A Billion Open Interfaces for Eve and Mallory: MitM, DoS, and Tracking Attacks on iOS and macOS Through Apple Wireless Direct Link

Milan Stute
TU Darmstadt

Sashank Narain
Northeastern University

Alex Mariotto
TU Darmstadt

Alexander Heinrich
TU Darmstadt

David Kreitschmann
TU Darmstadt

Guevara Noubir
Northeastern University

Matthias Hollick
TU Darmstadt

Abstract

Apple Wireless Direct Link (AWDL) is a key protocol in Apple's ecosystem used by over one billion iOS and macOS devices for device-to-device communications. AWDL is a proprietary extension of the IEEE 802.11 (Wi-Fi) standard and integrates with Bluetooth Low Energy (BLE) for providing services such as Apple AirDrop. We conduct the first security and privacy analysis of AWDL and its integration with BLE. We uncover several security and privacy vulnerabilities ranging from design flaws to implementation bugs leading to a man-in-the-middle (MitM) attack enabling stealthy modification of files transmitted via AirDrop, denial-of-service (DoS) attacks preventing communication, privacy leaks that enable user identification and long-term tracking undermining MAC address randomization, and DoS attacks enabling targeted or simultaneous crashing of all neighboring devices. The flaws span across AirDrop's BLE discovery mechanism, AWDL synchronization, UI design, and Wi-Fi driver implementation. Our analysis is based on a combination of reverse engineering of protocols and code supported by analyzing patents. We provide proof-of-concept implementations and demonstrate that the attacks can be mounted using a low-cost ($20) micro:bit device and an off-the-shelf Wi-Fi card. We propose practical and effective countermeasures. While Apple was able to issue a fix for a DoS attack vulnerability after our responsible disclosure, the other security and privacy vulnerabilities require the redesign of some of their services.

1 Introduction

With deployments on over one billion devices, spanning several Apple operating systems (iOS, macOS, tvOS, and watchOS) and an increasing variety of devices (Mac, iPhone, iPad, Apple Watch, Apple TV, and HomePod), Apple Wireless Direct Link (AWDL) is ubiquitous and plays a key role in enabling device-to-device communications in the Apple ecosystem. The AWDL protocol is little understood, partially due to its proprietary nature, especially when it comes to security and privacy. Considering the well-known rocky history of wireless protocols' security, with various flaws being repeatedly discovered in Bluetooth [7], WEP [74], WPA2 [88], GSM [12], UMTS [57], and LTE [51], the lack of information regarding AWDL security is a significant concern given the increasing number of services that rely on it, particularly Apple's AirDrop and AirPlay. It is also noteworthy that the design of AWDL and integration with Bluetooth Low Energy (BLE) are (1) driven by optimizing energy and bandwidth and (2) the devices do not require an existing Wi-Fi access point (AP) with secure connections but are open to communicating with arbitrary devices, thus, potentially exposing various attack vectors.

We conduct the first, to the best of our knowledge, security analysis of AWDL and its integration with BLE, starting with the reverse engineering of protocols and code supported by analyzing patents. Our analysis reveals several security and privacy vulnerabilities ranging from design flaws to implementation bugs enabling different kinds of attacks: we present a man-in-the-middle (MitM) attack enabling stealthy modification of files transmitted via AirDrop, a denial-of-service (DoS) attack preventing communication between devices, privacy leaks allowing user identification and long-term tracking undermining MAC address randomization, and targeted DoS and blackout DoS attacks (i. e., enabling simultaneous crashing of all neighboring devices). The flaws span AirDrop's BLE discovery mechanism, AWDL synchronization, UI design, and Wi-Fi driver implementation. We demonstrate that the attacks can be stealthy, low-cost, and launched by devices not connected to the target Wi-Fi network. We provide proof-of-concept (PoC) implementations and demonstrate that the attacks can be mounted using a low-cost ($20) micro:bit device and an off-the-shelf Wi-Fi card. The impact of these findings goes beyond Apple's ecosystem as the Wi-Fi Alliance adopted AWDL as the basis for Neighbor Awareness Networking (NAN) [19, 94] which, therefore, might be susceptible to similar attacks. Moreover, Google Android provides a NAN API since 2017 pending manufacturer support [38].

Specifically, our contributions are threefold. *First*, we discover security and privacy vulnerabilities in AWDL and AirDrop and present four novel network-based attacks on iOS

and macOS. These attacks are:

(1) A long-term device tracking attack which works *in spite* of MAC randomization, and may reveal personal information such as the name of the device owner (over 75% of experiment cases).
(2) A DoS attack aiming at the election mechanism of AWDL to deliberately desynchronize the targets' channel sequences effectively preventing communication.
(3) A MitM attack which intercepts and modifies files transmitted via AirDrop, effectively allowing for planting malicious files.
(4) Two DoS attack on Apple's AWDL implementations in the Wi-Fi driver. The attacks allow crashing Apple devices in proximity by injecting specially crafted frames. The attacks can be targeted to a single victim or affect all neighboring devices at the same time.

Second, we propose practical mitigations for all four attacks. *Third*, we publish open source implementations of both AWDL and AirDrop as the byproducts of our reverse engineering efforts to stimulate future research in this area.

The rest of this paper is structured as follows. Section 2 provides background information on AWDL. Section 3 shows the results of reverse engineering AirDrop. Section 4 presents an attack to activate AWDL on devices in proximity, while Section 5 shows how we leverage this activation mechanism for user tracking attacks. Sections 6 and 7 feature the desynchronization DoS attack and the MitM attack, respectively. Section 8 reports implementation security vulnerabilities and Section 9 concludes this work. We discuss mitigation techniques and related work in subsections of the respective sections describing the attacks.

2 Background on Apple Wireless Direct Link

AWDL is a proprietary wireless ad hoc protocol based on the IEEE 802.11 standard. In this section, we rely on the reverse engineering efforts of the Open Wireless Link project [81] and summarize the operation of AWDL as presented in [79, 80]. At its core, AWDL uses a channel hopping mechanism to enable "simultaneous" communication with an AP and other AWDL nodes on different channels. This channel hopping is implemented as a sequence of so-called Availability Windows (AWs). For each AW, a node indicates if it is available for direct communication and, if so, on which channel it will be. The channel value can be the channel of its AP, one of the dedicated AWDL social channels (6, 44, or 149), or zero indicating that it will not be listening on any channel. Each node announces its own sequence s consisting of 16 AWs[1] regularly in AWDL-specific IEEE 802.11 Action Frames (AFs). We call the length of such a full 16-AW sequence a *period* τ. Each AW

[1] Actually, [79] differentiates between AWs, Extension Windows (EWs), and Extended Availability Windows (EAWs) where one EAW consists of one AW and three EWs. In this work, we abstract from the smaller entities and simply use the term AW to refer to an EAW.

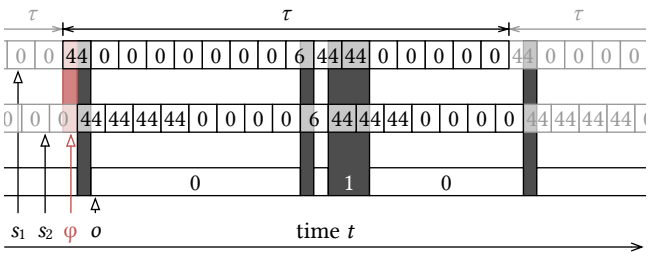


Figure 1: AWDL synchronization depicting period, phase offset, and the overlap function of two channel sequences.

has a length of 64 Time Units (TUs) where 1 TU = 1024 µs, so $\tau \approx 1$ s. Figure 1 depicts these and the following concepts.

To allow nodes to meet and exchange data on the same channel, they need to align their sequences in the time domain. AWDL nodes elect a common master and use its AFs as a time reference to achieve *synchronization*. Each master node transmits *synchronization parameters* which consist of the current AW sequence number i (0 to $2^{16}-1$) and the time until the next AW starts based on its local clock t_{AW}. When receiving an AF from its master at time T_{Rx}, a node approximates T_{AW}, the start of the next AW $i+1$ in local time as:[2]

$$T_{\text{AW}} \approx t_{\text{AW}} + T_{\text{Rx}} \tag{1}$$

and correct its clock accordingly. Eq. (1) does not achieve perfect alignment as it ignores the transmission delay as well as delays in the sender's Tx and receiver's Rx chains. The phase ϕ denotes the effective clock offset between two nodes. Typically, $\phi \leq 3$ ms in practice [79].

A node transmits frames to another AWDL node during AWs where the channels of both nodes are the same. We denote the *overlap* as the relative communication opportunities during one period: an overlap of one means that two nodes are on the same AWDL channel during all 16 AWs, while zero means that they are never on the same channel. Formally, we define the *overlap* O as the integral over the overlap function o of two sequences s_1 and s_2 taking into account the phase where $s(t)$ is the τ-periodic continuation of a sequence, i. e., $s(t+n\tau) = s(t), \forall n \in \mathbb{N}$. Then,

$$o(s_1, s_2, \phi, t) = \begin{cases} 1 & \text{if } s_1(t) = s_2(t-\phi) \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

and

$$O(s_1, s_2, \phi) = \int_{t=0}^{\tau} o(s_1, s_2, \phi, t) \quad . \tag{3}$$

3 Reverse Engineering AirDrop

AirDrop is an application that allows iOS and macOS users to exchange files between devices using AWDL as transport.

[2] Simplification of [79, Eq. (1)].

We reverse-engineered the AirDrop protocol by employing a MitM HTTPS proxy [20] and using a popular disassembler on macOS' `sharingd` daemon and `Sharing` framework where AirDrop is implemented. Based on our findings, we re-implement AirDrop in Python and make it available as open source software [78]. Next, we discuss how different discoverability settings affect the user experience. Then, we describe the technical protocol flow and, finally, explain the difference between authenticated and unauthenticated connections.

3.1 Discoverability User Setting

When opening the sharing pane (see left screenshot in Fig. 2) in AirDrop, nearby devices will appear in the user interface depending on their discoverability setting [2]. In particular, devices can be discovered (1) by *everybody* or (2) by *contacts only*. Alternatively, (3) the *receiving off* setting disables any AirDrop connection requests. AirDrop requires Wi-Fi and Bluetooth to be enabled. By default, Wi-Fi and Bluetooth are enabled, and AirDrop is set to *contacts only*. In addition, we found that devices need to be unlocked to be discovered. Based on a user study that we present in Section 5.2, we found that 80 % of the participants enable AirDrop (59.4 % in *contacts only* and 20.6 % in *everybody* mode) while the other 20 % disabled it. For the rest of the paper, we assume that a target device has AirDrop enabled and is unlocked.

