Apple ZeroConf Holes: How Hackers Can Steal iPhone Photos

Xiaolong Bai | Tsinghua University
Luyi Xing, Nan Zhang, and XiaoFeng Wang | Indiana University Bloomington
Xiaojing Liao | Georgia Tech
Tongxin Li | Peking University
Shi-Min Hu | Tsinghua University

Usability-oriented zero configuration (ZeroConf) designs, with automatic service discovery "plug-and-play" techniques, give rise to security risks. A study focusing on Apple—a major proponent of ZeroConf—brings to light a disturbing lack of security: major ZeroConf components are mostly unprotected, and popular apps and system services are vulnerable to man-in-the-middle attacks.

With the proliferation of portable computing systems, such as tablets, smartphones, and other Internet of Things (IoT) devices, ordinary users face an increasing burden to properly configure those devices as they work together. In response to this utility challenge, major device manufacturers and software vendors, including Apple, Microsoft, and Hewlett-Packard, tend to build their systems in a "plug-and-play" fashion, using *zero configuration* (ZeroConf) techniques. For example, the AirDrop service on iPhone, once activated, automatically detects a nearby Apple device running the service to transfer documents or photos. Such ZeroConf techniques can automatically assign IP addresses to devices, resolve hostnames of other devices, and discover available services on the local network. In addition to those working on the IP network, similar techniques have been developed for automatic service discovery on other channels—Bluetooth in particular.[1]

When the design pendulum swings toward usability, concerns arise as to whether the system has been adequately protected. To understand whether the protection those systems receive is commensurate with the threats they're facing, we performed a security analysis on popular ZeroConf systems on Apple iOS and Mac OS X platforms. We focus on Apple because it's a main advocate of ZeroConf techniques and is known for its rigorous security control. In our study, we inspected popular apps and system services to understand whether they're properly guarded against realistic attacks.

Given the strong demand for ZeroConf techniques, it's critical to come up with usable solutions to address their security risks. We made a first step toward this end. We first designed a conflict detection technique that checks whether a ZeroConf network is attack free. And then we proposed a more generic solution, called Speak Out Your Certificate (SPYC), that binds an Apple account certificate to its human owner. We analyzed the SPYC's security design and evaluated the mechanism's usability and security through two human subject studies with 60 participants.

Related Work on ZeroConf and Bluetooth Security

Security threats affecting Link-Local Multicast Name Resolution (LLMNR), a zero configuration (ZeroConf) protocol used in Microsoft Windows, have been mentioned in technical blogs[1–4] and Internet Engineering Task Force documentation.[5] Unlike Bonjour, LLMNR isn't designed for automatic service discovery but instead just supports name resolution.[6] In addition to spoofing name resolution, our study on Bonjour also focuses on the service discovery stage and the fundamental challenge in protecting it with TLS.

Related to Bluetooth ZeroConf are works on the security of the devices without input capabilities (for instance, no keyboard or display).[7,8] However, our research is the first to investigate how the Just Works and out-of-band pairing modes are supported on Apple's Core Bluetooth framework.[9]

References

1. J. Sternstein, "Local Network Attacks: LLMNR and NBT-NS Poisoning," Stern Security, 16 Nov. 2013; www.sternsecurity.com/blog/local-network-attacks-llmnr-and-nbt-ns-poisoning.
2. "Local Network Vulnerabilities—LLMNR and NBT-NS Poisoning," SureCloud Newsletter, 9 Mar. 2015; www.surecloud.com/newsletter/local-network-vulnerabilities-llmnr-and-nbt-ns-poisoning.
3. "LLMNR Spoofer," Rapid7 Vulnerability & Exploit Database; www.rapid7.com/db/modules/auxiliary/spoof/llmnr/llmnr_response.
4. "Responder," Spider Labs; github.com/SpiderLabs/Responder.
5. H. Rafiee, "Multicast DNS (mDNS) Threat Model and Security Consideration," Internet Eng. Task Force, 10 June 2014; tools.ietf.org/html/draft-rafiee-dnssd-mdns-threatmodel-00.
6. M. Krochmal, "LLMNR, mDNS and mDNS Responders in Windows," Apple Mailing Lists, 16 Apr. 2004; lists.apple.com/archives/rendezvous-dev/2004/Apr/msg00031.html.
7. K. Hypponen and K.M. Haataja, "Nino Man-in-the-Middle Attack on Bluetooth Secure Simple Pairing," *Proc. 3rd IEEE/IFIP International Conf. Central Asia* (ICI 07), 2007, pp. 1–5.
8. K.M. Haataja and K. Hypponen, "Man-in-the-Middle Attacks on Bluetooth: A Comparative Analysis, a Novel Attack, and Countermeasures," *Proc. 3rd Int'l Symp. Communications, Control and Signal Processing* (ISCCSP 08), 2008, pp. 1096–1102.
9. "Security, Bluetooth Smart (Low Energy)," Bluetooth, 2015; developer.bluetooth.org/TechnologyOverview/Pages/LE-Security.aspx.

