Security Now! #813 - 04-06-21

A Spy in Our Pocket

This week on Security Now!

This week, by popular demand, we examine the big coverup at Ubiquity. We look at the consequences of the personal data of 533+ million Facebook users appearing on the Net and how to tell if you're represented there. We look at another water treatment plant break-in with a very different outcome. We look at new moved by Google to further lock-down Android against abuses of its permissive-by-design API services. We look at the new threat to Call Of Duty cheaters and yet another set of serious vulnerabilities in QNAP NAS devices. Then after sharing a catchy Tweet, we look into some new research from researchers in Ireland into the unwarranted chattiness of iOS and Android mobiles phones.

A Backend Fix for Frontend Issues

Security News

The Ubiquiti Coverup

Our longtime listeners know that I've been a fan of the Ubiquiti EdgeRouter X which is an amazingly affordable 5-port Ethernet router. The thing that drew me to it was that its five interfaces were actually individual NICs which could each have their own subnet to create a truly segmented network. As I've often preached, this is the ONLY WAY to create a secure environment on today's LAN. With many of our IoT devices phoning home to Chinese cloud services, there can be no true security if those devices are allowed to sit on the same network as the family's or small business's LAN. IoT devices =MUST= be given their own isolated network. And at the time the Ubiquity was the only cost effective solution for doing that. Today, many consumer WiFi routers have one or two guest channels which support explicit inter-network isolation. When I'm unable to reach my Sonos speakers from my iPad it's because my iPad is on the wrong WiFi.

Since we so often talked about Ubiquity, it should be no surprise that many of our listeners have made sure that I knew of the mess that has recently erupted. It was fresh last week as this podcast was taking shape, and it seemed as though Brian Krebs, who was in the middle of it all, was still getting to the bottom of what was going on. We know much more today:

It now appears clear that Ubiquiti has been covering up the extreme severity of a data breach that put their customers' networks at significant risk of unauthorized access. Brian Krebs has cited an internal and unnamed whistleblower from within Ubiquity.

In January, Ubiquity seriously downplayed what it said was <quote> "unauthorized access to certain of our information technology systems hosted by a third-party cloud provider." While that was literally correct, it was also quite vague. The notice said that, while there was no evidence the intruders accessed user data, the company couldn't rule out the possibility that they obtained users' names, email addresses, cryptographically hashed passwords, addresses, and phone numbers. Ubiquiti recommended users change their passwords and enable two-factor authentication. So... at this point, standard generic post-breach boilerplate.

But last Tuesday's report from Brian Krebs was able to flesh this out by citing a security professional within Ubiquiti who helped the company respond to the two-month-long breach, which began in December 2020. The Brian's inside source reported that the breach was much worse than Ubiquiti had disclosed publicly and that executives were minimizing the severity to protect the company's stock price. <sigh> No surprise there.

And to make matters worse, the breach follows Ubiquiti's push —if not outright requirement— for cloud-based accounts for users to set up and administer devices running their newer firmware versions. An article on Ubiquity's site says that during the initial setup of a UniFi Dream Machine (a popular router and home gateway appliance), users will be prompted to log in to their cloud-based account or, if they don't already have one, to create an account.

The page states: *"You'll use this username and password to log in locally to the UniFi Network Controller hosted on the UDM, the UDM's Management Settings UI, or via the UniFi Network Portal (https://network.unifi.ui.com) for Remote Access."*

And these changes have not gone unremarked upon. Ubiquiti's customers are decidedly unhappy, complaining about the cloud login requirement and the risk it poses to the security of their devices.

As I noted previously, according to Brian's inside contact — whom he refers to as "Adam" for convenience — the data that was accessed was much more extensive and sensitive than Ubiquiti portrayed. Paraphrasing a bit what Brian wrote:

"In reality, Adam said, the attackers had gained admin access to Ubiquiti's servers within Amazon's cloud service, which secures the underlying server hardware and software but still requires the cloud's tenant to secure access to any data stored there.

"They were able to get cryptographic secrets for single sign-on cookies and remote access, full source code control contents, and signing keys exfiltration."

In other words, yeah, full administrative access rights. "Adam" says the attacker(s) had access to privileged credentials that were previously stored in the LastPass account of a Ubiquiti IT employee, and gained root administrator access to all Ubiquiti AWS accounts, including all S3 data buckets, all application logs, all databases, all user database credentials, and secrets required to forge single sign-on (SSO) cookies.

This access would allow the intruders to remotely authenticate to countless Ubiquiti cloud-based devices around the world — owned by Ubiquity's customers. According to its website, Ubiquiti has shipped more than 85 million devices that play a key role in networking infrastructure in over 200 countries and territories worldwide. All of those devices which had been updated to use Ubiquity's now-mandatory cloud facility were vulnerable to remote take over.