3.2 Protocol and User Interaction

We describe all mechanisms involved in AirDrop including discovery, authentication, and data transfer with a visual aid in Fig. 2. The *sender* initiates the discovery procedure and transfers the data while the *receiver* responds to requests: (1a) The sender emits BLE advertisements including its hashed contact identifiers (see Section 4.1 for details), while the prospective AirDrop receiver regularly scans for BLE advertisements. (1b) The receiver compares the sender's contact hashes with contact identifiers in its own address book if set to *contacts-only* mode. If there is at least one match or if the receiver is in *everyone* mode, the receiver activates its AWDL interface. (1c) Using mDNS/DNS-SD, the sender starts to look for AirDrop service instances via the AWDL interface. (2) For each discovered service, the sender establishes an HTTPS connection with the receiver and performs a full authentication handshake (*Discover*). If authentication is successful, the receiver appears as an icon in the sender's UI. (3) When the user selects a receiver, AirDrop sends a request containing metadata and a thumbnail of the file (*Ask*). The receiver decides whether they want to accept. If so, the sender continues to transfer the actual file (*Upload*).

Next, we discuss the client and server TLS certificates and explain their usage in combination with the sender's and receiver's record data to establish an authenticated connection.

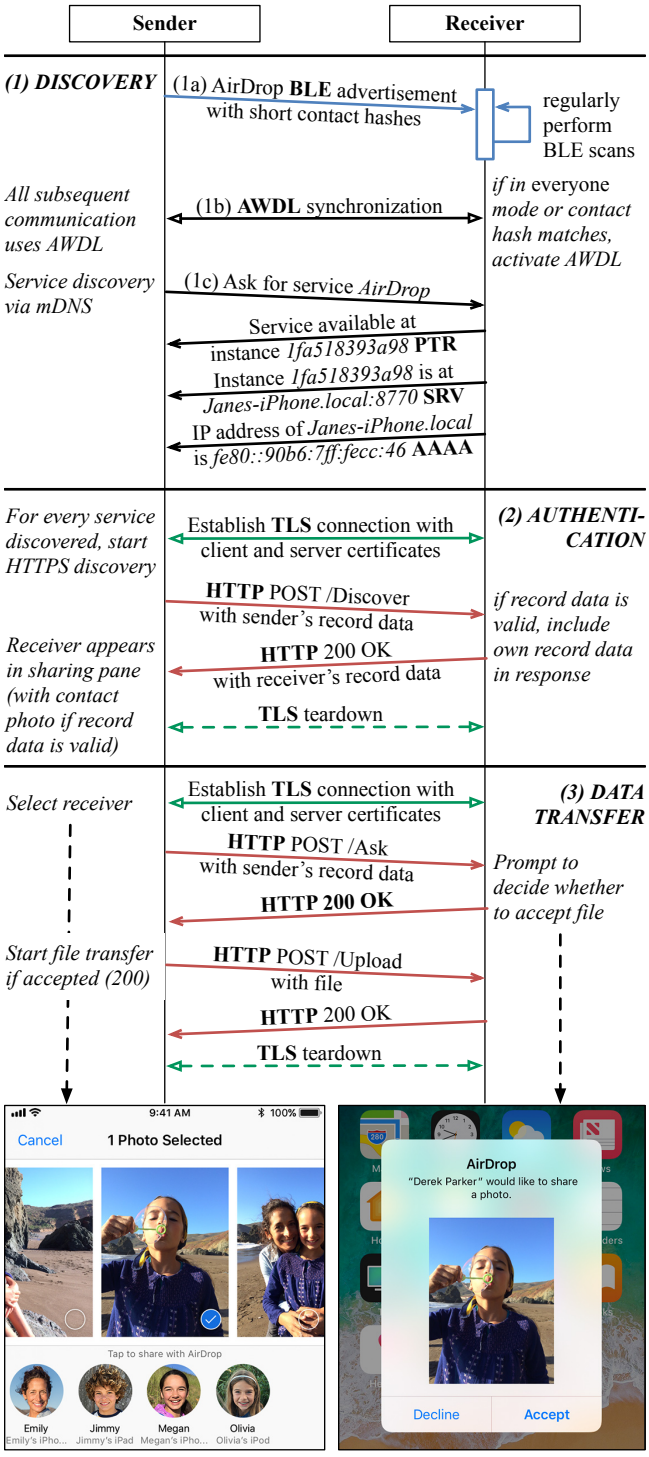


Figure 2: Typical AirDrop protocol workflow including screenshots [2] where user interaction is required.

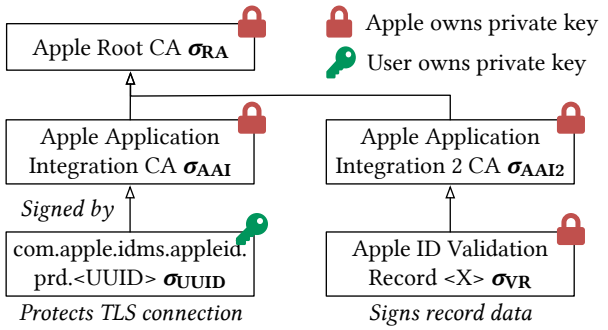


Figure 3: Certificates and CAs involved in AirDrop. Boxes contain the certificates' *common names*.

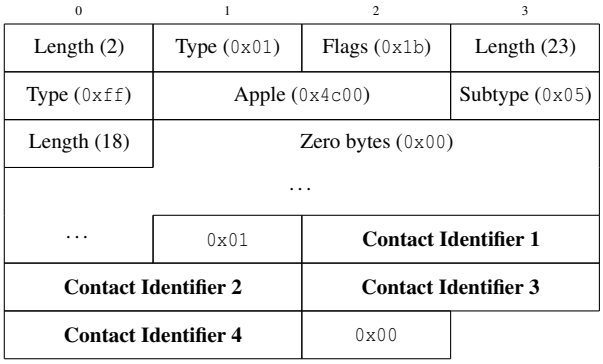


Figure 4: AirDrop BLE advertisement format showing semantics and values of individual bytes.

3.3 (Un)authenticated Connections

AirDrop will always try to set up what we call an *authenticated* connection. Such a connection can only be established between users with an Apple ID and that have each other in their address books. Authentication involves multiple certificates and CAs that we depict in Fig. 3. In order to authenticate, a device needs to prove that it "owns" a certain contact identifier c_i such as email address or phone number associated with its Apple ID, while the verifying device checks whether it has c_i in its address book. When establishing a TLS connection, AirDrop uses a device-specific Apple-signed certificate σ_{UUID} containing a *UUID*. σ_{UUID} is issued when a user logs into the device with its Apple ID. The UUID is not tied to any contact identifiers, so AirDrop uses an Apple-signed *record data* "blob" *RD* containing the UUID and all contact identifiers $c_1, \ldots, c_n$ that are registered with the user's Apple ID in a hashed form. This record data is retrieved once from Apple and then presented for any subsequent AirDrop connection. Formally, *RD* is a tuple:

$$RD = UUID, SHA2(c_1), \ldots, SHA2(c_n) \quad . \tag{4}$$

The signed record data RD_σ additionally includes a signature and a certificate chain (Fig. 3):

$$RD_\sigma = RD, sign(\sigma_{VR}, RD), \sigma_{VR}, \sigma_{AAI2} \quad , \tag{5}$$

where $sign(\sigma, X)$ is the signature of σ over X. When authenticating, a node computes SHA2 over each contact identifier in its address book and compares them with the hashes contained in the presented RD_σ and verifies that the UUID matches the certificate of the current TLS connection. The latter effectively prevents reuse of RD_σ by an attacker using a different TLS certificate.

AirDrop transparently treats a connection as *unauthenticated* if the sender or receiver fails to provide an Apple-signed TLS certificate or valid record data. This means that AirDrop will establish an unauthenticated connection with devices that use a self-signed certificate and provide no record data. While AirDrop's authentication mechanism appears to be cryptographically well-designed, we show in Section 7 how to downgrade an authenticated connection to an unauthenticated one and launch a MitM attack on the data transfer.

4 Activating AWDL on Devices in Proximity

Some of the attacks demonstrated in this work require the targets' AWDL interface to be active, which is typically not the case since an application has to request activation explicitly [79]. We have found that the BLE discovery mechanism integrated with AirDrop (see Section 3) can be exploited to activate all AWDL devices in proximity. Devices in *everyone* mode will enable AWDL immediately after receiving any AirDrop BLE advertisement. We analyze the theoretical performance of brute forcing the truncated contact hash values in AirDrop's BLE advertisements (Fig. 2) to activate the AWDL interfaces of targets in the default *contacts-only* mode. Finally, we build a PoC leveraging a low-cost ($20) BBC micro:bit device and experimentally confirm that the attack is feasible in practice with a target response time of about one second for devices that have 100 contact identifiers in their address book.

4.1 AirDrop BLE Advertisements

We show the actual BLE advertisement frames [17, Vol. 3, Sec. 11] that AirDrop uses including four contact identifier hashes in Fig. 4. They are broadcast as non-connectable undirected advertising (`ADV_NONCONN_IND`). The frames use manufacturer-specific data fields that have fixed values except for the contact hashes. In fact, we found that the contact hashes are the first two bytes of the SHA2 digest of the sender's contact identifiers that are also included in the record data (see Section 3.3). If the sender has less than four identifiers, the remaining contact hash fields are set to zero. Due to the short length, it appears feasible to use brute force to try all possible values.[3]

[3] Note that the sender still has to provide the complete hash during the HTTPS handshake before the receiver accepts the data on an *authenticated* connection.

Table 1: Symbols

SYMBOL	DESCRIPTION
$\mathbb{S}$	Contact hash search space
$\mathbb{C}$	Contacts in the target's address book
w	Target's BLE scan window
i	Target's BLE scan interval
i_{PHY}	Attacker's BLE PHY injection interval
r	Effective contact hash brute force rate
n	Tried hash values per scan window
p (p_j)	Hit probability after one (or j) scans

4.2 Brute Force Analysis

We assume that the attacker does not know the target's contacts and, thus, attempts to enable the target's AWDL interface using brute force. As the target has at least one contact identifier (the address book contains at least the user's own Apple ID), the attacker needs to try $\mathbb{S} = 2^{16} = 65\,536$ hashes in the worst case. Thus, the challenge for the attacker is to quickly send a large number of BLE advertisements *while* the target is conducting a BLE scan. In the following, we analyze *how fast* the attacker can deplete the search space and *how successful* they would be. We start investigating the results for a single BLE scan window and then extend our analysis to multiple scan intervals.