Background

The concept of zero configuration (www.zeroconf.org) was first defined over the IP network to set up a network without manual configuration. To this end, techniques were developed to self-assign IP addresses to networked devices; resolve conflicts; announce a hostname and its IP address; and automatically discover services of interest broadcasted by other devices, letting users choose services through browsing. Later, automatic service discovery was applied to bootstrap the devices running on other channels. We present two examples to show how ZeroConf works—Bluetooth low-energy (BLE) service discovery and IP service discovery. (For more information on ZeroConf and Bluetooth security, see the sidebar.)

First, BLE, a new Bluetooth technology, has been incorporated into iOS and OS X. BLE communication involves two main actors: a server that advertises and provides services and a client that discovers and uses these services. Each actor is identified by a universally unique identifier (UUID).

For the client to get services from the server, the two devices need to pair, a process that establishes a shared secret between them. This shared secret is used for authentication—that is, whoever possesses this secret is the right party. Traditionally, to pair, users are required to manually enter a PIN code. BLE comes with several new ZeroConf pairing methods that don't require such manual configuration. One pairing method, Just Works, enables a client to directly pair with a server. Another pairing method, out of band (OOB), automatically exchanges shared secrets in channels other than Bluetooth, for instance, through near-field communication or Apple's iCloud account. Apple's Core Bluetooth framework further hides the details of the pairing from both users and developers, which by default, takes care of the pairing process, easing both usage and development burdens.

Second, we look at IP service discovery. A prominent example of ZeroConf on the IP network is Apple's Bonjour protocol, which devices use to publish and discover services in a LAN. Bonjour assigns IP addresses and hostnames to devices without users' manual configuration and enables devices to automatically discover available services in a local network. All the user needs to do is choose, from a list, the service he or she wants to use.

Every service on a device has a unique service instance name, and every device is identified by its hostname. As an example, when publishing an AirDrop service, a Bonjour server broadcasts to the local network to register its unique service instance name, for example, 9c5e3d2, and hostname, for example, Jeffs-iPhone, which enables other devices to discover and access it.

A Bonjour client must go through two phases, discovery and resolution, to find and access a service of interest. In the discovery phase, the client broadcasts discovery requests to look for services of a specific