And even after all this, Ubiquity continues spewing the corporate line:

"As we informed on January 11, we were the victim of a cybersecurity incident that involved unauthorized access to our IT systems. Given the reporting by Brian Krebs, there is newfound interest and attention in this matter, and we would like to provide our community with more information.

At the outset, please note that nothing has changed with respect to our analysis of customer data and the security of our products since our notification on January 11. In response to this incident, we leveraged external incident response experts to conduct a thorough investigation to ensure the attacker was locked out of our systems.

[Yeah, after rummaging around in there, unfettered for two months.]

These experts identified no evidence that customer information was accessed, or even targeted. The attacker, who unsuccessfully attempted to extort the company by threatening to release stolen source code and specific IT credentials, never claimed to have accessed any customer information. This, along with other evidence, is why we believe that customer data was not the target of, or otherwise accessed in connection with, the incident.

At this point, we have well-developed evidence that the perpetrator is an individual with intricate knowledge of our cloud infrastructure. As we are cooperating with law enforcement in an ongoing investigation, we cannot comment further.

All this said, as a precaution, we still encourage you to change your password if you have not already done so, including on any website where you use the same user ID or password. We also encourage you to enable two-factor authentication on your Ubiquiti accounts if you have not already done so."

So, our takeaway is that Ubiquity's skeptical customers were 100% correct to object to upgrades of their system's firmware which offered them no alternative other than to manage their own local devices though an online cloud service. That's insane. And this meant that ALL of these devices were now vulnerable to remote access compromise. This was clearly wrong-headed from the start.

Anyone using these cloud-attached Ubiquiti devices should change their passwords and enable two-factor-authentication if they haven't already done so. And given that intruders into Ubiquiti's network had access to single sign-on cookies and secrets enabling remote access, including signing keys, it would also be a good idea — annoying though it is — to delete any profiles associated with a device, make sure the device is using the latest firmware, and then recreate profiles under brand new credentials. And, as always, remote access should be disabled unless it's truly needed.

Facebook's 533,313,128 Million User Whoopsie!

We talked a lot about this Facebook breach back when it happened two years ago in 2019. And now, the data taken during that intrusion has all just been made available online for free. This includes the full names, Facebook IDs, mobile numbers, locations, email addresses, gender, occupation, city, country, marital status, account creation date, and other profile details broken down by country, with over 32 million records belonging to users in the U.S., 11 million users the U.K., 6 million users in India, and so forth. And ya gotta love it that Facebook has a position titled: *"Director of strategic response communications."*

"Hi, what do you do?" ... "My name is Liz Bourgeois [it actually is] and I'm the director of strategic response communications for Facebook." "Wow. So, when the you-know-what hits the fan, you're the person who gets quoted in the media, while managing to never say anything meaningful?" "Yes. That's a perfect job description!"

In this case, Liz said: *"This is old data that was previously reported on in 2019. We found and fixed this issue in August 2019."*

Oh, well, if the data is old, then ...what... That creaky old data must no longer apply, right? People have all changed their names and phone numbers and Facebook IDs, they've all moved and changed their genders and so forth. So we really don't need to worry that data for 533,313,128 Facebook users from only two years ago — not SO old, really — is now being offered for free, in bulk, on the dark web? Thanks, Liz! Whew! For a while there, this seemed like it might be a big deal. That was a beautifully executed strategic response!

Just to remind our listeners, the data is known to have been obtained by exploiting a vulnerability in Facebook's 'Add Friend' feature which enabled automated scripts to scrape Facebook users' public profiles and associated private phone numbers in bulk. It's true that the leak was fixed. And at the time I was somewhat sympathetic to the complexity of doing what Facebook is doing. But also worried that their controls allowed it to happen in the first place. We do need better from them — but I suspect that Liz is likely a lost cause.

If you're wondering whether you might be affected by this massive Facebook data dump, Troy Hunt's "Have I Been Pwned" site has your answer…

Have I been Pwned by Facebook?
Troy Hunt has a long tweet thread on Twitter about his recent addition of the leaked Facebook data appearing on the web: https://twitter.com/troyhunt/status/1378463581604220931

The upshot is that the data appears to be not only incomplete but sparse when it comes to Facebook user eMail addresses, which is that Troy's "Have I been Pwned" site is currently setup to cross reference. Of the 533+ million Facebook member records, only 2.5 million include an eMail address.

What we really want is a phone number, which is far more highly represented in the data. Troy indicated that he's exploring the possibility of allowing users to search on their phone numbers.

In the meantime, you can go over to Troy's excellent "Have I Been Pwned" site to update your awareness of any leakage of your eMail, from Facebook and elsewhere. It will let you know which breaches contain your information: https://haveibeenpwned.com/

Don't mess with our water!
We never really got any closure on that hack to the water treatment plan in Oldsmar, Florida which occurred in early February. But it may have put other similar facilities on alert.