One Scan Window. Let the attacker inject BLE advertisement frames at the physical layer with an interval of i_{PHY}. Further, consider that the attacker has a single radio and that BLE uses three advertisement channels [17]. Also, recall that an AirDrop BLE frame has room for four contact hashes. Then, the attacker's effective brute force rate r can be calculated as:

$$r = \frac{4}{3 \cdot i_{\text{PHY}}} \,. \tag{6}$$

Now, we can compute the number of hash values n that the attacker can inject per scan window w [17] as:

$$n = w \cdot r \,. \tag{7}$$

Let X be a random variable, and $P(X = k)$ denote the probability that the target "sees" k known contact hashes during one scan window. Since the attacker moves through the search space sequentially, we can formulate the problem using the *urn model* in *drawing without replacement* mode which results in a hypergeometric distribution. We get:

$$P(X = k) = \frac{\binom{n}{k}\binom{\mathbb{S}-n}{\mathbb{C}-k}}{\binom{\mathbb{S}}{\mathbb{C}}} \,. \tag{8}$$

In particular, the attacker only requires one hit to activate the target's AWDL interface whose probability we call p:

$$\begin{aligned} p &= P(X \geq 1) = 1 - P(X = 0) \\ &= 1 - \frac{\binom{n}{0}\binom{\mathbb{S}-n}{\mathbb{C}-0}}{\binom{\mathbb{S}}{\mathbb{C}}} = 1 - \frac{\binom{\mathbb{S}-n}{\mathbb{C}}}{\binom{\mathbb{S}}{\mathbb{C}}} \,. \end{aligned} \tag{9}$$

Using the Stirling's approximation $\binom{n}{k} \approx \frac{n^k}{k!}$ for $k << n$, we can simplify Eq. (9) as:

$$\begin{aligned} p &\approx 1 - \frac{\frac{(\mathbb{S}-n)^{\mathbb{C}}}{\mathbb{C}!}}{\frac{\mathbb{S}^{\mathbb{C}}}{\mathbb{C}!}} = 1 - \frac{(\mathbb{S}-n)^{\mathbb{C}}}{\mathbb{S}^{\mathbb{C}}} \\ &= 1 - \left(\frac{\mathbb{S}-n}{\mathbb{S}}\right)^{\mathbb{C}} = 1 - \left(1 - \frac{n}{\mathbb{S}}\right)^{\mathbb{C}} \,. \end{aligned} \tag{10}$$

Multiple Scan Intervals. BLE devices perform scans regularly at a fixed interval i [17]. Let Y be a random variable indicating that the attacker has a hit ($Y = 1$) or not ($Y = 0$) during one scan. We assume that the attacker does not know when the target's scan window starts and, therefore, that Y is i.i.d. between scans.[4] Let j indicate the target's jth scan since the attacker started their brute force attack. Then, the probability that the attacker had k hits after j scans is given by a binomial distribution:

$$P(Y = k) = \binom{j}{k} p^k (1-p)^{j-k} \,. \tag{11}$$

Again, the attacker needs at least one hit whose probability we denote as p_j (note that $p_1 = p$):

$$p_j = P(Y \geq 1) = 1 - P(Y = 0) = 1 - (1-p)^j \,. \tag{12}$$

With Eq. (10), we get:

$$p_j \approx 1 - \left(1 - \frac{n}{\mathbb{S}}\right)^{j\mathbb{C}} \,. \tag{13}$$

We know that j depends on the time since the attack started and the target's BLE scan interval i (the target performs one BLE scan of length w per interval). Let t denote the attack duration, then $j \leq \lfloor t/i \rfloor$. Finally, we denote the success probability at time t as

$$p(t) \approx 1 - \left(1 - \frac{wr}{\mathbb{S}}\right)^{t\mathbb{C}/i} \,. \tag{14}$$

4.3 Jailbreaking BLE Advertisements

The Bluetooth standard imposes a minimum advertisement interval[5] of 100 ms for non-connectable undirected advertising [17, Vol. 6, Sec. 4.4.2.2], which we found is usually

[4] If the attacker knew the start of each scan window, they could follow a better strategy by only sending advertisements while the target is performing a scan. This way, they would deterministically succeed after they had gone through $\mathbb{S}$ once.

[5] The BLE advertisement interval accounts for a frame transmission on each of the three advertisement channels.

```
uint8_t *le_adv = airdrop_init_template()
for (uint16_t h = 0; /* loop */; h += 4) {
  airdrop_set_hashes(le_adv, h, h+1, h+2, h+3);
  for (uint16_t chan = 37; chan < 40; chan++) {
    le_adv_tx(le_adv, chan);
    sleep(0.625 /* in milliseconds */); } }
```

Figure 5: C pseudo code of our BLE brute force attack

enforced in the BLE firmware. By complying to the standard, the attacker would need at least $2^{16} = 27$ minutes to iterate through the entire search space once. If the attacker had access to the BLE physical layer to control and schedule individual transmissions, they could circumvent the standard's restrictions and, thus, iterate through the search space much faster. To this end, we extend an open source BLE firmware [65] for the Nordic nRF51822 [63] chipset to implement our brute force attack. In principle, our attack implementation is very simple and shown in Fig. 5. We use a send interval of $i_{\text{PHY}} = 0.625$ ms resulting in $r = 2133.3\,\text{s}^{-1}$ which allows the attack to iterate through $\mathbb{S}$ in only $2^{16}/f = 30.72$ s. By using three BLE radios (one for each advertisement channel), we could reduce this time to 10.24 s. However, we show that using one radio is sufficient in practice.

4.4 Target Response Times Micro Benchmark

We measure the target response time, i. e., the time it takes for a target to turn on its AWDL interface when being exposed to our attack. In particular, we measure the response time for a *contacts-only* receiver that has 10, 100, and 1000 contact identifiers in their address book. In addition, we include reference measurements for a receiver in *everyone* mode under the same attack.

Setup. For the experiment, we use a Wi-Fi sniffer (Broadcom BCM4360) to receive AWDL AFs and a $20 micro:bit device [58] to inject BLE advertisements. To get the response times, we start a brute force attack and measure the time until we receive the first AF from the target. We then stop the attack and wait until the target stops sending AFs which means that the AWDL interface has turned off. Then, we start over to collect 50 measurements per setting.

Results. We show the results for an iPhone 8 (iOS 12) in Fig. 6. The plot also includes the analytical response time distribution based on Eq. (14), assuming a BLE scan window w and interval i of 30 ms and 300 ms, respectively.[6]. We can make several observations: (1) Our analytical model does not capture our experimental results precisely but approximates them within an order of magnitude which is sufficient for our purposes. (2) The median response time of targets with only 10 contact identifiers in their address book is 10 seconds and

[6] https://lists.apple.com/archives/bluetooth-dev/2014/Sep/msg00001.html

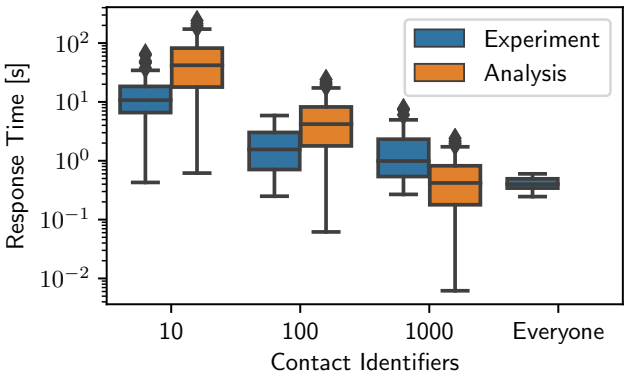


Figure 6: Time it takes until target turns on its AWDL interface after being exposed to our brute force attack.

decreases to about 1 second when more contacts are available. We found that a user has more than 136 contacts on average based on a user study that we describe in Section 5.2. (3) This means that the brute force attack is feasible for scenarios where the target will be in the attacker's communication range for a few seconds.

5 Privacy: Tracking Apple Device Users

In this section, we assess privacy issues in AWDL and find that AWDL devices are easily trackable. First, we discuss protocol fields that enable tracking. Then, we leverage the attack presented in Section 4 to perform an experimental vulnerability assessment at different locations and compare the results with a user study spanning 500 participants. Finally, we discuss possible mitigations.

5.1 Identifying Devices and Users via AWDL Protocol Fields

Even though AWDL implements MAC randomization for the IEEE 802.11 header, AWDL-specific fields contain long-term device identifiers that disclose sensitive information about the user, undermining MAC randomization. In particular, AWDL includes the following sensitive fields in the AFs which devices broadcast *in the clear* multiple times per second when the AWDL interface is active:

- The *hostname* may include parts of the user's *name*, e. g., "Janes-iPhone," which is the default when setting up a new device.
- The *real MAC address* as well as the AP the device is currently connected to.
- The *device class* differentiates between devices running macOS, iOS/watchOS, and tvOS.
- In combination with the *protocol version*, this can be used to infer the OS version, e. g., AWDL v2 is used in macOS 10.12 while AWDL v3 is used in macOS 10.13. The attacker could exploit the OS information during

reconnaissance to mount attacks on vulnerable driver implementations.

Targets need to broadcast AFs to make these vulnerabilities exploitable, which an attacker can practically enforce by mounting the attack presented in Section 4.

5.2 A Survey on the Potential of Apple Device User Tracking

The hostname set by default during Apple iOS and macOS device installation includes the user's name [3]. Due to its frame structure, the AWDL protocol aids an adversary in mapping a hostname with the MAC address of the device. This enables them to track users even if users change this hostname on their device. The combination also enables more sophisticated threats as the person's name can be combined with information from public databases (e. g., US census [86]) to infer their home and work locations, while the MAC address can be used to track them in real-time. To assess what percentage of device hostnames contain parts of the owner's name, we conducted a survey among 500 Apple device users on Amazon Mechanical Turk. This survey contained questions[7] relevant to the attacks demonstrated in this paper, and we report the statistics in the relevant sections. In particular, in the context of tracking, we asked the surveyors if it was easy for other users to find their device because their hostname contained parts of their real name. We report the results of this question along with the results of an experimental evaluation in the next section.

5.3 Experimental Vulnerability Analysis

To demonstrate the feasibility of user tracking using AWDL, we collect the number of discovered devices and check whether that device's hostname includes a person's name in four different locations within the US. We selected the locations to reflect static as well as dynamic environments. In particular, we recorded at a departure gate of an airport, in the reading section of a public library, in a moving metro train, and in the food court of a university.

Determining Whether a Hostname Contains a Person's Name. We use two databases to determine whether a hostname contains a person's name: the 2010 US Census [85] containing 162 253 family names, and the 1918–2017 baby names from the US Social Security Administration [87] containing 96 743 given names. When detecting a new AWDL node, we check string segments separated by hyphens against these two databases.[8] Note that when one segment matches the given name database, it is not matched again as a family name because it is more likely that an Apple device will include a person's given name [3].

[7] The survey questionnaire is available at https://goo.gl/forms/0okC4UphTQBnQ0FB3

[8] If a segment ends with the letter "s," we also check the segment without a trailing "s." In addition, we ignore segments containing common device names such as "iPhone," "Mac," etc. For example, for the hostname "Johns-iPhone," we try to match the strings "Johns" and "John" to our name databases.

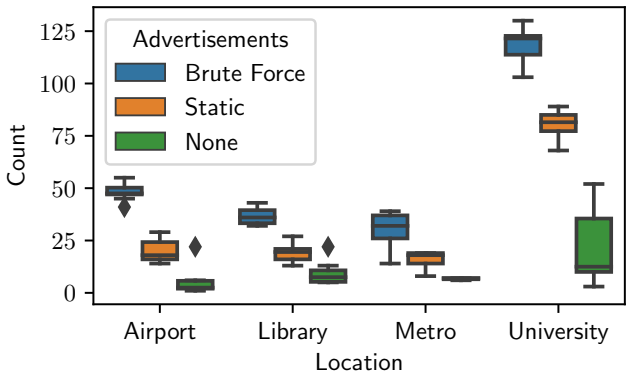


Figure 7: Discovered AWDL devices at one location during one minute.

