DEMO: AirCollect: Efficiently Recovering Hashed Phone Numbers Leaked via Apple AirDrop

(Full Version)*

Alexander Heinrich
TU Darmstadt, Germany
aheinrich@seemoo.de

Matthias Hollick
TU Darmstadt, Germany
mhollick@seemoo.de

Thomas Schneider
TU Darmstadt, Germany
schneider@encrypto.cs.tu-darmstadt.de

Milan Stute
TU Darmstadt, Germany
mstute@seemoo.de

Christian Weinert
TU Darmstadt, Germany
weinert@encrypto.cs.tu-darmstadt.de

ABSTRACT

Apple's file-sharing service AirDrop leaks phone numbers and email addresses by exchanging vulnerable hash values of the user's own contact identifiers during the authentication handshake with nearby devices. In a paper presented at USENIX Security'21, we theoretically describe two attacks to exploit these vulnerabilities and propose "PrivateDrop" as a privacy-preserving drop-in replacement for Apple's AirDrop protocol based on private set intersection.

In this demo, we show how these vulnerabilities are efficiently exploitable via Wi-Fi and physical proximity to a target. Privacy and security implications include the possibility of conducting advanced spear phishing attacks or deploying multiple "collector" devices in order to build databases that map contact identifiers to specific locations. For our proof-of-concept, we leverage a custom rainbow table construction to reverse SHA-256 hashes of phone numbers in a matter of milliseconds. We discuss the trade-off between success rate and storage requirements of the rainbow table and, after following responsible disclosure with Apple, we publish our proof-of-concept implementation as "AirCollect" on GitHub.

KEYWORDS

privacy, personal information, hashing, rainbow table, iOS, macOS

1 INTRODUCTION

Apple AirDrop is a file-sharing service that allows users to send photos and other media over a direct Wi-Fi connection from one Apple device to another. As people typically want to share sensitive data exclusively with people they know, AirDrop only shows receiver devices from address book contacts by default. To determine whether the other party is a contact, AirDrop uses a mutual authentication mechanism that compares a user's phone number and email address with entries in the other user's address book [10].

2 VULNERABILITIES

In our paper [6], we discovered two severe privacy leaks in this authentication mechanism. In particular, we showed that it is possible to learn the phone numbers and email addresses of AirDrop users—even as a complete stranger. An attacker only requires a Wi-Fi-capable device[1] and physical proximity to a target.

The discovered problems are rooted in Apple's use of hash functions for "obfuscating" the exchanged contact identifiers, i.e., phone numbers and email addresses, during the discovery process. It is well-known in industry and academia that hashing fails to provide privacy-preserving contact discovery since hash values of phone numbers can be quickly reversed using simple techniques such as brute-force attacks or database lookups [4].

2.1 Sender Leakage

During the AirDrop authentication handshake, the sender always discloses their own hashed contact identifiers as part of an initial discover message. A malicious receiver can therefore learn all hashed contact identifiers of the sender without requiring any prior knowledge of their target.

To obtain these identifiers, an attacker simply can wait (e. g., at a public hot spot) until a target device scans for AirDrop receivers, i. e., the user opens the sharing pane. After collecting the hashed contact identifiers, the attacker can recover phone numbers and email addresses offline. As shown in prior work [4], recovering phone numbers is possible in the order of milliseconds. Recovering email addresses is less trivial but possible via dictionary attacks that check common email formats such as first.lastname@{gmail.com, yahoo.com, …}. Alternatively, an attacker could utilize data breaches or use an online lookup service for hashed email addresses [2].

This vulnerability was also independently discovered and published by the Apple Bleee project in July 2019 [1], shortly after our initial responsible disclosure to Apple in May 2019.

2.2 Receiver Leakage

AirDrop receivers present their hashed contact identifiers in response to the discover message if they know any of the sender's contact identifiers (e. g., if the receiver has stored the sender's email address). A malicious sender can thus learn all contact identifiers (including the receiver's phone number) without requiring any prior knowledge of the receiver—if the receiver knows the sender.

*Please cite the conference version of this demo paper published at ACM WiSec'21 [5].

[1] AirDrop relies on a proprietary Wi-Fi-based link-layer protocol called Apple Wireless Direct Link (AWDL) [9].

Table 1: Structure of the validation record certificate exchanged during the AirDrop authentication handshake. Problematic fields are highlighted in bold.