Saturday before last, on March 27th, a young 22-year-old named Wyatt Travnichek, living in Ellsworth County, Kansas is believed to have broken into a protected computer system belonging to the county's Post-Rock Rural Water District. For some reason he used his illegal access to shut down the cleaning and disinfecting processes at the facility. The public reports don't specify whether any contamination may have resulted to the water supply, but authorities are not taking this lightly at all.

Wyatt was quickly identified and indicted on the serious charges that he had accessed a public water facility's computer system, jeopardizing the safety and health of the residents' of the local community. So Wyatt has been charged with one count of tampering with a public water system and one count of reckless damage to a protected computer during unauthorized access, according to the Department of Justice (DoJ).

Lance Ehrig, the Special Agent in Charge of the Environmental Protection Agency (EPA) Criminal Investigation Division in Kansas, said: "By illegally tampering with a public drinking water system, the defendant threatened the safety and health of an entire community. EPA and its law

enforcement partners are committed to upholding the laws designed to protect our drinking water systems from harm or threat of harm. Today's indictment sends a clear message that individuals who intentionally violate these laws will be vigorously prosecuted."

Lance's comments suggest that there might be some embarrassment and reaction over no one having been identified and held accountable for the previous very high profile event in Oldsmar, Florida. Wyatt's indictment does not specify whether his attack was successful, nor how it was detected. But if Wyatt should be found guilty, he faces up to 25 years in federal prison and a total fine of $500,000.

The other salient fact is that it turns out Wyatt was not some naïve opportunistic post-teenage hacker. The indictment tells that he was previously employed by the water district and in his employment capacity he remotely logged into the water district's computer system on a regular basis. So, he knew what he was doing and how to do it. He quite deliberately shutdown the water cleaning and disinfecting processes at the facility. Which, at least in my mind, extra creepy and far less prone for sympathy.

At the same time, this also suggests that the water officials at that facility also failed to properly secure credentials by not proactively removing Wyatt's remote access account after he left. They may not have been able to do so conveniently if, for example, everyone was sharing the same credentials which is, of course, a big no-no. But in any event, let's hope that word of this spreads and that, at least our own domestic hackers learn the lesson that messing around with public utilities is not something the US criminal justice system is going to ever take lightly.

Android moves to limit inter-app visibility.

On the Android platform, apps have always been able to detect the presence of specific apps, even collecting a full list of installed apps through the "QUERY_ALL_PACKAGES" api. And what's more, apps can be set to receive OS notifications when a new app is installed.

That feature really has the smell of "Hey, wouldn't it be neat if we let apps be informed when other apps are installed? Just think of all the cool things you could do with that!" Unfortunately, as we frequently see, "cool things you could do" is often quickly turned to the dark side. It doesn't take a rocket scientist — or even a computer scientist — to observe that this wide open facility would provide yet another means for fingerprinted devices and profiling their users.

And, it's not just theoretical. Seven years ago, back in 2014, Twitter began tracking the list of apps installed on users' devices as part of its "app graph" initiative with an aim to deliver tailored content. And the digital wallet company MobiKwik was also caught collecting information about installed apps in the wake of a data breach that just recently came to light earlier this week. And a study published by a team of Swiss researchers two years ago, in 2019, concluded that "free apps are more likely to query for such information and that third-party libraries are the main requesters of the list of installed apps." The Swiss researchers said that "As users have on average 80 apps installed on their phones, most of them being free, there is a high chance of untrusted third-parties obtaining the list of installed apps."

And a year ago, in March of 2020, another academic study found that 4,214 Google Play apps stealthily collected a list of all other installed apps to allow developers and advertisers to build

detailed profiles of users. "Hey! Google lets us do it, so it must be okay!" Android makes this convenient with two OS function calls: getInstalledPackages() and getInstalledApplications(). "Look! They're right there! I'll impress my boss by using them!"

Well... the reason we're highlighting this long-running behavior today, is because Google is about to clamp down on this cool but overly permissive and abuse-begging facility:

https://support.google.com/googleplay/android-developer/answer/10446026
From Google's "Developer Program Policy: March 31, 2021 announcement"

We're updating the following policies. All new and existing apps will receive a grace period of at least 30 days from March 31, 2021 (unless otherwise stated) to comply with the following changes:

Package Visibility - effective Summer 2021

The inventory of installed apps queried from a device are regarded as personal and sensitive user data subject to the Personal and Sensitive Information policy, and the following requirements:

Apps that have a core purpose to launch, search, or interoperate with other apps on the device, may obtain scope-appropriate visibility to other installed apps on the device as outlined below:

- **Broad app visibility:** Broad visibility is the capability of an app to have extensive (or "broad") visibility of the installed apps ("packages") on a device.
    - For apps targeting API level 30 or later, broad visibility to installed apps via the QUERY_ALL_PACKAGES permission is restricted to specific use cases where awareness of and/or interoperability with any and all apps on the device are required for the app to function.
        - You may not use QUERY_ALL_PACKAGES if your app can operate with a more targeted scoped package visibility declaration(e.g. querying and interacting with specific packages instead of requesting broad visibility).
    - Use of alternative methods to approximate the broad visibility level associated with QUERY_ALL_PACKAGES permission are also restricted to user-facing core app functionality and interoperability with any apps discovered via this method.
    - Please see this Help Center article for allowable use cases for the QUERY_ALL_PACKAGES permission.