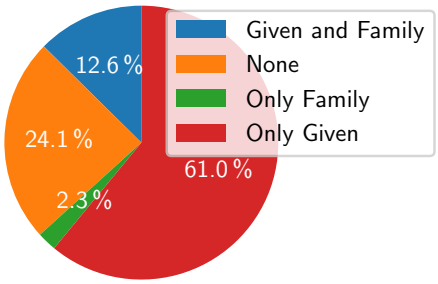


Figure 8: Persons' names distribution in hostnames.

Ethical Statement. To preserve user privacy and not having to store any sensitive user information, we fully automated the name matching procedure. In particular, we only stored salted hashes of the discovered hostnames (to differentiate between devices) together with two bits indicating whether the hostname contained a given or a family name. The salt was generated randomly, kept in memory only, and discarded after the completion of each experiment.

Setup. We do the measurements (a) without an attack (passive), (b) with static BLE advertisements containing only the "zero" contact hash, and (c) with our BLE brute force approach. With (b), only devices in the *everyone* mode should respond, with (c) we also capture those that are in *contacts-only* mode. We run each setting for 60 seconds and repeat it 10 times per location. To avoid statistical bias, we cycle through the (a) to (c) settings back to back in each iteration and use a cooldown time of 40 seconds between them. The cooldown ensures that all devices in proximity have turned off their AWDL interfaces again.

Experimental Results. Fig. 7 shows the number of discovered AWDL devices in the different locations. By using the

brute force approach, we can discover about twice as many devices compared to sending only regular advertisements. This means that in our experiments, approximately 50 % of the Apple devices are in AirDrop's *everyone* mode. Our survey complements our experimental results by indicating that 20 % of Apple device users have AirDrop turned off and, thus, are not trackable via AWDL. It is interesting to note that we are able to pick up AWDL devices even when not sending any advertisements. This can happen if a device (not controlled by us) sends out advertisements itself, for example, when a user opens the AirDrop sharing pane which apparently occurred regularly at the university location. Finally, we found that among all discovered devices, more than 75 % contain a person's name in the hostname. Most devices contain only a given name which is the default for freshly set up Apple devices [3], some contain a combination of a given and family name, and very few contain only a family name. Our survey confirms these results as 68 % answered that it was "easy" or "very easy" for others to recognize their device because it contained their name.

Outlook for Large-Scale Attack. In this analysis, we show what kind of information a motivated attacker would be able to collect. We used a single fixed physical location for each experiment and did not attempt to track any user movement. However, given that we can receive unique identifiers of Apple devices (Wi-Fi MAC address and hostname), mounting a large-scale tracking attack should be trivial for an adversary that can deploy multiple low-cost Wi-Fi and BLE nodes throughout an area.

5.4 Mitigation

We present a short-term solution and then propose two mitigation techniques that remove stable device identifiers to prevent user tracking via AWDL.

Disable AirDrop. Until Apple fixes the problem, the only way to thwart user tracking is to disable AirDrop completely. This presents a countermeasure to our attack presented in Section 4, i. e., the AWDL interface cannot be remotely activated via BLE advertisements.

Hide Real MAC Address When Not Connected to an AP. When a device connects to an AP it uses its real MAC address for communication, in which case AWDL does not disclose new information. However, we have found that the MAC address is occasionally included in AFs even when the device is not connected to an AP. This appears to be unintended behavior and should be fixed via a software update.

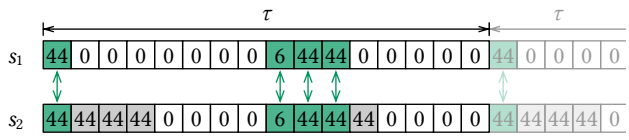
Randomize Hostname for AWDL. Apple devices transmit their hostname in AWDL AFs as well as the mDNS responses during service discovery that are used to find AirDrop instances (see Section 3). As a countermeasure, we propose to use randomized hostname with AWDL similar to MAC address randomization. If an application such as AirDrop needs the real hostname for identification, it should only be transmitted via an encrypted and authenticated channel such as TLS. In fact, AirDrop already transmits the device name in the HTTPS handshake and uses this name in the UI ignoring the hostname from mDNS responses. Therefore, hostname randomization would not require any changes to the AirDrop implementation which would retain backward compatibility.

5.5 Related Work: User Tracking

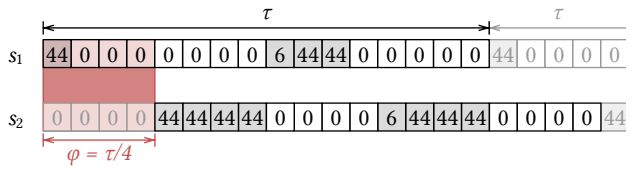
Several related works have studied the topic of user tracking from mobile devices. Some common attack vectors include using the GPS sensor [35, 48, 55, 83, 95], cellular [5, 44, 50, 67], Wi-Fi [15, 40, 69, 96, 97], radio interface fingerprinting [90], and motion sensors [25, 39, 59, 61, 62]. We believe that the above works are orthogonal to our approach, and could be used in conjunction with our approach to improve tracking performance. Many countermeasures have also been proposed to prevent tracking from the above vectors. Some of them include recommending new location frameworks and privacy metrics [9, 28, 29, 30, 49, 64], location obfuscation [1, 6, 18, 72, 92], location cloaking [42, 43], synthesizing locations [16, 46, 53, 82, 98], sensor data obfuscation [22, 23], and permission analysis [32, 45, 68]. Along with resource permissions on mobile devices, these countermeasures limit the practicality of some of the above attacks.

Some device-specific identifiers have also been used for tracking, e. g., IMEI [36, 93], BLE addresses [24, 31, 47], and MAC addresses [21, 33, 54, 60]. While IMEI-based tracking can be easily mitigated by protecting access to this information, BLE is a dominant standard for fitness trackers and smartphone communication and their addresses must be exposed. Tracking using BLE identifiers has been demonstrated to be easy. However, our approach has the added benefit for an attacker that the hostname is exposed. This allows inferring additional user information such as home and work locations, family members, or movement patterns, which are useful for more targeted tracking [34, 84]. Like BLE addresses, MAC addresses are also essential as they form the backbone of layer 2 network communication and must be exposed for networking (e. g., Wi-Fi probe requests).

MAC address randomization has been proposed to prevent device tracking through Wi-Fi probe requests [11, 26]. Today, both Apple and Google implement MAC address randomization in their mobile operating systems. Randomization does improve user privacy; however, some works have demonstrated that devices are still trackable. For example, [89] implemented an algorithm using probe request fingerprinting that has a 50 percent success rate for tracking users for 20 minutes. Another work [56] demonstrated that MAC randomization could be defeated through timing attacks, where a signature based on inter-frame arrival times of probe requests can be used to group frames coming from the same device with distinct MAC addresses. Their framework could group random MAC addresses of the same device up to 75% of cases for about 500 devices. Our work advances the scalabil-



(a) In normal operation, two channel sequences result in non-zero overlap, allowing two nodes to communicate. In this example, they can communicate during four out of 16 EAWs.



(b) A phase shift of a quarter period ($\phi = \tau/4$) results in zero overlap preventing the two nodes from communicating with each other.

Figure 9: Sketch of the desynchronization attack.

ity, tracking time and accuracy of the prior works. We show that, owing to implementation nuances in the AWDL protocol, an adversary can track millions of Apple device owners globally with 100 % accuracy.

6 DoS: Impairing Communication with Desynchronization

AWDL does not employ any security mechanisms. Instead, Apple decided to leave security mechanisms to the upper layers. Thus, while end-to-end confidentiality and integrity can be achieved using a secure transport protocol such as TLS, AWDL frames are vulnerable to forgery which renders any upper layer using AWDL susceptible to attacks on availability. In this section, we present a novel DoS attack that targets AWDL's synchronization mechanism (Section 2) to prevent two nodes from communication with each other. In the following, we describe a novel *desynchronization* attack which aims to minimize the channel sequence overlap of two targets. Next, we evaluate the attack's performance and present an effective mitigation method. Finally, we discuss related work.

6.1 Desynchronizing Two Targets

We exploit AWDL's synchronization mechanism to reduce the channel overlap by inducing an artificial *phase* offset between two targets. In order to succeed, the attacker needs to (1) get recognized as the master by both targets, (2) communicate with each target separately to (3) send different sets of synchronization parameters that result in zero (or minimal) channel overlap. Figure 9 depicts the non-zero overlap in normal operation and the zero overlap as the result of the desynchronization attack. We describe the three steps in the following.

(1) Winning the Master Election. The master election in AWDL is based on a numeric comparison of two values that are transmitted in the *election parameters*. The first value

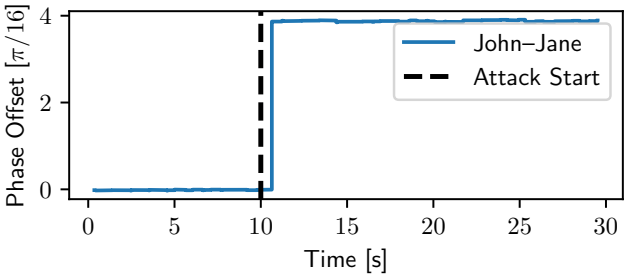


Figure 10: Phase offset between two targets before and after mounting a desynchronization attack which induces a phase shift of $\phi = \tau/4$.

is called *metric*, and each node draws one randomly upon initialization. The numeric range of the *metric* is bounded and depends on the AWDL version that runs on the node [79]. The second value is called *counter* which is initialized to a random value and increases linearly over time while the node is elected as a master. Given the metric and counter values of two nodes A and B as (m_A, c_A) and (m_B, c_B), respectively, then, A wins the master election if

$$c_A > c_B \vee (c_A = c_B \wedge m_A > m_B) \tag{15}$$

and loses otherwise. To consistently win the election, the attacker sets c and m to their maximum values.

(2) Unicasting AFs. The attacker needs to send the synchronization parameters to each target without the other one noticing. We have found that while AFs are typically sent to the broadcast MAC address `ff:ff:ff:ff:ff:ff`, AWDL nodes also accept unicast AFs. Therefore, the attacker can unicast their AFs to make sure that only the intended target receives them.

(3) Phase Shift: Different Synchronization Parameters. To desynchronize two targets, the attacker needs to send incompatible synchronization parameters that will result in a controllable offset. We explain how the attacker calculates the relevant parameters i and t_{AW} for both targets. Let us assume that the attack starts at some time T_s. An AF sent to the *first* target at some time T_{Tx} with $t = \left\lfloor \frac{T_{\text{Tx}} - T_s}{1024} \right\rfloor$ (in TU) will include the following parameters:

$$i = \left(\left\lfloor \frac{t \mod 64}{16} \right\rfloor + 4 \left\lfloor \frac{t}{64} \right\rfloor \right) \mod 2^{16} \quad \text{and,} \tag{16}$$

$$t_{\text{AW}} = 64 - t \mod 64 \quad . \tag{17}$$

For the *second* target, the attacker will calculate t as $t^{\phi} = \left\lfloor \frac{T_{\text{Tx}} - T_s - \phi}{1024} \right\rfloor$ and compute i^{ϕ}, t^{ϕ}_{AW} analogously to Eqs. (16) and (17). We verify the correctness of these calculations experimentally and show the resulting phase offset between two targets for a target phase $\phi = \tau/4$ in Fig. 10.