Field name	Content
Version	2
altDsID	UUID of Apple account
encDsID	UUID of Apple account
SuggestValidDuration	2592000 seconds (= 30 days)
ValidAsOf	date and time
ValidatedEmailHashes	array of SHA-256 hashes
ValidatedPhoneHashes	array of SHA-256 hashes

Importantly, the malicious sender does not have to know the receiver: A popular person within a certain context (e. g., the manager of a company) can exploit this design flaw to learn all (private) contact identifiers of other people who have the popular person in their address book (e. g., employees of the company).

3 PROOF-OF-CONCEPT ATTACKS

We demonstrate two attacks exploiting the vulnerabilities with a proof-of-concept implementation called "AirCollect" that is publicly available on GitHub (cf. §5). It combines the efforts of OpenDrop [8], an open-source AirDrop implementation, with RainbowPhones [3], an open-source rainbow table implementation that is optimized for non-uniform input domains such as mobile phone numbers.

Extracting Phone Number Hashes from Certificates. We extend the OpenDrop [8] implementation with a module that parses the validation record certificates exchanged during the authentication handshake. We depict the structure of validation record certificate in Table 1, which is identical for both sender and receiver devices. Our proof-of-concept code parses the array of phone hashes and passes them to the RainbowPhones tool for lookup.

Rainbow Table: Success Rate vs. Storage. We use the open-source tool RainbowPhones [3] to efficiently find the preimage to a given hashed phone number. Compared to a conventional rainbow table implementation, RainbowPhones features a specialized reduction function that considers the non-uniform input domain of phone numbers [4]. To make RainbowPhones usable for recovering hashes exchanged in the AirDrop protocol, we added support for the SHA-256 hash algorithm.

For this demo, we compute a total 5 tables (each 15.3 MB in size) for German mobile phone numbers that together achieve a success rate of 99.8 % (measured by reversing 10 k hashes of randomly chosen German mobile phone numbers). In our published proof-of-concept implementation, we *omit* these precomputed tables that would allow attackers to reverse almost any given phone number hash in a matter of milliseconds. In Fig. 1, we analyze the trade-off between success rate and storage requirements, finding that even only a single table already achieves 71 % success rate, making the approach attractive for small embedded devices. The figure also includes similar measurements for 5 tables precomputed for all phone numbers in the US that with a total size of 765 MB achieve 100 % success rate (again measured over 10 k randomly chosen samples).

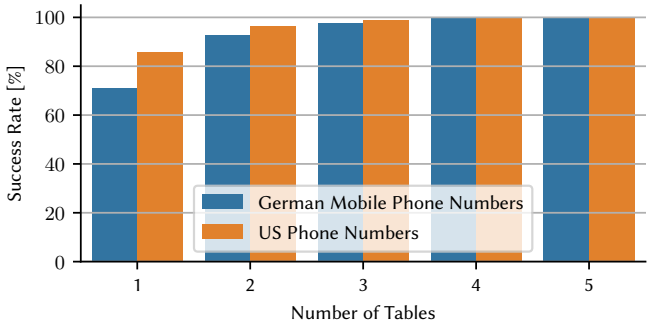

Figure 1: Analysis of success rate for SHA-256 hash reversal with RainbowPhone [3] when combining an increasing number of equally-sized precomputed rainbow tables for German mobile and all US phone numbers.

Exploiting Sender Leakage in Practice. Finally, we can execute an attack to exploit sender leakage via a single command-line call on a MacBook.[2] The OpenDrop program starts listening for active AirDrop senders, e. g., iPhones with an open sharing pane, and immediately logs discovered phone numbers to the standard output of the console. An example output would look as follows:

```
$ python3 -m opendrop receive
Announcing service: host opendrop, address fe80::
    c8b9:fbff:fee9:d544, port 8771
Starting HTTPS server
Nearby phone number: +49<...>
```

Our test setup is depicted in Fig. 2.

[2] Alternatively, we can also use a Linux-based machine such as a Raspberry Pi, but the setup then also requires running an open version of the AWDL protocol [10].

Figure 2: Test setup for demonstrating sender leakage. The iPhone user (right) opens the sharing pane on the device to share a document with the person in the background. The attacker (left) running "AirCollect" on a MacBook immediately sees the sender's phone number.

