. Braden, and D. Borman. Tcp extensions for high performance. 1992.

[24] D. Johansson. Privacy Risks with Using Client Certificates for Authentication. http://www.infosecurityeurope.com/__novadocuments/89008?v=635703263638330000. Accessed: 2017-11-28.

[25] D. Kerr. Russian police spy on people's mobile data to catch thieves. https://www.cnet.com/news/russian-police-spy-on-peoples-mobile-data-to-catch-thieves/, Jul 2013.

[26] T. Kohno, A. Broido, and k. c. claffy. Remote Physical Device Fingerprinting. *IEEE Transactions on Dependable and Secure Computing*, 2(2):93–108, 2005.

[27] M. Luckie, R. Beverly, T. Wu, M. Allman, et al. Resilience of Deployed TCP to Blind Attacks. In *Proceedings of the 2015 ACM Conference on Internet Measurement Conference*, pages 13–26. ACM, 2015.

[28] B. Marczak, N. Weaver, J. Dalek, R. Ensafi, D. Fifield, S. McKune, A. Rey, J. Scott-Railton, R. Deibert, and V. Paxson. China's great cannon. *Citizen Lab*, 10, 2015.

[29] J. Martin, D. Rhame, R. Beverly, and J. McEachen. Correlating GSM and 802.11 Hardware Identifiers. In *IEEE Military Communications Conference*, 2013.

[30] J. Martin, E. Rye, and R. Beverly. Decomposition of MAC Address Structure for Granular Device Inference. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 78–88. ACM, 2016.

[31] J. Martin, T. Mayberry, C. Donahue, L. Foppe, L. Brown, C. Riggins, E. C. Rye, and D. Brown. A Study of MAC Address Randomization in Mobile Devices and When it Fails. *Proceedings on Privacy Enhancing Technologies*, pages 365–383, 2017.

[32] B. Möller, T. Duong, and K. Kotowicz. This POODLE Bites: Exploiting the SSL 3.0 Fallback. *PDF online*, pages 1–4, 2014.

[33] E. Network and I. S. Agency. Privacy and Security Risks when Authenticating on the Internet with European eID Cards. https://www.enisa.europa.eu/publications/eid-online-banking/at_download/fullReport. Accessed: 2017-11-28.

[34] B. L. Owsley. Spies in the Skies: Dirtboxes and Airplane Electronic Surveillance. *Mich. L. Rev. First Impressions*, 113: 75–75, 2015.

[35] A. Parsovs. Practical Issues with TLS Client Certificate Authentication. In *NDSS*, volume 14, pages 23–26, 2014.

[36] Z. Qian and Z. M. Mao. Off-path TCP Sequence Number Inference Attack-How Firewall Middleboxes Reduce Security. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 347–361. IEEE, 2012.

[37] A. Ramaiah, R. Stewart, and M. Dalal. Improving TCP's Robustness to Blind In-Window Attacks. Technical report, 2010.

[38] T. Dierks. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, Aug. 2008.

[39] S. Thurm and Y. I. Kane. Your apps are watching you. *The Wall Street Journal*, 17:1, 2010.

[40] M. Vanhoef, C. Matte, M. Cunche, L. S. Cardoso, and F. Piessens. Why MAC Address Randomization is not Enough: An Analysis of Wi-Fi network discovery mechanisms. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pages 413–424. ACM, 2016.

[41] L. Völker and M. Schöller. Secure TLS: Preventing DoS Attacks with Lower Layer Authentication. In *Kommunikation in Verteilten Systemen (KiVS)*, pages 237–248. Springer, 2007.

[42] M. Wachs, Q. Scheitle, and G. Carle. Push Away Your privacy: Precise User Tracking Based on TLS Client Certificate Authentication. In *Network Traffic Measurement and Analysis Conference (TMA), 2017*, pages 1–9. IEEE, 2017.

[43] P. Watson. Slipping in the Window: TCP Reset Attacks. *Presentation at*, 2004.

[44] N. Weaver, R. Sommer, and V. Paxson. Detecting Forged TCP Reset Packets. In *NDSS*, 2009.

[45] X. Xu, Z. M. Mao, and J. A. Halderman. Internet Censorship in China: Where Does the Filtering Occur? In *International Conference on Passive and Active Network Measurement*, pages 133–142. Springer, 2011.

[46] M. Zalewski. Strange Attractors and TCP/IP Sequence Number Analysis, 2001.