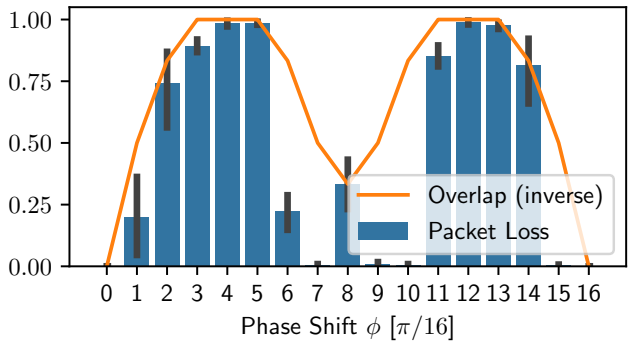


Figure 11: Packet loss for different phase shifts.

6.2 Evaluating Packet Loss

We evaluate the impact of our desynchronization attack by measuring the packet loss via the `ping` program. In particular, we use an APU board [66] equipped with a Qualcomm Atheros AR928X Wi-Fi card to act as an attacker which can inject AWDL AFs. The ICMP echo requests are sent from a MacBook Pro (Late 2015, macOS 10.13) to an iPhone 8 (iOS 12). The attacker induces different phase shifts spanning one period. The sender emits 100 ICMP echo requests per experiment which we repeat 10 times and plot the resulting packet loss in Fig. 11. The error bars indicate the standard deviation. In addition, we include the inverse channel overlap (see Section 2) which is calculated for two identical sequences which we have observed were the most common ones during our experiment: $44,44,44,0,0,0,0,0,6,44,44,0,0,0,0,0$.

At $\phi = 0$, there is no attack, while at $\phi = 16$, the targets are desynchronized by a full period which unsurprisingly does impair communication reliability. The other results indicate that the desynchronization attack significantly degrades communication between the targets, peaking at phase shifts where the targets are off by a quarter ($\phi = 4$) and three-quarter period ($\phi = 12$) which is where the channel overlap has its minima (and the inverse overlap its maxima). At these settings, the packet loss is almost 100 %. Surprisingly, some phase shifts (e. g., $\phi = 6,7,9,10,15$) result in less packet loss than the overlap predicts. We suspect the reason to be retransmissions on the MAC layer (up to 7 times in Wi-Fi [73]) which, at the cost of longer latency, increase the chance that a frame will be received in a subsequent AW.

6.3 Mitigating Desynchronization

Devices can mitigate our desynchronization attack by discarding unicast AFs. Not accepting unicast frames is an extremely effective and practical countermeasure because it will cause all nodes in range to process the same information exclusively. While this does not prevent an attacker from winning the master election and, thus, sending invalid synchronization parameters, as all nodes process the same frames, it becomes much harder to create a deterministic offset between two targets. A more sophisticated attacker could employ attacks on the PHY layer (e. g., using directional antennæ) to achieve a similar effect as that of unicasting. However, such attacks are difficult to carry out in practice.

6.4 Related Work: Reactive Jamming

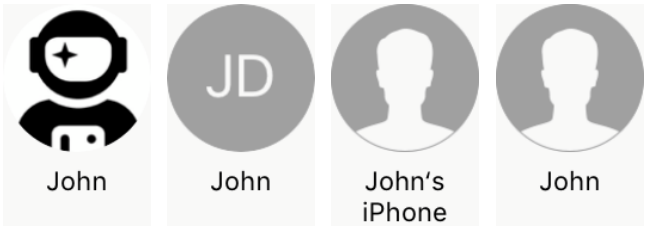
At first glance, our desynchronization attack achieves a similar effect as a *reactive jammer* [37, 52, 70, 91]. However, the desynchronization attack can be more attractive for two reasons: first, desynchronization in principle needs less energy than a jamming attack. The desynchronization attacker only needs to emit one frame every 1.5 s to maintain their position as a master node because AWDL nodes elect a new master if they have not received an AF for more than 1.5 s from their current one. In contrast, a reactive jammer needs to emit a jamming signal for every packet that the target sends. Second, it allows intercepting frames from its targets which enables to mount more sophisticated MitM attacks as presented in Section 7. In contrast, a normal jammer *kills* the frame in transit disallowing anyone (even the attacker themselves) to decode the frame [70]. There exist more sophisticated receiver designs that cancel out the jammer's own signal, but this typically requires special hardware [37]. Our desynchronization attack only requires a system with an off-the-shelf Wi-Fi chip and, thus, could even be implemented in a smartphone [71].

7 MitM: Planting Malware via AirDrop

This section describes a MitM attack on the popular AirDrop service which allows iOS and macOS devices to exchange files directly via AWDL. First, we assess the security of AirDrop and find that poor UI design choices enable an attacker to masquerade as a valid receiver. Then, we describe a complete MitM attack on AirDrop which prevents any sender to discover a valid receiver using a DoS attack and subsequently can intercept and modify any AirDrop file transmission. Finally, we discuss possible mitigations for the attack.

7.1 Ambiguous Receiver Authentication State

We have observed that AirDrop employs two different kinds of connections which we term *authenticated* and *unauthenticated* (see Section 3). Further, the user can set its device to be discoverable by *contacts only* or *everyone*. Counter-intuitively, the discoverability setting *only applies to the receiver side.* In particular, while a receiver in *contacts-only* mode will only accept files from authenticated senders, a sender will see all discoverable receivers irrespective of whether they are authentic or not. This ambiguity has profound implications for security because it is up to the user of the sending device to decide whether a connection is authenticated or not which can be non-trivial. The only visual cue to differentiate between an authenticated and unauthenticated connection is that an authenticated connection will show the receiver's name and photo from the sender's address book. Neither provides sufficient evidence to unambiguously decide whether a receiver



authentic	authentic	unauthentic	unauthentic
(a) Sender has John's contact photo	(b) Sender does not have John's contact photo	(c) Sender does not have John as a contact	(d) Attacker spoofs John's identity

Figure 12: UI representation of a receiver.

is authentic. First, if no contact photo is available (users augment only 27 % of their contacts with a photo according to our survey), the icon contains the receiver's initials in a grey circle which is similar to that of an unauthenticated receiver (a grey circle with a head's silhouette). Second, the name that is displayed underneath unauthenticated receivers is the receiver's device name. Based on our results in Section 5, a device name contains the user's given name in the majority of the cases (more than 70 % according to our experimental evaluation in Section 5), which the attacker can exploit to create a trustworthy-looking device name. Unless users are sensitive to such subtle UI changes, an attacker can easily trick them into sending files via an unauthenticated connection. Figure 12 compares the different receiver icons including a spoofed identity by the attacker. We want to highlight the similarity between an authenticated identity (Fig. 12b) and a spoofed identity (Fig. 12d).

7.2 The Complete AirDrop MitM Attack

Our MitM attack on AirDrop is carried out in three phases. First, we break the discovery process to put ourselves in a privileged position. Second, we wait until the target receiver becomes discoverable by *everyone*, effectively forcing the user to downgrade the connection. Third, we relay and manipulate the actual data transfer to plant arbitrary files at the receiver. We illustrate the attack in Fig. 13 and explain each phase in more detail below. Also, we provide a video PoC of the attack [77].

(1) Breaking Discovery via DoS. The most crucial part of the attack is preventing the sender to discover the receiver such that it appears as an icon in the sharing pane. In particular, we need to prevent that the *Discovery* handshake via HTTPS completes successfully. In principle, such a DoS attack could be carried out via our desynchronization attack (Section 6). However, we found that it could not reliably prevent the short *Discovery* request and responses from being received. This is due to the fact that AirDrop senders increase the channel allocation when starting the discovery process, thus, increasing the overlap with the receiver even when desynchronized. As an alternative, we used the well-known TCP reset attack which sends TCP segments including an RST flag to the targets which, in turn, immediately drops the connection. For this attack, the attacker sends out an RST reply for every TCP segment that is not addressed to itself and effectively prevents any reconnection attempts from the sender to the receiver.

(2) Downgrading an Authenticated Connection. For a complete MitM attack, we need to authenticate to the receiver. Otherwise, it will deny any *Ask* or *Upload* requests. If the receiver is discoverable by *everyone*, this is trivial, since it accepts all authentication attempts, even those with a self-signed certificate which the attacker can easily generate (see Section 3.3). The receiver indicates a successful authentication attempt from a non-contact by including its device name in the *Discover* response. However, we have found that in most cases (59.4 % in our survey), users set their device to *contacts only*. In such cases, we leverage the ongoing DoS attack to force the receiver to try the *everyone* setting.

(3) Relaying and Modifying Data Transfer. Once the receiver becomes discoverable (we can check when a receiver is discoverable by everyone by periodically sending *Discovery* requests), we advertise our own AirDrop identity via mDNS and wait until the sender tries to perform the authentication handshake via HTTPS for discovery which we let succeed. We relay the sender's *Ask* request to the receiver including the original file thumbnail to make the request appear valid. After the receiver accepts the transmission request, we relay the answer back to the sender which—in turn—starts to send the actual file. We can now decide whether to relay a modified version of the file or send an entirely new one possibly containing malware to the receiver.

7.3 Implementation

Our PoC of the MitM attack consists of two components. First, we re-implement AWDL such that we were able to overhear and parse data frames not addressed to us which is required to mount a TCP reset attack. Second, we implement an AirDrop-compatible client and server which we use to probe the discoverability status of the receiver target and finally implement the MitM attack as depicted in Fig. 13. We make both projects available as open source software [76, 78].

AWDL. Our AWDL implementation [76] is written in C and runs on Linux as well as on macOS. On macOS, the implementation can be used as a drop-in replacement for Apple's own AWDL interface. We use the monitor mode and frame injection of the system's Wi-Fi card to receive and inject raw IEEE 802.11 frames. In addition, we provide a virtual network interface (via *tuntap*) to the system such that any IPv6-capable application can use AWDL. Internally, our implementation takes care of frame parsing, synchronization, election, and scheduling data frames in the correct AWs.