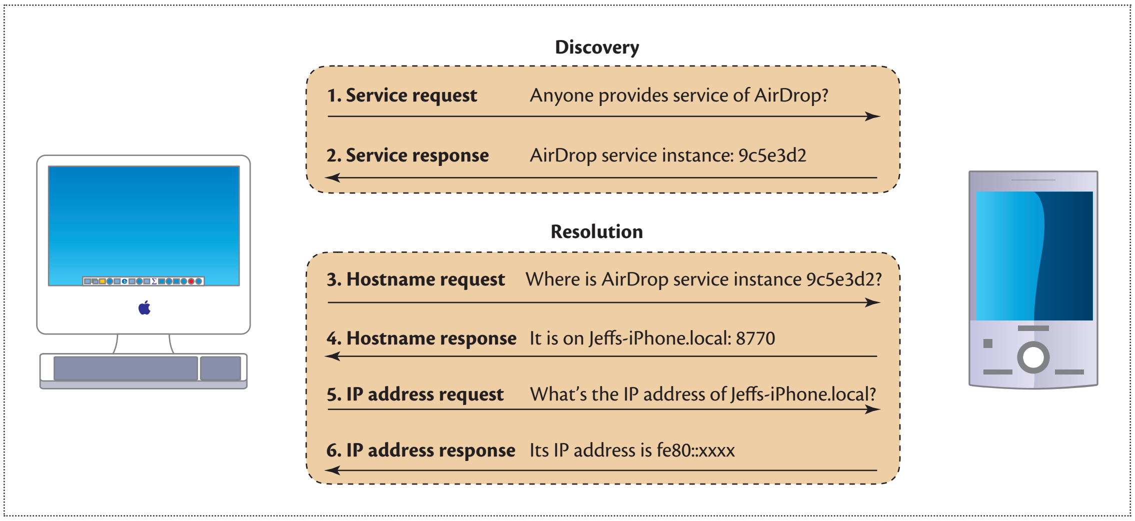

Discovery
1. Service request
Anyone provides service of AirDrop?
2. Service response
AirDrop service instance: 9c5e3d2
Resolution
3. Hostname request
Where is AirDrop service instance 9c5e3d2?
4. Hostname response
It is on Jeffs-iPhone.local: 8770
5. IP address request
What's the IP address of Jeffs-iPhone.local?
6. IP address response
Its IP address is fe80::xxxx

Figure 1. Bonjour service discovery and host resolution. In the discovery phase, the client broadcasts to request the server's service instance name. In the resolution phase, the client broadcasts again to query the server's hostname and IP address.

type, for instance, AirDrop. The Bonjour server—for instance, Jeff's iPhone—then responds with its service instance name, 9c5e3d2. Next, in the resolution phase, the client broadcasts again to query the hostname and IP address of the server of interest. With the IP address, the client can connect to the server and use its service. This resolution phase occurs each time the service instance name is used to find the server's current address and port number. Apple recommends saving the service instance name discovered (for instance, "HP Printer [928FE5]" of an HP printer), because it's relatively stable, unlike hostnames, IP addresses, and so on, which change frequently. Figure 1 illustrates the process of Bonjour service discovery.

In our adversary model, we assume that the attacker has already infected a device with malware, in an attempt to use the device to collect sensitive information from other uninfected devices. Such an adversary could not only listen on the communication channel (for example, BLE, LAN, or Wi-Fi direct) but also actively send out messages to impersonate a legitimate and uninfected device. We later demonstrate that such an adversary can perform a man-in-the-middle (MITM) attack, intercepting data transferred between nearby uninfected devices, although the infected device isn't the right data recipient. On the other hand, we don't consider a targeted attack on owners of an uninfected device, in which an adversary studies the owners' behavior and background, or even uses social engineering to collect information about them.

Understanding Apple ZeroConf

We conducted a security analysis on popular ZeroConf Apple services and apps to understand whether they're properly protected and, if not, which technical hurdles must be overcome to put protection in place. Our study reveals that most Apple ZeroConf systems, including Handoff, printer discovery, AirDrop, and other high-profile apps, are unguarded, subject to various MITM or data-stealing attacks.

Breaking Bluetooth ZeroConf

Apple integrates BLE ZeroConf techniques into its frameworks and system services. Many popular apps and services have adopted these techniques to improve usability. However, our study shows that their service discovery and pairing methods are often problematic, making many Apple apps and system services vulnerable to MITM attacks. Here, we discuss two examples.

Insecure pairing. Apple's Core Bluetooth framework lets iOS and Mac apps automatically discover and pair with other BLE devices. To ease the development process, this framework hides low-level BLE details from developers, such as which pairing mode to choose. Its default pairing mode (an abstraction of Just Works) is also designed to reduce users' burden, avoiding manual input of PINs (like traditional Bluetooth). We found that this default mode doesn't authenticate the client and the server. Therefore, apps adopting this framework are typically unprotected. For example, we studied Scribe, a free app that transfers a copied item from Mac to iPhone, which we found was vulnerable to MITM attacks.[2]

Attacking Handoff. Unlike Just Works, the OOB mechanism lets ZeroConf devices authenticate each other over the BLE channel. A prominent example is Apple

Handoff, a service that lets iOS and OS X synchronize data through BLE without configuration. Pairing between the devices happens through OOB: when users log in to their iCloud account on their Mac or iPhone, the devices' UUIDs and credentials are exchanged through their account to ensure that only authorized devices are paired.

