How a macOS bug could have allowed for a serious phishing attack against users

Mar 14 2022 3:00 PM

Phishing attacks are a very common threat in our digital lives. So much so that many companies try to trick their own employees into falling for fake phishing attacks in order to assess their skills when trying to identify a certain message as genuine or not.

If you don't know what a phishing attack is, here's what Wikipedia has to say about it:

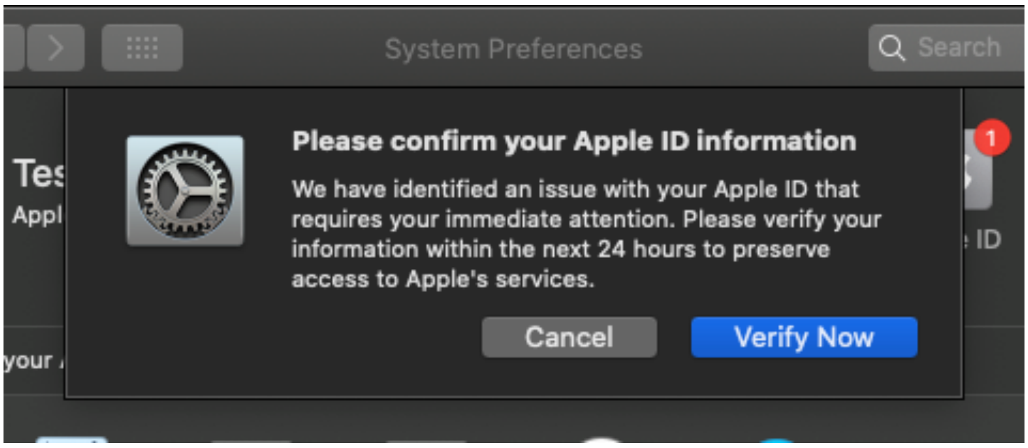
> *Phishing is a type of social engineering where an attacker sends a fraudulent (e.g., spoofed, fake, or otherwise deceptive) message designed to trick a person into revealing sensitive information to the attacker or to deploy malicious software on the victim's infrastructure like ransomware.*

The issue on macOS

One example of a phishing attack would be someone sending you an email pretending to be Apple, saying that there's some sort of problem with your Apple ID and that you need to act quickly to sort it out. Of course most savvy users would notice such an attempt and simply ignore the email or log in to their account manually to check if everything was ok.

There are many signs of a phishing attack one could check for in that example, such as whether the sender of the email is really Apple and whether the domain of the link directs to an Apple official website.

However, what would you do if you suddenly received a notification on macOS telling you to "Verify your Apple ID information", then upon launching System Preferences, saw an alert like the one below?


Please confirm your Apple ID information

We have identified an issue with your Apple ID that requires your immediate attention. Please verify your information within the next 24 hours to preserve access to Apple's services.

Apologies for not including Retina-quality assets in this post.

Check out this video to see the whole thing in action:

I'd say most users would accept the default "Verify Now" action, which then launches a form where you fill in your Apple ID email and password.

The only problem is that such a scary-looking alert, right within System Preferences, could be sent by ANY app running on your Mac. The malicious scenario would involve an app that looks like a simple, regular app (it could even be a sandboxed app) sending such a notification, which would then open up an Apple ID login panel that when submitted sent your email and password to the bad actor. The notification could even be made to look like it came from the System Preferences app itself, making it much more believable.

“But what about two-factor authentication?”, you might be asking. A sophisticated attacker could devise a system to try to authenticate your credentials on a machine under their control, cause a two-factor code to appear on your device, and convince you to type this code into the panel. This would send that code to the attacker, who would then gain access to pretty much all the data in your Apple account.

Additionally, it is possible to extract some information from an Apple ID just by authenticating with the email and password, without two-factor authentication.

The technical details

The phishing attack described here relies on a mechanism on macOS (which is also present on iOS) called CoreFollowUp. You know those annoying “Verify your Apple ID” or “Finish Setting Up Your Device” messages you get all the time? Those are being posted via CoreFollowUp.

CoreFollowUp is used by several components of both macOS and iOS, and they communicate with it through a daemon called `followupd`. The problem was that the daemon failed to validate connections made to it on macOS, which meant that any process that could look up its Mach service (including sandboxed apps) would be able to send it commands, including ones that would trigger that scary dialog within System Preferences.

A partial fix, released in macOS 11.3, included an allow list for the URL that’s launched when the user clicks the “Verify Now” button in the example shown above. By ensuring that only Apple-approved URLs could be used, this prevents the attack from being very useful except for sending unsolicited notifications without user permission.

A more complete fix was released in macOS 12.3, preventing any random binary on the system from talking to the daemon and registering notifications, regardless of the destination URL. Apple addressed it by introducing a new entitlement: `com.apple.private.followup`. As of macOS 12.3, when a process attempts to connect to the CoreFollowUp daemon, it will validate the connection by checking if the connecting process has that entitlement in its code signature, denying communication if it doesn't.

This was an interesting attack vector and it shows how important it for Apple to protect powerful daemons on the system against any random process that could be trying to do bad things.

It's not just an Apple problem though, many third-party apps on macOS ship with their own privileged daemons that accept connections over XPC. It's important that such services use some form of authentication of the calling process, in order to prevent potentially malicious use and exfiltration of user data. If you are a developer and would like to learn more about the subject, there's an excellent series of posts on theevilbit.

Timeline:

- December 23, 2020: I discovered the issue and reported it to Apple's security team
- January 11, 2021: I got a reply confirming that they were investigating it
- February 15, 2021: Another reply stating that it would be fixed in a future update
- April 26, 2021: macOS 11.3 released with a partial fix
- August 31, 2021: I received a US$5000 bug bounty payment from Apple
- March 14, 2022: macOS 12.3 released with a complete fix (CVE-2022-22660)

As you can tell, it took Apple quite a long time to fully address this issue, which is far from ideal.

P.S. About Apple's Bug Bounty Program

Note: the information below is how it worked for my specific situation, I'm not sure but it might vary according to where you're located (I'm in Brazil).

This was my first time ever participating in Apple's bug bounty program. One thing to note about the Apple program specifically has to do with how you get paid: through an Apple Developer account.

In order to receive the payment, you must have an Apple Developer account with the paid applications agreement properly signed and banking set up. If you don't have a developer account yet, you have to sign up for one and Apple will refund the $99 fee with your bounty payment.

This account requirement does not apply to submitting the initial bug report and communicating with Apple's security team, it only becomes a requirement once it gets to the point of getting paid for it. Anyone can submit security vulnerability reports by emailing `product-security@apple.com`, a process that is detailed here.

The email that you use to submit the original bug report must be associated with that developer account, otherwise they can't pay you. In my case, I sent the report with an email that is part of my Apple ID, which is in turn part of my developer account, but the email that I used to submit was not my *primary* Apple ID email, so it took a bit of back and forth until they understood that the email was already a part of my developer account due to it being a part of my Apple ID.

So if you're already a member of the Apple Developer program, I strongly recommend submitting your security vulnerability reports through the