AirDrop. Our AirDrop implementation [78] is written in Python and implements an unauthenticated AirDrop sender

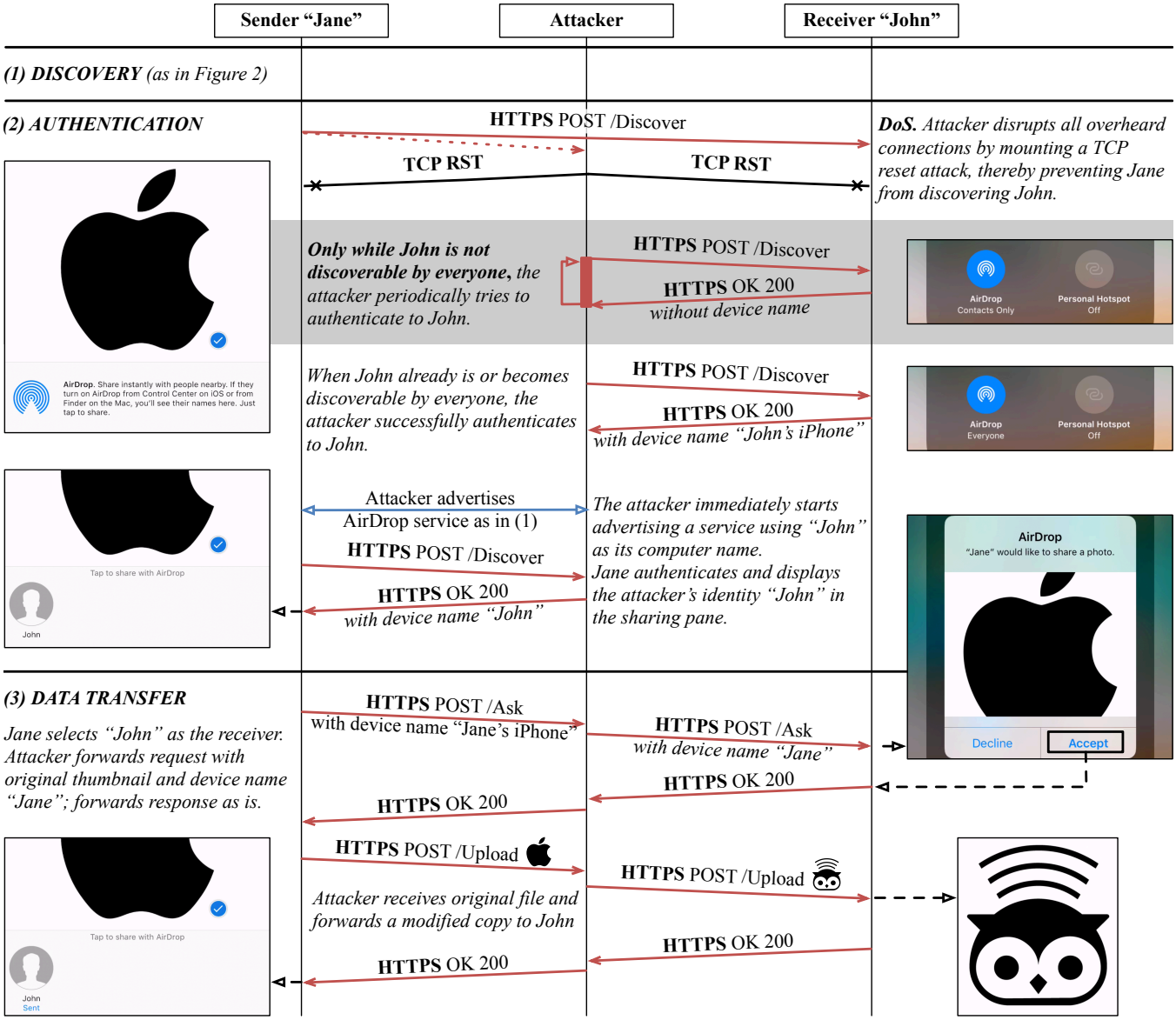


Figure 13: Protocol flow and user interaction of our MitM attack on AirDrop.

and receiver. The code exposes a command line interface which allows to *find* discoverable receivers, *send* files to them, and *receive* files from any sender.

7.4 Mitigation

We discuss possible mitigation strategies. We examine them according to complexity to implement, starting with the mitigation requiring the least number of changes to existing AirDrop implementations.

Provide Stronger Visual Cues for Authenticated Receivers. One of the core problems of the current design of AirDrop is that a user might have a hard time to differentiate between authenticated and unauthenticated receivers (see Section 7.1 and Fig. 12). Currently, the only cues to decide whether a receiver is authenticated are the display of a contact photo and contact name. We have shown that the former is not commonly available (users augment 27.4 % of their contacts with photos) and the latter can be spoofed. Therefore, we propose to provide stronger visual cues that unambiguously tell the user if a receiver is authenticated or not. This is already customary in web browsers where HTTPS-protected websites are augmented with a green (lock) symbol telling the user that the website they are visiting is authentic.

Reset *Everyone* to *Contacts Only* After a Timeout. Users might set the discoverability setting of their device to *everyone* for convenience or if they used it for one occasion and then forgot to reset it. In either case, we believe that the *everyone* setting should only be used except when it is required,

i. e., if one wants to receive a file from a non-contact. To protect negligent users, we propose to use a timeout after which the discoverability setting is reset to *contacts only*. Alternatively, one could reset the setting the next time the device is locked. This would also prevent past cases where people would receive inappropriate photographs from strangers in public places [13, 41] because, in *contacts-only* mode, devices will transparently reject all request from unauthenticated senders.

Introduce Secure AirDrop Mode for Non-Contacts. Our last proposal involves deprecating unauthenticated connections and instead establish authentication with a non-contact via an out-of-band (OOB) channel. AirDrop could transmit one-time cookies with similar functionality as the *record data* (see Section 3.3) during the initial HTTPS authentication handshake (see Section 3.2). The one-time cookies could be validated via an OOB channel such as NFC or via QR codes. After one transfer (or after a specific timeout), each device deletes its one-time cookie. By committing to the one-time cookie in the TLS handshake, a MitM attack on the OOB channel would be fruitless because the attacker could not establish a TLS connection with the same key. Unfortunately, such a mode would require one more manual step by both parties and, therefore, would impair usability.

7.5 Related Work: Attacks on AirDrop

Other attacks on AirDrop have been presented before. An impersonation attack [10] exploits mDNS/DNS-SD to redirect file transmissions to an attacker for unauthenticated connections. In particular, the attack uses forged SRV and AAAA responses to redirect an AirDrop ID to the attacker. In contrast to our work, [10] does not differentiate between *authenticated* and *unauthenticated* connections and claims that the UUID certificate (see Section 3.3) could not be bound to any contact identifiers, which we have found to be untrue. Also, the attack only works on unauthenticated connections, while our attack also targets authenticated connections via a downgrade attack and we present a complete MitM attack which allows an attacker to send malicious files to the receiver stealthily. Finally, [10] proposes a conflict detection mechanism for Multicast DNS (mDNS) to prevent their attack, which is based on the assumption that "disrupting two parties' communication through a Wi-Fi direct link or a local network is difficult for the adversary without access to the routing infrastructure of the network." In this work, we show that it is indeed practical to mount a DoS on the link layer since AWDL does not employ any security mechanisms. An earlier work [27] targeted a vulnerability in AirDrop's implementation which allowed the attacker to install files in arbitrary directories on the target's system. Apple fixed this bug in 2015.

8 Implementation Security

During our AWDL analysis and building an AWDL prototype, we found two implementation flaws in Apple's OSes that allow an attacker to crash devices in proximity.

DoS: Kernel Panic and System Crash. These flaws can be exploited by sending corrupt AFs. In particular, we can trigger kernel panics by setting invalid values in the synchronization parameters (affecting macOS 10.12) and the channel sequence (affecting macOS 10.14, iOS 12, watchOS 12, and tvOS 5), respectively. To showcase our findings, we provide a video of our PoC which exploits the second vulnerability on iOS devices [75]. The video demonstrates how an attacker mounts a targeted DoS attack that crashes a single device and a blackout DoS attack that crashes all devices in range of the attacker at the same time.

Outlook: Remote Code Execution. While not critical by themselves, the mere existence of these vulnerabilities creates a new class of threats to Wi-Fi devices as an attacker can exploit them *without any authentication towards the target*, i. e., they do not have to be on the same network. In light of past discovered remote code execution in implementations of standardized Wi-Fi procedures [8, 14], we think that a determined attacker can find similar flaws for AWDL.

9 Conclusion

The deployment of open Wi-Fi interfaces enables new types of applications for mobile devices. They allow devices in proximity to communicate with each other without being connected to the same Wi-Fi network. On the downside, this also opens new opportunities for an attacker as they no longer have to provide any kind of authentication (e. g., access to a secure Wi-Fi network). In this paper, we investigate the first protocol of this kind, i. e., Apple's proprietary AWDL. In particular, we find three distinct protocol-level vulnerabilities that allow for DoS, user tracking, and MitM attacks. In addition, we discovered two implementation bugs in Apple's OSes that cause DoS. Given the complexity of the protocol and implementations, we conjecture that more severe vulnerabilities will be found in the future. To build PoCs for these attacks, we have reverse-engineered AirDrop, a system service that runs on top of AWDL, and have implemented open versions of both AWDL and AirDrop which we make available as open source software. Finally, our findings have implications for the non-Apple world: NAN, commonly known as Wi-Fi Aware, is a new standard supported by Android which draws on AWDL's design and, thus, might be vulnerable to the similar attacks as presented in this work. This is pending further investigation.

Responsible Disclosure

We have contacted Apple about our findings on December 17, 2018. We have shared a draft of this paper as well as our PoC code and support Apple in fixing the privacy leaks (Section 5) and desynchronization issue (Section 6) in AWDL as well as the ambiguous authentication state in AirDrop (Section 7). We have reported the two implementation vulnerabilities (Section 8) earlier on August 14 and 27, 2018, respectively. Apple will not fix the first one affecting only macOS 10.12, but

has released software updates addressing the second one on October 30, 2018, for all Apple OSes [4].

Acknowledgements

This work is funded by the LOEWE initiative (Hesse, Germany) within the NICER project and by the German Federal Ministry of Education and Research (BMBF) and the State of Hesse within CRISP-DA. The work was partially supported by NSF grant 1740907. We thank the Apple Product Security team for feedback on the paper.

Availability

We release the source code of our AWDL [76] and AirDrop [78] implementations as part of the Open Wireless Link project [81] (https://owlink.org).

References

[1] Miguel E. Andrés, Nicolás E. Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Geo-indistinguishability: Differential Privacy for Location-based Systems. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, November 2013. doi: 10.1145/2508859.2516735.

[2] Apple Inc. Use AirDrop on your iPhone, iPad, or iPod touch, June 2018. URL https://support.apple.com/en-us/HT204144. [Accessed September 20, 2018].

[3] Apple Inc. Change the name of your iPhone, iPad, or iPod, October 2018. URL https://support.apple.com/en-us/HT201997. [Accessed November 14, 2018].

[4] Apple Inc. About the security content of iOS 12.1, 2018. URL https://support.apple.com/en-in/HT209192. [Accessed Feburary 14, 2019].

[5] Myrto Arapinis, Loretta Ilaria Mancini, Eike Ritter, and Mark Dermot Ryan. Analysis of Privacy in Mobile Telephony Systems. *International Journal of Information Security*, October 2017. doi: 10.1007/s10207-016-0338-9.