The problem is that data synchronization should happen only between specific server/client apps, but the Apple's ZeroConf design doesn't provide authentication at the app level. As a result, any advertised BLE service on the iPhone is completely exposed to any BLE-capable app on the Mac. Specifically, we successfully exploited the Apple Notification Center Service (ANCS) on the iPhone using a sandboxed Mac app. iPhone's ANCS is responsible for managing all notifications. In the attack, as soon as a Bluetooth connection is established between the Mac and the iPhone (which happens when the user launches a Handoff process with the Handoff setting on), the attack app can discover the advertised ANCS service on the phone. By registering with the ANCS service, the attacker is informed whenever a notification appears on the iPhone and then acquires the notification from it. In this way, we found that the sandboxed app, with only the Bluetooth permission, stole all notifications from the iPhone, including SMS, emails, and instant messages. Such a malicious app bypassed the vetting of Apple's Mac App Store and got published. A demo is online (sites.google.com/site/applezeroconf).

After we reported our findings to Apple, it decided to discontinue support for transferring iOS notifications to Mac OS in the versions following 10.10.4.

Exploiting File-Sharing Apps

An important support provided by ZeroConf techniques is file sharing between devices, such as Macbook and iPhone, across an ad hoc network (local Wi-Fi network or peer-to-peer Wi-Fi direct connections) when the Internet is unavailable or considered to be less economical for the amount of data to be transferred. Apple provides an easy-to-use ZeroConf framework for file-sharing apps, called Multipeer Connectivity (MC; developer.apple.com/reference/multipeerconnectivity), which supports automatic service advertisement, discovery, target host resolution, and file transfer between devices across Wi-Fi and Bluetooth.

Typically, the file receiver device runs an MC advertiser interface to advertise an identifier object peerID and other information, which is picked up by the sender running another MC browser interface. The problem here is that an attack device can also browse and acquire the advertised peerID of a victim receiver, and then launch a service using exactly the same peerID object, to impersonate the receiver to the sender. Furthermore, the browser interface on the sender side considers the discovered peerID from the attacker as an update to the existing peerID from the victim receiver. Consequently, it will map this peerID to the attacker's IP address, enabling MITM attacks.

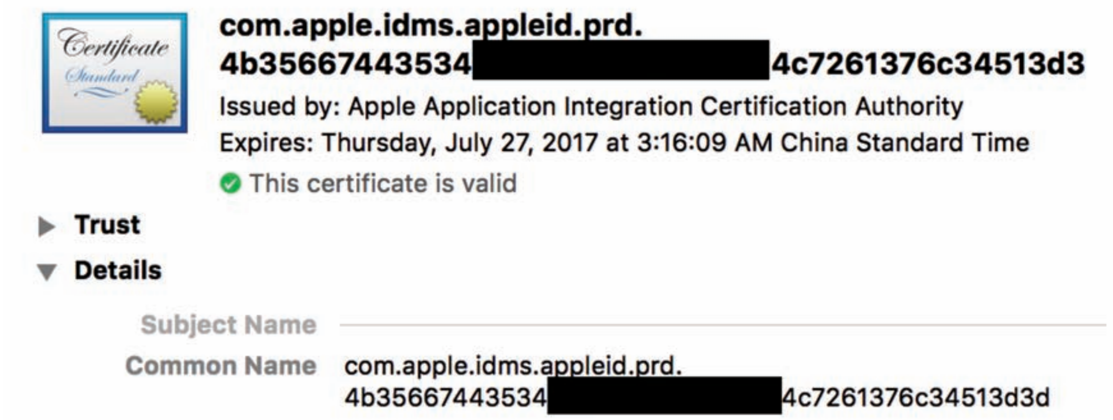
We also found that some file-sharing apps implement their own ZeroConf capabilities, which become necessary when file transfer needs to happen across platforms and therefore can't rely solely on Apple's service. A prominent example is Filedrop, a popular paid app designed to quickly share documents among iOS, Mac, Android, and Windows devices in a Wi-Fi ad hoc network. We found that although the app provides cryptographic protection for the file transfer process, it's still vulnerable to MITM attacks, which highlights the challenge in protecting an automatic, self-configured service in the absence of a preshared secret. Details of our attacks on these popular file-sharing apps, including Filedrop and the popular instant messaging app Tencent QQ, are explained elsewhere.[2]

Cracking Bonjour Protection

As we mentioned, Bonjour is a major ZeroConf mechanism developed by Apple. It supports automatic service discovery and hostname/IP resolution. In the discovery phase, the Bonjour client broadcasts requests to discover services of specific types (for instance, printing), then the server (for instance, an HP printer) responds with a service instance name such as "HP Printer [928FE5]." In the resolution step, the client resolves the server's IP address and hostname, for instance, LaserJet.local.