- **Limited app visibility**: Limited visibility is when an app minimizes access to data by querying for specific apps using more targeted (instead of "broad") methods (e.g. querying for specific apps that satisfy your app's manifest declaration). You may use this method to query for apps in cases where your app has policy compliant interoperability, or management of these apps.

- Visibility to the inventory of installed apps on a device must be directly related to the core purpose or core functionality that users access within your app.

App inventory data queried from Play-distributed apps may never be sold nor shared for analytics or ads monetization purposes.

It is truly unfortunate that we cannot have nice things. Or at least that a massively widely used and deliberately permissive system, with all the original features and benefits of Android, is inevitably being slowly whittled down, locked down and tightened up. I recall hearing Leo so often proclaim that he was using Android explicitly for its deliberately non-Apple freedom and openness. It's clear that Google and their Android engineering designers had their hearts in the right place. They wanted to create a free and open platform for the world to use. But naïve users need to be protected from all the things that their open pocket computer might do that's against their interest and expectations.

This podcast is titled "A Spy in Our Pocket" due to some new and worrisome research from Dublin, Ireland. But there's a much broader sense in which we're all carrying around Pocket Spys.

Beware malicious "Call of Duty: Warzone" cheats

Everywhere I looked in the tech press this past week I saw mentions of the new malicious game cheats for Activision's "Call of Duty: Warzone". The idea of infecting gamers through malicious cheats has been a longtime persistent and popular means for bad guys sneaking their malicious code into the machines of unsuspecting — though also somewhat less than completely virtuous — game players who are willing to cheat to get ahead.

Still, malware is malware, and the need to obtain an edge, even if not ethically, is inducing gamers to drop their system's built-in protections. The cheats instruct gamers to run the program as an administrator and to disable antivirus. That's interesting, because the gamer knows they are downloading and attempting to run something that's shady. So to them it might make sense that they would need to give the installer admin rights and, of course, disable their A/V system upon which they would otherwise be depending. And, as we know, while these voluntary circumventions are often needed for a cheat to work, they also make it easier for malware to survive reboots and to go undetected, since users won't get warnings of the infection or that software is seeking heightened privileges — which it has been explicitly granted by its user.

Activision noted this in their report, by writing: "While this method is rather simplistic, it is ultimately a social engineering technique that leverages the willingness of its target (players who wish to cheat) to voluntarily lower their security protections and ignore warnings about running potentially malicious software." Activision also provided a long list of Warzone Cheat Engine variants that installed malware of all descriptions.

And, this is being distributed using the increasingly popular affiliate model. Activision's analysis indicated that multiple malware forums are regularly advertising a kit that customizes the fake cheat. The kit makes it easy to create versions of Warzone Cheat Engine that deliver malicious payloads chosen by the soon-to-be-attacker who plans to offer it. The people selling the kit advertised it as an effective way to spread malware and "some nice bait for your first malware project." The sellers have also posted YouTube videos that promote the kit and explain how to use it.

So, turnkey cheat-ware for a super-popular game packaged for repackaging by affiliates. It might not be worth that endless ammo, speed, and invincibility. In any event, while I'm sure

that none of OUR listeners would stoop so low as to use something like this, if anyone knows a hot gamer who might, it could be worth dropping them a word of caution in this case.

QNAP — Just Say No!

For this podcast, I've been trying to avoid talking about QNAP, sort like I try to avoid talking about ransomware. By this point we all know that it exists and it's bad, and "what are ya gonna do?" But every indication is that QNAP is a BAD company which produces BAD — meaning insecure and in many cases insecurable — products. Yet they are the #1 Chinese supplier of Network Attached Storage (NAS) devices and they hold, by some accounts, about a 69% share of the NAS market globally. So, when something really extra bad happens, and continues to happen, we have to talk about it here.

Few things are more important than the security of Network attached storage. It's "Network" and it's "Attached" and it's "Storage." Presumably it's storing things that its users might like to be kept secure and perhaps even confidential. And given that the new attacks are "pivot style" where an attacker gets into a device on a private network LAN boundary which forms a bridge between that private LAN and the public Internet, it's not so much that someone's laundry list might become public as that an intruder can leverage their position on such appliances as just the start of much more devastating and serious intrusions.