[6] Claudio A. Ardagna, Marco Cremonini, Sabrina De Capitani di Vimercati, and Pierangela Samarati. An Obfuscation-Based Approach for Protecting Location Privacy. *IEEE Transactions on Dependable and Secure Computing*, 8, January 2011. doi: 10.1109/TDSC.2009.25.

[7] Armis Inc. BlueBorne, 2018. URL https://armis.com/blueborne/. [Accessed November 14, 2018].

[8] Nitay Artenstein. Broadpwn: Remotely Compromising Android and iOS via a Bug in Broadcom's Wi-Fi Chipsets, July 2017. URL https://blog.exodusintel.com/2017/07/26/broadpwn/. [Accessed June 28, 2018].

[9] Michael Backes, Sven Bugiel, Christian Hammer, Oliver Schranz, and Philipp von Styp-Rekowsky. Boxify: Full-fledged App Sandboxing for Stock Android. In *USENIX Security Symposium*, August 2015.

[10] Xiaolong Bai, Luyi Xing, Nan Zhang, Xiaofeng Wang, Xiaojing Liao, Tongxin Li, and Shi-Min Hu. Staying Secure and Unprepared: Understanding and Mitigating the Security Risks of Apple ZeroConf. In *IEEE Symposium on Security and Privacy (S&P)*, May 2016. doi: 10.1109/SP.2016.45.

[11] Marco V. Barbera, Alessandro Epasto, Alessandro Mei, Vasile C. Perta, and Julinda Stefa. Signals from the Crowd: Uncovering Social Relationships through Smartphone Probes. In *ACM Internet Measurement Conference (IMC)*, October 2013. doi: 10.1145/2504730.2504742.

[12] Elad Barkan, Eli Biham, and Nathan Keller. Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication. In *Advances in Cryptology (CRYPTO)*, August 2003. doi: 10.1007/978-3-540-45146-4_35.

[13] Sarah Bell. Police investigate 'first cyber-flashing' case, 2015. URL https://www.bbc.com/news/technology-33889225. [Accessed September 25, 2018].

[14] Gal Beniamini. Over The Air: Exploiting Broadcom's Wi-Fi Stack (Part 2), April 2017. URL https://googleprojectzero.blogspot.com/2017/04/over-air-exploiting-broadcoms-wi-fi_11.html. [Accessed June 28, 2018].

[15] Laurent Bindschaedler, Murtuza Jadliwala, Igor Bilogrevic, Imad Aad, Philip Ginzboorg, Valtteri Niemi, and Jean-Pierre Hubaux. Track Me If You Can: On the Effectiveness of Context-based Identifier Changes in Deployed Mobile Networks. In *Network & Distributed System Security Symposium (NDSS)*, 2012.

[16] Vincent Bindschaedler and Reza Shokri. Synthesizing Plausible Privacy-Preserving Location Traces. In *IEEE Symposium on Security and Privacy (S&P)*, May 2016. doi: 10.1109/SP.2016.39.

[17] Bluetooth Special Interest Group (SIG). Bluetooth Specification Version 4.1. Technical report, December 2013.

[18] Nicolás E. Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Optimal Geo-Indistinguishable Mechanisms for Location Privacy. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, November 2014. doi: 10.1145/2660267.2660345.

[19] Stuart D. Cheshire. Proximity Wi-Fi. *U.S. Patent Application*, (US 2018/0083858 A1), March 2018. URL https://patents.google.com/patent/US20180083858A1.

[20] Aldo Cortesi, Maximilian Hils, Thomas Kriechbaumer, and contributors. mitmproxy: A free and open source interactive HTTPS proxy, 2010–. URL https://mitmproxy.org. [Version 3].

[21] Mathieu Cunche. I Know Your MAC Address: Targeted Tracking of Individual Using Wi-Fi. *Journal of Computer Virology and Hacking Techniques*, 10, November 2013.

[22] Anupam Das, Nikita Borisov, and Matthew Caesar. Tracking Mobile Web Users Through Motion Sensors: Attacks and Defenses. In *Network and Distributed System Security Symposium (NDSS)*, February 2016.

[23] Anupam Das, Nikita Borisov, and Edward Chou. Every Move You Make: Exploring Practical Issues in Smartphone Motion Sensor Fingerprinting and Countermeasures. *Proceedings on Privacy Enhancing Technologies (PoPETs)*, 2018, January 2018. doi: 10.1515/popets-2018-0005.

[24] Aveek K. Das, Parth H. Pathak, Chen-Nee Chuah, and Prasant Mohapatra. Uncovering Privacy Leakage in BLE Network Traffic of Wearable Fitness Trackers. In *ACM Workshop on Mobile Computing Systems and Applications (HotMobile)*, February 2016. doi: 10.1145/2873587.2873594.

[25] Sanorita Dey, Nirupam Roy, Wenyuan Xu, Romit Roy Choudhury, and Srihari Nelakuditi. AccelPrint: Imperfections of Accelerometers Make Smartphones Trackable. In *Network and Distributed System Security Symposium (NDSS)*, February 2014.

[26] Adriano Di Luzio, Alessandro Mei, and Julinda Stefa. Mind Your Probes: De-Anonymization of Large Crowds Through Smartphone WiFi Probe Requests. In *IEEE International Conference on Computer Communications (INFOCOM)*, April 2016. doi: 10.1109/INFOCOM.2016.7524459.

[27] Mark Dowd. MalwAirDrop: Compromising iDevices via AirDrop. In *Ruxcon*, October 2015. URL http://2015.ruxcon.org.au/slides/.

[28] William Enck, Peter Gilbert, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth. TaintDroid: An Information-flow Tracking System for Realtime Privacy Monitoring on Smartphones. In *USENIX Conference on Operating Systems Design and Implementation (OSDI)*, October 2010.

[29] Kassem Fawaz and Kang G. Shin. Location Privacy Protection for Smartphone Users. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, November 2014. doi: 10.1145/2660267.2660270.

[30] Kassem Fawaz, Huan Feng, and Kang G. Shin. Anatomization and Protection of Mobile Apps' Location Privacy Threats. In *USENIX Security Symposium*, August 2015.

[31] Kassem Fawaz, Kyu-Han Kim, and Kang G. Shin. Protecting Privacy of BLE Device Users. In *USENIX Security Symposium*, August 2016.

[32] Adrienne Porter Felt, Erika Chin, Steve Hanna, Dawn Song, and David Wagner. Android Permissions Demystified. In *ACM Conference on Computer and Communications Security (CCS)*, October 2011. doi: 10.1145/2046707.2046779.

[33] Julien Freudiger. How Talkative is Your Mobile Device?: An Experimental Study of Wi-Fi Probe Requests. In *ACM Conference on Security & Privacy in Wireless and Mobile Networks (WiSec)*, June 2015. doi: 10.1145/2766498.2766517.

[34] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. Show Me How You Move and I Will Tell You Who You Are. In *ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS (SPRINGL)*, 2010. doi: 10.1145/1868470.1868479.

[35] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez Del Prado Cortez. De-anonymization Attack on Geolocated Data. *Journal of Computer and System Sciences*, 80, December 2014. doi: 10.1016/j.jcss.2014.04.024.

[36] Clint Gibler, Jonathan Crussell, Jeremy Erickson, and Hao Chen. AndroidLeaks: Automatically Detecting Potential Privacy Leaks in Android Applications on a Large Scale. In *International Conference on Trust and Trustworthy Computing (TRUST)*. Springer, June 2012. doi: 10.1007/978-3-642-30921-2_17.

[37] Shyamnath Gollakota, Haitham Hassanieh, Benjamin Ransford, Dina Katabi, and Kevin Fu. They Can Hear Your Heartbeats: Non-Invasive Security for Implantable Medical Devices. In *ACM SIGCOMM Computer Communication Review*, October 2011. doi: 10.1145/2043164.2018438.

[38] Google. Wi-Fi Aware, 2017. URL https://developer.android.com/guide/topics/connectivity/wifi-aware. [Accessed June 28, 2018].

[39] Jun Han, Emmanuel Owusu, Le T. Nguyen, Adrian Perrig, and Joy Zhang. ACComplice: Location Inference Using Accelerometers on Smartphones. In *IEEE International Conference on Communication Systems and Networks (COMSNETS)*, January 2012. doi: 10.1109/COMSNETS.2012.6151305.

[40] Xiuping Han, Zhi Wang, and Dan Pei. Preventing Wi-Fi Privacy Leakage: A User Behavioral Similarity Approach. In *IEEE International Conference on Communications (ICC)*, May 2018. doi: 10.1109/ICC.2018.8422764.

[41] Harry Harris. Oakland-Maui flight: Pepper spray emergency follows disturbing photo, 2018. URL https://www.eastbaytimes.com/2018/09/01/oakland-maui-pepper-spray-disturbing-photo-delay/. [Accessed September 25, 2018].

[42] Benjamin Henne, Christian Kater, Matthew Smith, and Michael Brenner. Selective Cloaking: Need-to-Know for Location-based Apps. In *IEEE Conference on Privacy, Security and Trust*, July 2013. doi: 10.1109/PST.2013.6596032.

[43] Baik Hoh and Marco Gruteser. Preserving Privacy in GPS Traces via Uncertainty-aware Path Cloaking. In *ACM Conference on Computer and Communications Security (CCS)*, October 2007. doi: 10.1145/1315245.1315266.

[44] Byeongdo Hong, Sangwook Bae, and Yongdae Kim. GUTI Reallocation Demystified: Cellular Location Tracking with Changing Temporary Identifier. In *Network and Distributed System Security Symposium (NDSS)*, February 2018. doi: 10.14722/ndss.2018.23349.

[45] Jinseong Jeon, Kristopher K. Micinski, Jeffrey A. Vaughan, Ari Fogel, Nikhilesh Reddy, Jeffrey S. Foster, and Todd Millstein. Dr. Android and Mr. Hide: Fine-grained Permissions in Android Applications. In *ACM Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM)*, October 2012. doi: 10.1145/2381934.2381938.

[46] Ryo Kato, Mayu Iwata, Takahiro Hara, Akiyoshi Suzuki, Xing Xie, Yuki Arase, and Shojiro Nishio. A Dummy-based Anonymization Method Based on User Trajectory with Pauses. In *ACM Conference on Advances in Geographic Information Systems (SIGSPATIAL)*, November 2012. doi: 10.1145/2424321.2424354.

[47] Constantinos Kolias, Lucas Copi, Fengwei Zhang, and Angelos Stavrou. Breaking BLE Beacons For Fun But Mostly Profit. In *ACM European Workshop on Systems Security (EuroSec)*, April 2017. doi: 10.1145/3065913.3065923.

[48] John Krumm. Inference Attacks on Location Tracks. In *International Conference on Pervasive Computing (PERVASIVE)*. Springer, 2007.

[49] B. Krupp, N. Sridhar, and W. Zhao. SPE: Security and Privacy Enhancement Framework for Mobile Devices. *IEEE Transactions on Dependable and Secure Computing*, 14, July 2015. doi: 10.1109/TDSC.2015.2465965.

[50] Denis F. Kune, John Koelndorfer, Nicholas Hopper, and Yongdae Kim. Location Leaks Over the GSM Air Interface. In *Network & Distributed System Security Symposium (NDSS)*, February 2012.