A problem for this fully automated mechanism is that, again, little protection is in place to ensure that the parties involved properly authenticate each other. With this weakness, the mechanism is still used in a not-fully-trusted environment, in the absence of additional security measures. Actually, even when people want to protect it, it's difficult to provide authentication on top of Bonjour without preconfiguring a shared secret, as we found in our research. Here, we elaborate on our findings through two examples of popular Bonjour-capable systems: automatic printer discovery and AirDrop.

Misleading printer discovery. Today, all major printer vendors support Bonjour-based automatic printer discovery. More specifically, whenever Mac users search their local network for printers, their computer runs Bonjour to find printer service instances from which the users can choose. A selected printer has its service instance name (for example, "HP Printer [928FE5]") saved on the Mac, which enables users to access the printer without going through the service discovery step again. On the other


com.apple.idms.appleid.prd.
4b35667443534[redacted]4c7261376c34513d3
Issued by: Apple Application Integration Certification Authority
Expires: Thursday, July 27, 2017 at 3:16:09 AM China Standard Time
This certificate is valid

Trust
Details

Subject Name
Common Name com.apple.idms.appleid.prd.
4b35667443534[redacted]4c7261376c34513d3d

Figure 2. Apple certificate issued to an Apple account. The certificate is bound to a random string prefixed with com.apple.idms.appleid.prd.

hand, each time a user prints through the service instance name, the target printer's hostname and IP address must be resolved, using the printer's service instance name. We confirmed that this process can be manipulated to steal the document the user intends to print.

The attack happens when a malicious host, such as a compromised Mac in the network, broadcasts to publish and register a service with an existing printer's instance name, in this case, "HP Printer [928FE5]." Nevertheless, each device observes the response and automatically caches it (the mapping between a service type to a service instance), and when a conflict is discovered (the printer finds that the response carries its own instance name), the printer automatically resolves the conflict by changing its own instance name—"HP Printer [928FE5] (2)." The problem is that the Mac keeping the printer's instance name doesn't know about that. When the Mac uses the printer, the printer won't respond to the request sent out to resolve the printer's hostname and IP because the instance name on the request no longer belongs to it. The malicious host, however, will reply with its IP. As a result, the user's document will be sent to the malicious host, which can forward the document to the original printer, silently serving as a man in the middle.

We implemented the attack on a real-world organizational network. Our approach successfully intercepted documents to be printed out on the target printer. Note that this problem isn't limited to printer discovery: most apps and systems using Bonjour don't have protection at all and therefore are equally vulnerable to such an MITM attack. An example is the popular PhotoSync app, whose communication between a Mac and an iPhone for synchronizing photos is exploited by our MITM attack, stealing the photos exchanged across the devices.

Hacking AirDrop. A unique feature of Bonjour is that all identifiers of a device using the mechanism, including its service instance name, hostname, and IP address, are generated dynamically and can be changed at any time. This feature enables automatic configuration of an ad hoc network through which devices easily discover each other and establishes communication channels among them. However, it also makes device authentication difficult. A prominent example here is Apple AirDrop, an ad hoc service that supports short-range exchange of documents between OS X and iOS devices. The service is built on top of Bonjour, enhancing the ZeroConf mechanism with TLS-based security protection.

We revealed the AirDrop process through reverse engineering and inspecting the system component for AirDrop. It turns out that, after the Bonjour discovery and resolution steps, the AirDrop sender running on iOS or OS X discovers the service instance name, IP, and port of another device supporting AirDrop (the server). Then, the sender establishes a TLS connection with the server to collect its device name (a name for the user to recognize the part, such as Jeff's iPhone), Apple account information, and so forth. This TLS connection employs Apple's public-key infrastructure (PKI) to encrypt the data transferred between the devices. Each device uses a user's Apple-signed certificate for authentication and encryption during the TLS connection. The name and information transferred during the connection are used to build a list of discovered devices from which the user chooses one to drop documents. After the user chooses the receiver, the documents are transferred through the TLS connection, and the server confirms the transfer's success.

With TLS protection, it's less clear how the sender verifies the server's TLS certificate, which belongs to the device owner's Apple account (Apple ID). Because

Table 1. Summary of vulnerable apps.