Exploiting Receiver Leakage in Practice. Similarly, we demonstrate how to exploit receiver leakage, which does not require any interaction with the target. However, we must be able to present a valid AirDrop certificate containing contact identifiers known to the target. For this, we leverage a tool to extract valid AirDrop certificates to be used with AirCollect [7]. An example output after executing the attack looks as follows:

```
$ python3 -m opendrop find
Looking for receivers. Press Ctrl+C to stop ...
Nearby phone number: +49<...>
Found  index 0  ID a019<...>  name John's iPhone
```

4 RELATED WORK

The "Apple Bleee" project [1] independently discovered the issue of *sender leakage* and published their findings including a proof-of-concept implementation—two months after our initial disclosure to Apple in May 2019. Their implementation is also based on OpenDrop [8] and uses database lookups to reverse hashes. In contrast, we rely on a custom rainbow table construction that represents an interesting computation/storage trade-off and makes on-the-fly recovery even feasible for small embedded devices. Furthermore, in our demo, we practically show that the recently discovered *receiver leakage* vulnerability can be exploited as well.

5 CONCLUSION

We demonstrated how easy and with how little resources an attacker can learn private information (especially phone numbers) of AirDrop users in proximity. We leave the extension of our prototype to incorporate efficient reversal of email addresses as future work (e. g., by including calls to existing online services [2]).

We responsibly disclosed the issue to Apple in May 2019 as well as a practical solution [6] in October 2020. As of May 2021, Apple has not indicated if they are working on mitigating this issue. This means that Apple users are still vulnerable to the described attacks.

AVAILABILITY

Our proof-of-concept implementation "AirCollect" is available online at https://privatedrop.github.io.

ACKNOWLEDGMENTS

We thank Ann-Kathrin Braun and Daniela Fleckenstein for providing the photograph of our proof-of-concept setup. Also, we thank Christoph Hagen and Christoph Sendner for answering questions about RainbowPhones.

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 850990 PSOTI). It was co-funded by the Deutsche Forschungsgemeinschaft (DFG) – SFB 1119 CROSSING/236615297 and GRK 2050 Privacy & Trust/251805230, by the LOEWE initiative (Hesse, Germany) within the emergenCITY center, by the German Federal Ministry of Education and Research and the Hessian State Ministry for Higher Education, Research and the Arts within ATHENE.

REFERENCES

[1] Dmitry Chastuhin. *Apple Bleee: Everyone Knows What Happens on Your iPhone*. July 25, 2019. URL: https://hexway.io/research/apple-bleee/ (visited on 10/15/2020).

[2] Datafinder. *Recover Encrypted Email Addresses*. 2020. URL: https://web.archive.org/web/20191211152224/https://datafinder.com/products/email-recovery (visited on 10/15/2020).

[3] Christoph Hagen and Sebastian Schindler. *RainbowPhones*. 2021. URL: https://github.com/contact-discovery/rt_phone_numbers.

[4] Christoph Hagen, Christian Weinert, Christoph Sendner, Alexandra Dmitrienko, and Thomas Schneider. "All the Numbers are US: Large-scale Abuse of Contact Discovery in Mobile Messengers". In: *NDSS*. 2021. URL: https://www.ndss-symposium.org/wp-content/uploads/ndss2021_1C-3_23159_paper.pdf.

[5] Alexander Heinrich, Matthias Hollick, Thomas Schneider, Milan Stute, and Christian Weinert. "DEMO: AirCollect: Efficiently Recovering Hashed Phone Numbers Leaked via Apple AirDrop". In: *WiSec*. ACM, 2021. DOI: 10.1145/3448300.3468252.

[6] Alexander Heinrich, Matthias Hollick, Thomas Schneider, Milan Stute, and Christian Weinert. "PrivateDrop: Practical Privacy-Preserving Authentication for Apple AirDrop". In: *USENIX Security Symposium*. 2021. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/heinrich.

[7] Milan Stute. *Extracting Apple ID Validation Record, Certificate, and Key for AirDrop*. 2020. URL: https://github.com/seemoo-lab/airdrop-keychain-extractor.

[8] Milan Stute and Alexander Heinrich. *OpenDrop: An Open Source AirDrop Implementation*. 2019. URL: https://github.com/seemoo-lab/opendrop.

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

[10] Milan Stute, Sashank Narain, Alex Mariotto, Alexander Heinrich, David Kreitschmann, Guevara Noubir, and Matthias Hollick. "A Billion Open Interfaces for Eve and Mallory: MitM, DoS, and Tracking Attacks on iOS and macOS Through Apple Wireless Direct Link". In: *USENIX Security Symposium*. 2019. URL: https://www.usenix.org/conference/usenixsecurity19/presentation/stute.