So what happened? Security researchers at SAM Seamless Networks published their report last Wednesday containing news that they had been sitting on for months in an attempt to be responsible. But QNAP was not.

https://securingsam.com/new-vulnerabilities-allow-complete-takeover/

SAM's report is titled: "New Vulnerabilities Allow Complete Device Takeover"

I've edited and shortened it a bit for the podcast:

> SAM's security research team is actively looking for vulnerabilities in IoT devices that have yet to be discovered in order to ensure our network security coverage is as accurate and up to date as possible. SAM's engine subsequently blocks such vulnerabilities from the first day of their discovery, often prior to the vendor resolving it.
>
> Below is a summary of two recent vulnerabilities and their potential impacts that our research team discovered in a specific kind of NAS device made by QNAP.
>
> It's standard practice to report the discovered vulnerabilities to the vendor and allow for a 90-120 day grace period for them to resolve it prior to notifying the public. As seen in the timeline below, we followed the responsible disclosure procedure and immediately reported it to the vendor especially as the impacts of their exploitation are significant. These vulnerabilities are severe in nature as they allow for full takeover of devices from the network, including access to the user's stored data, without any prior knowledge.
>
> We discovered two critical vulnerabilities in QNAP TS-231's latest firmware (version 4.3.6.1446 – 2020/09/29).

- Web server: allows a remote attacker with access to the web server (default port 8080) to execute arbitrary shell commands, without prior knowledge of the web credentials.

- DLNA server: allows a remote attacker with access to the DLNA server (default port 8200) to create arbitrary file data on any (non-existing) location, without any prior knowledge or credentials. It can also be elevated to execute arbitrary commands on the remote NAS as well.

[So, yeah, their "Complete Device Takeover" titling is not an exaggeration.]

These may affect other models and firmware versions as well.

[And here it comes:...]

We reported both vulnerabilities to QNAP with a 4-month grace period to fix them. Unfortunately, as of the publishing of this article, the vulnerabilities have not yet been fixed.

Due to the seriousness of the vulnerabilities, we decided not to disclose the full details yet, as we believe this could cause major harm to 10's of thousands of QNAP devices exposed to the Internet.

Then, under their "Technical Details" they briefly lay out the scenario. And note that even now, these guys are being much more responsible than some of the events we've recently discussed:

Technical Details

Vulnerability #1 – RCE vulnerability: affects any QNAP device exposed to the Internet.

This vulnerability resides in the NAS web server (default TCP port 8080).

Previous RCE attacks on QNAP NAS models relied on web pages which do not require prior authentication, and run/trigger code on the server-side. We've therefore inspected some cgi files (which implement such pages) and fuzzed a few of the more relevant ones.

Most of the cgi files that are available through the web server reside at */mnt/HDA_ROOT/home/httpd/cgi-bin* directory on the TS-231 file system.

During the inspection, we fuzzed the web server with customized HTTP requests to different cgi pages, with focus on those that do not require prior authentication. We've been able to generate an interesting scenario, which triggers remote code execution indirectly (i.e., triggers some behavior in other processes).

[And that's all they're saying about it in the hopes that QNAP will awaken from their long QNAP and get this fixed!]

Process for solving vulnerability

The vendor can fix the vulnerability by adding input sanitizations to some core processes and library APIs, but it has not been fixed as of this writing.

Disclosure timeline:

- October 12, 2020 – Full disclosure reported to QNAP security team.
- October 23, 2020 – Sent another e-mail to QNAP security team.
- October 31, 2020 – Automatic reply from "QNAP support" with a ticket number.
- January 26, 2021 – Sent a notification to QNAP about end of the grace period (which is planned to end on February 12).
- January 26, 2021 – Reply from QNAP Helpdesk: the problem is confirmed but still in progress.
- February 12, 2021 – Grace period has elapsed.
- March 31, 2021 – Initial blog post published.

Vulnerability #2 – Arbitrary file write vulnerability:

This vulnerability resides in the DLNA server (default TCP port 8200).

The DLNA server is implemented as the process *myupnpmediasvr*, and handles UPNP requests on port 8200.

[WHAT could possibly go wrong??!!!??]

We discovered the vulnerability during investigation of the process's behavior and communication both externally and internally.

We've been able to elevate that vulnerability to remote code execution on the remote NAS as well.

Disclosure timeline:

- November 29, 2020 – Full disclosure reported to QNAP security team. No reply from QNAP has been received yet for this specific disclosure.
- March 29, 2021 – Grace period has elapsed.
- March 31, 2021 – Initial blog post has been published.