ZeroConf channel	No. vulnerable/sampled	Sensitive information leaked	App examples
Bluetooth low energy	10/13	Username and password for Mac OS X	Near Lock
Multipeer connectivity	24/24	Files and photos transferred, and instant message	Bluetooth U, Photo Transfer, and AirDates
Bonjour	18/22	Files, directories and clipboard synced, documents printed, and instant message	Copybin and Printer Pro Lite
Homegrown	2/2	Remote keyboard input and files transferred	Remote Mouse and SHAREit

none of the server device's identifiers (service instance name, device name, IP, and so on) are meant for long-term use, they can be changed on the fly and therefore can't be bound to the user's TLS certificate. Unlike a website, whose certificate uses the site's hostname (for instance, apple.com) that needs to be checked during a TLS connection, the Apple account of one individual doesn't have identity information that other people can easily verify. Actually, as we found out, what's bound to a user's Apple certificate (used for the TLS connection) is a random string prefixed with com.apple.idms.appleid .prd (see Figure 2), which is supposed to be related to the user's Apple ID. This random string is hard for other users to manually check whether it indeed belongs to the intended user.

Fundamentally, linking a human to a certificate is complicated, due to the challenge in finding any identifiable information both well-known and unique: a name can be duplicate, and date of birth and Social Security number have privacy implications—people might not want to share them with a party with whom they just want to share a file. We found that Apple binds users' Apple ID, denoted by an email address, to the aforementioned random string in their certificates. However, Apple's design isn't secure in practice; oftentimes, Apple users don't save known people's Apple IDs into contacts. Indeed, in our measurement study, we checked all 1,230 contacts saved on nine individuals' iPhones. It turns out that only 119 contacts (9.7 percent) out of 1,230 were saved with their Apple IDs.[2]

Although it's highly likely for this identity check to fail in practice, Apple still shows to users the list of device names, even when the certificates involved can't be bound to any known contacts through the Apple IDs (email addresses). Once users choose a device (through the device's name, like Jeff's iPhone), their documents and photos will be transferred through the AirDrop mechanism, even when the validity of the server's certificate can't be fully verified.

Exploiting this weakness, we successfully attacked AirDrop. Specifically, the attack happens when the attack device sends a response to the AirDrop client during the first step of the Bonjour resolution phase, to bind the service instance name of the real AirDrop server to its own hostname (see Figure 3a). Note that this resolution response can be unicast to the victim—that is, the AirDrop client (the party that initiates the AirDrop communication)—to avoid detection by the server. After that, the TLS connection initiated by the client will go to the attack device.

Alternatively, an attacker can cheat the AirDrop client in the second step of resolution, binding the server's hostname to the attacker's IP address (see Figure 3b). Again, this network packet can be delivered through a unicast channel, without exposure to the server. This, again, will cause the TLS request from the client to go to the attack device. In both cases, the attack device impersonates the server to the client, then connects to the server to act as a man in the middle. Because users have no way to find out whether they're talking to the right person during the process, they might choose the wrong device on the list and send their documents and photos to the attacker. A demo is online at sites.google .com/site/applezeroconf.

Measurement

To find out the scope and magnitude of the security weaknesses in ZeroConf systems, we performed a measurement study, analyzing 61 popular Mac and iOS apps designed to operate without configurations. Our findings demonstrate the issue's significance: a vast majority (88.5 percent) of the apps we analyzed were unprotected, even though the environment they work in can't be fully trusted. Examples of such apps and our findings are summarized in Table 1.

Protecting Apple ZeroConf

Our security analysis shows that there's significant misalignment between the usability-oriented design that characterizes existing Apple ZeroConf systems and the security threats they face in practice.

To better protect the ZeroConf system without undermining its usability, we developed a suite of novel techniques. We first examine an optimistic approach in which the device considers its operating environment safe if the necessary condition of an impersonation or