[51] Chi-Yu Li, Guan-Hua Tu, Chunyi Peng, Zengwen Yuan, Yuanjie Li, Songwu Lu, and Xinbing Wang. Insecurity of Voice Solution VoLTE in LTE Mobile Networks. In *ACM Conference on Computer and Communications Security (CCS)*, October 2015. doi: 10.1145/2810103.2813618.

[52] Guolong Lin and Guevara Noubir. On Link Layer Denial of Service in Data Wireless LANs. *Wiley Journal on Wireless Communications and Mobile Computing*, 5, May 2005.

[53] Hua Lu, Christian S. Jensen, and Man Lung Yiu. PAD: Privacy-area Aware, Dummy-based Location Privacy in Mobile Services. In *ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE)*, June 2008. doi: 10.1145/1626536.1626540.

[54] Adriano Di Luzio, Alessandro Mei, and Julinda Stefa. Mind Your Probes: De-anonymization of Large Crowds Through Smartphone WiFi Probe Requests. In *IEEE INFOCOM*, April 2016. doi: 10.1109/INFOCOM.2016.7524459.

[55] Sathiamoorthy Manoharan. On GPS Tracking of Mobile Devices. In *IEEE International Conference on Networking and Services (ICNS)*, April 2009. doi: 10.1109/ICNS.2009.103.

[56] Célestin Matte, Mathieu Cunche, Franck Rousseau, and Mathy Vanhoef. Defeating MAC Address Randomization Through Timing Attacks. In *ACM Conference on Security & Privacy in Wireless and Mobile Networks (WiSec)*, July 2016. doi: 10.1145/2939918.2939930.

[57] Ulrike Meyer and Susanne Wetzel. A Man-in-the-Middle Attack on UMTS. In *ACM Workshop on Wireless Security (WiSe)*, October 2004. doi: 10.1145/1023646.1023662.

[58] Micro:bit Educational Foundation. Micro:bit website, 2018. URL https://microbit.org. [Accessed September 20, 2018].

[59] Arsalan Mosenia, Xiaoliang Dai, Prateek Mittal, and Niraj K. Jha. PinMe: Tracking a Smartphone User Around the World. *IEEE Transactions on Multi-Scale Computing Systems*, 3, 2017. doi: 10.1109/TMSCS.2017.2751462.

[60] A. B. M. Musa and Jakob Eriksson. Tracking Unmodified Smartphones Using Wi-fi Monitors. In *ACM Conference on Embedded Network Sensor Systems (SenSys)*, November 2012. doi: 10.1145/2426656.2426685.

[61] Sashank Narain, Triet D. Vo-Huu, Kenneth Block, and Guevara Noubir. Inferring User Routes and Locations Using Zero-Permission Mobile Sensors. In *IEEE Symposium on Security and Privacy (S&P)*, May 2016. doi: 10.1109/SP.2016.31.

[62] Sarfraz Nawaz and Cecilia Mascolo. Mining Users' Significant Driving Routes with Low-power Sensors. In *ACM Conference on Embedded Network Sensor Systems (SenSys)*, November 2014. doi: 10.1145/2668332.2668348.

[63] Nordic Semiconductor. nRF51822, 2018. URL https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF51822. [Accessed September 20, 2018].

[64] Simon Oya, Carmela Troncoso, and Fernando Pérez-González. Back to the Drawing Board: Revisiting the Design of Optimal Location Privacy-preserving Mechanisms. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, October 2017. doi: 10.1145/3133956.3134004.

[65] Paulo Borges. BLESSED, 2014. URL https://github.com/pauloborges/blessed. [Accessed September 20, 2018].

[66] PC Engines. APU2 platform, 2018. URL https://www.pcengines.ch/apu2.htm. [Accessed November 14, 2018].

[67] Zhiyun Qian, Zhaoguang Wang, Qiang Xu, Z. Morley Mao, Ming Zhang, and Yi-Min Wang. You Can Run, but You Can't Hide: Exposing Network Location for Targeted DoS Attacks in Cellular Networks. In *Network & Distributed System Security Symposium (NDSS)*, February 2012.

[68] Franziska Roesner. Designing Application Permission Models that Meet User Expectations. *IEEE Security & Privacy*, 15, February 2017. doi: 10.1109/MSP.2017.3.

[69] Piotr Sapiezynski, Arkadiusz Stopczynski, David Kofoed Wind, Jure Leskovec, and Sune Lehmann. Inferring Person-to-Person Proximity Using WiFi Signals. *ACM Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1, June 2017. doi: 10.1145/3090089.

[70] Matthias Schulz, Francesco Gringoli, Daniel Steinmetzer, Michael Koch, and Matthias Hollick. Massive Reactive Smartphone-based Jamming Using Arbitrary Waveforms and Adaptive Power Control. In *ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, July 2017. doi: 10.1145/3098243.3098253.

[71] Matthias Schulz, Daniel Wegemer, and Matthias Hollick. The Nexmon Firmware Analysis and Modification Framework: Empowering Researchers to Enhance Wi-Fi Devices. *Computer Communications*, 2018. doi: 10.1016/j.comcom.2018.05.015.

[72] Reza Shokri, George Theodorakopoulos, Carmela Troncoso, Jean-Pierre Hubaux, and Jean-Yves Le Boudec. Protecting Location Privacy: Optimal Strategy Against Localization Attacks. In *ACM Conference on Computer and Communications Security (CCS)*, October 2012. doi: 10.1145/2382196.2382261.

[73] IEEE Computer Society. Wireless LAN medium access control (MAC) and physical layer (PHY) specification, December 2016.

[74] Adam Stubblefield, John Ioannidis, and Aviel D. Rubin. Using the Fluhrer, Mantin, and Shamir Attack to Break WEP. In *Network and Distributed System Security Symposium (NDSS)*, February 2002.

[75] Milan Stute. Video of Proof-of-Concept Denial-of-Service Attack Crashing iOS Devices, 2018. URL https://youtu.be/M5D9NeKapUo.

[76] Milan Stute. Open Apple Wireless Direct Link Implementation in C, 2019. URL https://seemoo.de/owl.

[77] Milan Stute. Video of Proof-of-Concept Man-in-the-Middle Attack on AirDrop, 2019. URL https://youtu.be/5T7Qatoh0Vo.

[78] Milan Stute and Alexander Heinrich. Open AirDrop Implementation in Python, 2019. URL https://seemoo.de/opendrop.

[79] Milan Stute, David Kreitschmann, and Matthias Hollick. One Billion Apples' Secret Sauce: Recipe for the Apple Wireless Direct Link Ad hoc Protocol. In *ACM Conference on Mobile Computing and Networking (MobiCom)*, October 2018. doi: 10.1145/3241539.3241566.

[80] Milan Stute, David Kreitschmann, and Matthias Hollick. Demo: Linux Goes Apple Picking: Cross-Platform Ad hoc Communication with Apple Wireless Direct Link. In *ACM Conference on Mobile Computing and Networking (MobiCom)*, October 2018. doi: 10.1145/3241539.3267716.

[81] Milan Stute, David Kreitschmann, and Matthias Hollick. The Open Wireless Link Project, 2018. URL https://owlink.org.

[82] Akiyoshi Suzuki, Mayu Iwata, Yuki Arase, Takahiro Hara, Xing Xie, and Shojiro Nishio. A User Location Anonymization Method for Location Based Services in a Real Environment. In *ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS)*, November 2010. doi: 10.1145/1869790.1869846.

[83] Galini Tsoukaneri, George Theodorakopoulos, Hugh Leather, and Mahesh K. Marina. On the Inference of User Paths from Anonymized Mobility Data. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, March 2016. doi: 10.1109/EuroSP.2016.25.

[84] Jayakrishnan Unnikrishnan and Farid Movahedi Naini. De-anonymizing Private Data by Matching Statistics. In *IEEE Allerton Conference on Communication, Control, and Computing (Allerton)*, October 2013. doi: 10.1109/Allerton.2013.6736722.

[85] US Census Bureau. Frequently Occurring Surnames from the 2010 Census. URL https://www.census.gov/topics/population/genealogy/data/2010_surnames.html. [Accessed September 25, 2018].

[86] US Department of Commerce. US Census Bureau. URL https://www.census.gov. [Accessed September 25, 2018].

[87] US Social Security Administration. Popular Baby Names: Beyond the Top 1000 Names. URL https://www.ssa.gov/oact/babynames/index.html. [Accessed September 25, 2018].

[88] Mathy Vanhoef and Frank Piessens. Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2. In *ACM Conference on Computer and Communications Security (CCS)*, October 2017. doi: 10.1145/3133956.3134027.

[89] Mathy Vanhoef, Célestin Matte, Mathieu Cunche, Leonardo S. Cardoso, and Frank Piessens. Why MAC Address Randomization is not Enough: An Analysis of Wi-Fi Network Discovery Mechanisms. In *ACM Asia Conference on Computer and Communications Security (ASIA CCS)*, May 2016. doi: 10.1145/2897845.2897883.

[90] Tien Dang Vo-Huu, Triet Dang Vo-Huu, and Guevara Noubir. Fingerprinting Wi-Fi Devices Using Software Defined Radios. In *ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, July 2016. doi: 10.1145/2939918.2939936.

[91] Triet Dang Vo-Huu, Tien Dang Vo-Huu, and Guevara Noubir. Interleaving Jamming in Wi-Fi Networks. In *ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, July 2016. doi: 10.1145/2939918.2939935.

[92] Yu Wang, Dingbang Xu, Xiao He, Chao Zhang, Fan Li, and Bin Xu. L2P2: Location-aware Location Privacy Protection for Location-based Services. In *IEEE INFOCOM*, March 2012. doi: 10.1109/INFCOM.2012.6195577.

[93] Te-En Wei, Albert B. Jeng, Hahn-Ming Lee, Chih-How Chen, and Chin-Wei Tien. Android Privacy. In *IEEE Conference on Machine Learning and Cybernetics*, July 2012. doi: 10.1109/ICMLC.2012.6359654.

[94] Wi-Fi Alliance. Neighbor Awareness Networking Technical Specification Version 2.0. Technical report, 2017.

[95] Hao Wu, Weiwei Sun, and Baihua Zheng. Is Only One Gps Position Sufficient to Locate You to the Road Network Accurately? In *ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*, 2016. doi: 10.1145/2971648.2971702.

[96] Yunze Zeng, Parth H. Pathak, and Prasant Mohapatra. WiWho: Wifi-based Person Identification in Smart Spaces. In *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, April 2016. doi: 10.1109/IPSN.2016.7460727.

[97] Jin Zhang, Bo Wei, Wen Hu, and Salil S. Kanhere. WiFi-ID: Human Identification Using WiFi Signal. In *IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, May 2016. doi: 10.1109/DCOSS.2016.30.

[98] Lichen Zhang, Zhipeng Cai, and Xiaoming Wang. FakeMask: A Novel Privacy Preserving Approach for Smartphones. *IEEE Transactions on Network and Service Management*, 13, June 2016. doi: 10.1109/TNSM.2016.2559448.