Hopefully, this public disclosure may finally work to get QNAP's long overdue attention. No good would come from SAM's more full disclosure. Given the nature of attacks, I'd argue that there would never be any reason for them to disclose. But if they don't convincingly threaten to do so, it appears that QNAP's irresponsible behavior will continue.

As I started out by saying, I've been avoiding talking about QNAP and have already let many similar stories go uncovered here. And PLEASE everyone take this to heart when you're choosing a NAS supplier. And if you already have QNAP devices, find some use for it inside your network. Remove it from the public Internet.

Listener Feedback

The Realest M.F. @MaxFeinleib
Great remark from my college CS professor: "There are two hard problems in computer science: cache coherence, naming things, and off by one errors."

A Spy in Our Pocket

This week's podcast owes its title to the title of the research paper recently published by Douglas Leith's group at Trinity College in Dublin, Ireland. Douglas is in the School of Computer Science & Statistics there. As I've mentioned before, I have a fond memory of Trinity College from my visit to the Dublin chapter of OWASP in September of 2019 to demonstrate SQRL. Lorrie and I watched the beautiful Trinity campus pass by while we attempted to figure out how to open the doors of the metropolitan transit. We never did, so we missed our opportunity to walk around.

But, in any event, Douglas titled his team's research paper: "Mobile Handset Privacy: Measuring The Data iOS and Android Send to Apple And Google" Now, needless to say, neither Apple nor Google are pleased about having a third party poking into their device's communications. They'd rather that no one did. But we're all familiar with the phrase "keeping them honest" which I think applies here. It's not at all that either of these behemoths are attempting to do anything nefarious. I would argue that it's more likely the case — and recent history with all forms of computing teaches us — that without adequate supervision, and especially when an entity imagines that no one will ever look, the strong tendency is to collect data less because it's truly needed than because it's there and today bandwidth, storage and computation might as well be free. So, slurp it all up just in case, and we'll see whether we have any actual need for any of it later. I'll also note that one thing these researchers do is notice how this information could be used and potentially abused.

Against that backdrop, their paper's Abstract summarizes what their independent analysis found:

> We investigate what data iOS on an iPhone shares with Apple and what data Google Android on a Pixel phone shares with Google. We find that even when minimally configured and the handset is idle, both iOS and Android share data with Apple & Google on average every 4.5 minutes. The phone IMEI, hardware serial number, SIM serial number and IMSI, handset phone number etc. are shared with Apple and Google. Both iOS and Android transmit telemetry, despite the user explicitly opting out of this. When a SIM is inserted, both iOS and Android send details to Apple & Google. iOS sends the MAC addresses of [all] nearby devices — other handsets and the network gateway — to Apple together with their GPS location. Users have no opt out from this and currently there are few, if any, realistic options for preventing this data sharing.

Again, it's not that there's anything inherently wrong with doing this. But someone needs to ask why? Why are they collecting this data? What's being done with it. Certainly, as technologists, we CAN readily envision many things that we would not like to have done with all of this information. So if Apple and Google are not doing any of those worrisome things, what are they doing? And if they're doing nothing with any of that information then why is it being collected?

Certainly some of that is truly essential to the operation of any cellular radio system. We know that the nature of cellular service requires a back and forth dialog with nearby cell towers. But none of that should involve any data above the level of cellular connection maintenance. Or, if it does, documentation and disclosure should be available. Yet none is.

The researchers organized their presentation beautifully in layers of successive detail. So I want to share their top level of detail which is all we need to fill-in the relevant facts. I've lightly edited their presentation for better delivery through this podcast and I've included a link to their full research paper for anyone who's interesting in greater granularity:

https://www.scss.tcd.ie/doug.leith/apple_google.pdf

In this paper we investigate the data that mobile handset operating systems share with the mobile OS developer; in particular what data iOS on an iPhone shares with Apple and what data Google's Android on a Pixel phone shares with Google. While the privacy of mobile handsets has been much studied, most of this work has focussed on measurement of the app tracking/advertising ecosystem and much less attention has been paid to the data sharing by the handset operating system with the mobile OS developer.

Handset operating systems do not operate in a standalone fashion, but rather operate in conjunction with backend infrastructure. For example, handset operating systems check for updates to protect users from exploits and malware, to facilitate running of field trials (e.g. to test new features before full rollout), to provide telemetry, and so on. Hence, while people are using an iPhone, the iOS operating system shares data with Apple; and when using a Pixel, the operating system shares data with Google, and this is part of [each device's] normal operation.

To allow direct comparisons, we define experiments that can be applied uniformly to the handsets studied that generate reproducible behaviour. Both Apple and Google provide services that can be, and almost always are, used in conjunction with their handsets, e.g. search (Siri, OkGoogle), cloud storage (iCloud, Google Drive), maps/location services (Apple Maps, Google Maps), photo storage/analytics (Apple Photo, Google Photos). We endeavor to keep these two aspects separate and to focus on the handset operating system in itself, separate from optional services such as those.