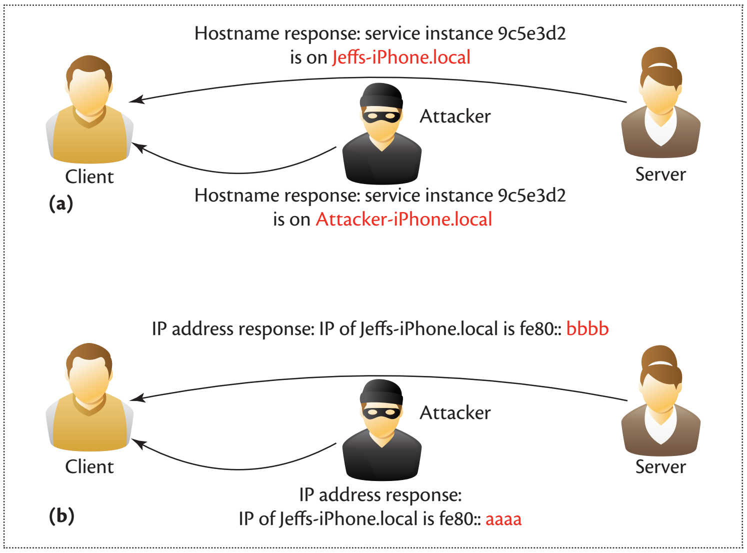

Hostname response: service instance 9c5e3d2 is on Jeffs-iPhone.local
Attacker
Client
Server
(a)
Hostname response: service instance 9c5e3d2 is on Attacker-iPhone.local
IP address response: IP of Jeffs-iPhone.local is fe80:: bbbb
Attacker
Client
Server
IP address response: IP of Jeffs-iPhone.local is fe80:: aaaa
(b)

Figure 3. Attacks on AirDrop. (a) This attack happens when the attack device sends a response to the AirDrop client during the first step of the Bonjour resolution phase, binding the service instance name of the real AirDrop server to its own hostname. (b) Alternatively, an attacker can cheat the AirDrop client in the second step of resolution, binding the server's hostname to the attacker's IP address. In both cases, the attack device impersonates the server to the client, then connects to the server to act as a man in the middle.

MITM attack isn't satisfied. A more generic solution is to leverage Apple's PKI to authenticate the parties involved in ZeroConf operations. Our research shows where the existing PKI fails and how to bridge the gap and make it work on today's ZeroConf systems.

Conflict Detection

A key observation from our security analysis of ZeroConf systems is that all attempts to impersonate an existing device involve hijacking that device's service instance name or hostname. This will cause a conflict observable to the party searching for the victim (the device being impersonated) when the victim also responds to the party's request and the adversary can't disrupt the communication between the party and the victim. Because identity hijacking is a necessary condition for the impersonation and MITM attacks, and the conflict is inevitable assuming no disruption and the victim device is always on, we can conclude that a ZeroConf network is attack free if no conflict is observed. This observation leads to a conflict detection design and its implementations on OS X and iOS, which defeats these attacks and also fully preserves the existing ZeroConf systems' usability.[2]

Speaking Out Your Certificate

The conflict detection techniques are completely automatic, fully preserving the zero configuration property of existing systems. However, in the presence of a conflict, it can't help users identify the trusted party to connect to. In addition, a more fundamental solution against impersonation and MITM attacks should rely on authentication of the parties involved in the communication. In the absence of a shared secret (which needs to be configured across multiple devices), apparently the best solution is to leverage Apple's PKI, using each party's Apple certificate to establish a secure channel between authenticated peers. However, this treatment turns out to be more complicated than it appears: in all the data-sharing cases mentioned here, only Handoff can be potentially secured by authenticating two apps (across iPhone and Mac) with their app signatures; all other cases involve users' Apple account certificate. The challenge here is properly verifying one's ownership of a certificate, which hasn't been addressed by existing techniques.

To link a certificate to a user, we need to attach to the certificate some identifiable but nonsensitive user information, which must also be well-known to his or her contacts. Our idea, named SPYC vouch, is to use voice biometrics to tie users' certificate to their identity, assuming that the parties verifying the certificate know their voice.[2] Specifically, we developed a technique that lets users "speak out" their certificates and use the voice recording to vouch for the relation between the certificate and their identity. To verify the certificate, users must check whether the voice indeed belongs to the person they know as well as whether the certificate content has been correctly spoken. The logic here is that, to impersonate someone else, an attacker needs a victim's cooperation to speak out the attacker's certificate. As a result, the attempt to deceive other parties into using an attacker's certificate as the victim's will fail if those parties know the victim's voice.