We assume a privacy-conscious but busy/non-technical user, who when asked, does not select options that share data with Apple and Google but otherwise leaves handset settings at their default value. In these tests we evaluate the data shared: (1) on first startup following a factory reset, (2) when a SIM is inserted or removed, (3) when a handset lies idle, (4) when the settings screen is viewed, (5) when location is enabled/disabled, and (6) when the user logs in to the pre-installed app store.

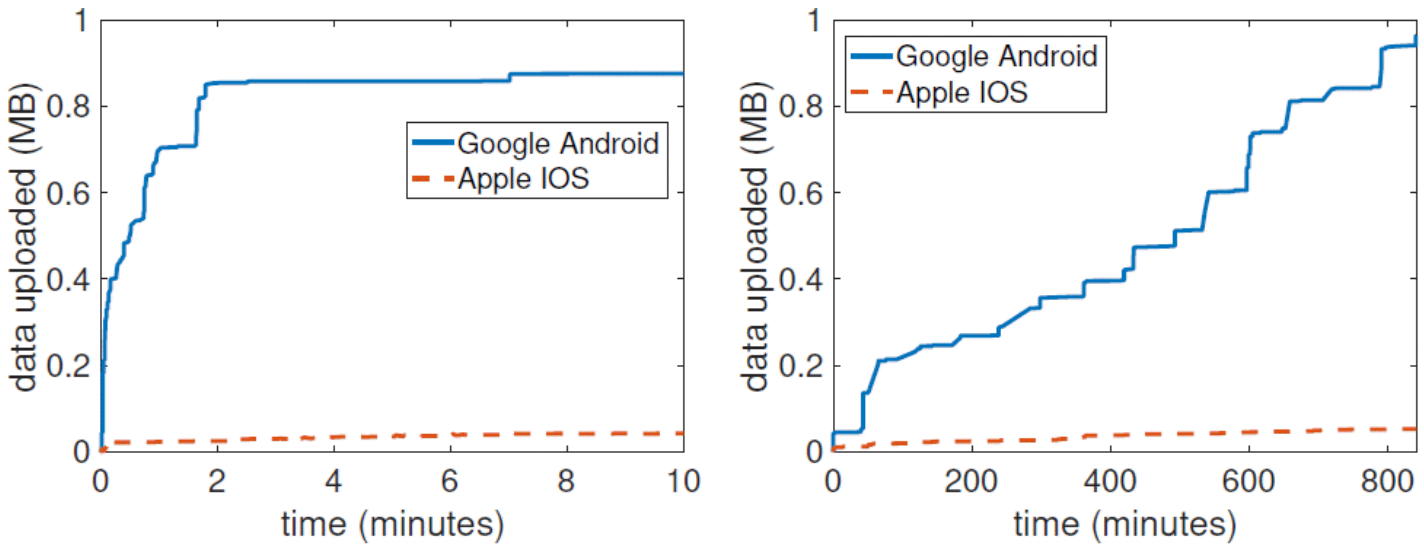
We note that these tests can be partly automated and used for handset operating system privacy benchmarking that tracks changes in behaviour over time as new software versions are released. This table summarises the main data that the handsets send to Apple and Google:

	IMEI	Hardware serial number	SIM serial number	Phone number	Device IDs	Location	Tele-metry	Cookies	Local IP Address	Device Wifi MAC address	Nearby Wifi MAC addresses
Apple iOS	✓	✓	✓	✓	UDID, Ad ID	✓	✓	✓	✓	✗	✓
Google Android	✓	✓	✓	✓	Android ID, RDID/Ad ID, Droidguard key	✗	✓	✓	✗	✓	✗

This data is sent even when a user isn't logged in (indeed even if they have never logged in).

In addition to the data listed in this table, iOS shares with Apple the handset's Bluetooth UniqueChipID, the Secure Element ID (associated with the Secure Element used for Apple Pay and contactless payment) and the Wifi MAC addresses of nearby devices e.g. of other devices in a household of the home gateway. When the handset's location setting is enabled, these MAC addresses are also tagged with their GPS location. And it takes only one device to tag the home gateway MAC address with its GPS location and thereafter the location of any other devices reporting that MAC address to Apple is revealed. Also note that sharing of these Wifi MAC addresses allows linking of devices using the same network, e.g. in the same household, office, shop, cafe, and so the construction of a social graph over time and place.

Both iOS and Google Android transmit telemetry, despite the user explicitly opting out of this. However, Google collects a notably larger volume of handset data than Apple. During The first 10 minutes of startup, the Pixel handset sends around 1MB of data to Google compared with the iPhone sending around 42KB of data to Apple. When the handsets are sitting idle, the Pixel sends roughly 1MB of data to Google every 12 hours compared with the iPhone sending 52KB to Apple i.e., Google collects around 20 times more handset data than Apple.



(a) Startup (b) Idle

In 2020 it is estimated that in the US there are 113M iPhone users and 129M Android users. Assuming all of the Android users have Google Play Services enabled then scaling up our measurements suggests that in the US alone Apple collects around 5.8 GB of handset data every 12 hours, while Google collects around 1.3 TB of handset data.
When the handset is idle, the average time between iOS connections to Apple is 264 seconds, while Android connects to Google on average every 255 seconds. In other words, both operating systems connect to their back-end servers on average every 4.5 minutes even when the handset is not being used. [I'll note that this is not cellular-layer tower connections. This is back to the mothership.]
With both iOS and Android, inserting a SIM into the handset generates connections that share the SIM details with Apple or Google. Simply browsing the handset settings screen generates

multiple network connections to Apple or Google. A number of the pre-installed apps/services are also observed to make network connections, despite never having been opened or used. In particular, on iOS these include Siri, Safari and iCloud; and on Android these include the Youtube app, Chrome, Google Docs, Safetyhub, Google Messaging, the Clock and the Google Search bar.

The collection of so much data by Apple and Google raises at least two major concerns:

Firstly, this device data can be fairly readily linked to other data sources. For example, once a user logs in (as they must to use the pre-installed app store) then this device data gets linked to their personal details (name, email, credit card etc.) and so potentially to other devices owned by the user, shopping purchases, web browsing history and so on. This is not [merely] a hypothetical concern, since both Apple and Google operate payment services, supply popular web browsers, and benefit commercially from advertising.

Secondly, every time a handset connects with a back-end server it necessarily reveals the handset's IP address, which is a rough proxy for location. The high frequency of network connections made by both iOS and Android (on average every 4.5 minutes) therefore potentially allow tracking by Apple and Google of device location over time. With regard to mitigations, of course users also have the option of choosing to use handsets running mobile OSs other than iOS and Android, e.g. /e/OS Android. But if they choose to use an iPhone then they appear to have no options to prevent the data sharing that we observe. They are unable to opt out. If they choose to use a Pixel phone then it is possible to startup the handset with the network connection disabled (to prevent data sharing), then to disable the various Google components (especially Google Play Services, Google Play store and the Youtube app) before enabling connection to the network. In our tests, this prevented the vast majority of the data sharing with Google, although of course it means that any subsequent phone apps must be installed via an alternative store and cannot depend upon Google Play Services (we note that many popular apps are observed to complain if Google Play Servicesis disabled).

However, further testing across a wider range of handsets and configurations is needed to confirm the viability of this potential mitigation. When Google Play Services and/or the Google Play store are used, this mitigation is not feasible and the data sharing with Google that we observe appears to be unavoidable.

Ethical Disclosure
The mobile OS's studied here are deployed and in active use. Measurements of Google Play Services backend traffic were previously disclosed by our group, but the present study is broader in scope.

We informed Apple and Google of our findings and delayed publication to allow them [time] to respond. To date, Apple have responded only with silence (we sent three emails to Apple's Director of User Privacy, who declined even to acknowledge receipt of an email. And we also posted an information request at the Apple Privacy Enquiries contact page at https://apple.com/privacy/contact but we have had no response).
Google responded with a number of comments and clarifications, which we have incorporated into this report. They also say that they intend to publish public documentation on the

telemetry data that they collect.

A key consideration is what mitigations are possible, and on what time scale can they be deployed. It seems likely that any changes to Apple iOS or Google's Android, even if they were agreed upon, will take a considerable time to deploy while keeping handset users in the dark for a long open-ended period seems incorrect.

My take on this is that mitigation of this data collection, as they described, is largely impractical. The only way to truly avoid it is to choose not to use a mobile handset. And who's going to do that in this day and age? These devices have become our Sci-Fi globally-connected super-powerful pocket computers. But they are also inextricably tethered to their mothers, and there's no breaking that bond. We know that all of the privacy downside the researchers painted for us is completely feasible.

Are Apple and Google actually performing all of that linking to build sophisticated social graphs of our connections at such a low level that it's inaccessible to us? They certainly could be. Sharing our phone's IMEI and SIM serial number seem benign and are likely necessary to provide the service we need. But there's really no way to characterize the aggregation and forwarding of every MAC address within the phone's reach, on an ongoing active basis, as anything less than surveillance. Apple may boast — and they certainly do — that they are unable to see inside a locked iPhone. But they certainly have the ability to tell law enforcement everything that iPhone has been and everyone it's been near. That's a pretty serious encroachment into every iPhone user's privacy.

I won't be giving up my phone; and I'm sure that few, if any, people will be. But it might behoove us to keep in mind that we are each carrying a little spy in our pocket.