Lessons Learned

We evaluated our protection techniques in two human subject studies involving 60 participants and showed that speaker identification using SPYC vouch is reliable, convenient, and resilient to vouch-forging attacks.[2]

Our analysis highlights the fundamental security challenges underlying ZeroConf techniques: in the absence of any preconfigured secrets across different devices, it's difficult to provide proper authentication. Although the problem is caused by the systems' usability-oriented design, the lack of effective protection today comes largely from the inadequate assessment of the security risks these systems face. Although their designs are meant for operating in a friendly setting, ZeroConf systems like AirDrop are actually used in public environments, such as airports, where the security guarantee becomes hard to assure. An important lesson learned from such a misalignment is that the design of a usability-oriented system must be predicated on careful evaluation of the security threats the system faces as well as a clear indication of when it can be safely used and when it can't. We strongly believe that guidelines should be in place to help developers build such systems with proper protection in line with the security risks.

We also highlight the urgent need to develop effective authentication technologies for ZeroConf systems. Our SPYC design already demonstrates preliminary success. In general, these technologies should be intuitive, avoiding complicated configurations a ZeroConf system isn't supposed to have. A follow-up effort is expected to find the right balance between security and usability.

More research is needed to improve our current design and implementation of the protection. For example, a non-biometric solution to the certificate verification problem should be studied. One possibility is to let Apple users choose their own publishable identifiers, such as Facebook profiles, personal websites, and so on, and include it as part of their account information. During email communication with their contacts, such information can be automatically exchanged across different Apple devices. Further effort is needed to find out how to make the approach work. ■

Acknowledgments

The two lead authors, Xiaolong Bai and Luyi Xing, are listed alphabetically.

References

1. T. Smith, "Sony Preps ZeroConf-Style Bluetooth Tech," *The Register*, 18 June 2003; www.theregister.co.uk/2003/06/18/sony_preps_zeroconfstyle_bluetooth_tech.
2. X. Bai et al., "Staying Secure and Unprepared: Understanding and Mitigating the Security Risks of Apple ZeroConf," *Proc. IEEE Symp. Security and Privacy* (SP 16), 2016; doi:10.1109/SP.2016.45.

Xiaolong Bai is a PhD candidate in the Department of Computer Science and Technology at Tsinghua University. His research interests include finding and mitigating new vulnerabilities in mobile systems, including Android and iOS/OS X. Contact him at bxl12@mails.tsinghua.edu.cn.

Luyi Xing is a researcher in the System Security Lab at Indiana University Bloomington. His research interests include finding previously unknown logic and architecture problems in modern systems, including iOS, OS X, and Android, and high-profile applications on them. Xing received a PhD in security informatics from Indiana University Bloomington. Contact him at luyixing@indiana.edu.

Nan Zhang is a PhD candidate in the System Security Lab at Indiana University Bloomington. His research interests include system security and mobile security, including finding vulnerabilities and designing defense techniques on Android, iOS, and Internet of Things systems. Contact him at nz3@indiana.edu.

XiaoFeng Wang is a professor in the School of Informatics and Computing at Indiana University Bloomington. His work focuses on cloud and mobile security as well as data privacy, particularly the privacy challenges in large-scale analysis and dissemination of human genomic data. Wang received a PhD in electrical and computer engineering from Carnegie Mellon University. He's a recipient of the 2011 Award for Outstanding Research in Privacy Enhancing Technologies (the PET Award) and the Best Practical Paper Award at the 32nd IEEE Symposium on Security and Privacy. Contact him at xw7@indiana.edu.

Xiaojing Liao is a PhD candidate in the School of Electrical and Computer Engineering at Georgia Tech and a member of the Communications Assurance and Performance (CAP) group. Her research interests include network security, online crime modeling, and cyber-physical system privacy. She's a student member of IEEE. Contact her at xliao@gatech.edu.

Tongxin Li is a PhD candidate in the Department of Computer Science at Peking University. His research interest is mobile security, including program analysis and vulnerability discovery on Android and iOS. Contact him at litongxin@pku.edu.cn.

Shi-Min Hu is a professor in the Department of Computer Science and Technology at Tsinghua University. His research interests include system software and security, computer graphics, and computer vision. Hu received a PhD in mathematics from Zhejiang University. He's editor in chief of *Computational Visual Media* and on the editorial board of several journals, including *IEEE Transactions on Visualization and Computer Graphics*, *Computer Aided Design*, and *Computer & Graphics*. He's a Senior Member of IEEE and ACM. Contact him at shimin@tsinghua.edu.cn.