Security Now! #762 - 04-14-20
Virus Contact Tracing

This week on Security Now!

This week we follow-up on a bunch of continuing Zoom news, since Zoom appears to be poised to become the teleconferencing platform of choice for the world at large. They've made more changes, have been sued and have been rapidly taking steps to fix their remaining problems. We have some browser news and another worrisome look into Android apps using a novel approach to quickly characterize them. We have an interesting and sad bit of miscellany and a progress report on my SpinRite work, and then we take the sort of full technical deep dive into the joint Apple/Google Contact Tracing system that our listeners have come to expect from this podcast. By the end of this podcast everyone will understand exactly what Apple and Google have done and how the system functions, in detail.

When the privacy policy is changed
AFTER **we're already using the product…**

Zoom Follow-Ups

Zoom rushed out another Zoom-bombing mitigation

Until they were updated, each of the three Zoom desktop clients (for Linux, Mac and Windows) were prominently (and conveniently) displaying the current Zoom meeting ID in each app's title bar. This behavior, which had caused no trouble at all during Zoom's first 9 years of life, since its birth in 2011, had suddenly become a significant liability as new Zoomers were sharing screenshots of their Zoom meetings on social media... without stopping to consider that their meeting ID was also being shared.

With last week's across-the-board update, the meeting ID has been moved from the title bar to a drop-down panel for the session's "info" icon, in the top-left of each Zoom app.

Additionally, Zoom has made the management (and, presumably the need) for security-related settings more obvious to meeting hosts. Last week's update adds a new dedicated Security icon to the app's control panel where the meeting organizer can manage all security settings from one place, rather than bounce around among separate dialogs as was needed before. These new settings include all of the things we talked about last week: The ability to lock meetings, enable the Waiting Room and more. And adding another new feature, meeting hosts can now enable the Waiting Room on the fly, even if the feature was turned off before the start of the meeting. But that will be less often needed since the update now also enables Waiting Rooms by default for all new meetings and also mandates that all new conferences must be password protected. So... Zoom went mainstream and lost its innocence.

And a Class-Action Lawsuit

And speaking of losing its innocence... Predictably, last Tuesday, one of the company's shareholders, Michael Drieu, filed a class-action lawsuit on behalf of all other shareholders. The suit alleges that Zoom made “materially false and misleading statements” that overstated its privacy and security measures, that it engaged in deception when it claimed that its product supported end-to-end encryption. It also alleges that Zoom only uses encryption for the transport link, allowing the service to still access user data and putting users “at an increased risk of having their personal information accessed by unauthorized parties, including Facebook. <sigh>

Meanwhile, Zoom has enlisted the aid of Alex Stamos

Zoom reached out to Alex Stamos who previously worked for Facebook and before that at Yahoo! as their Chief Information Security Officer. As we know, Alex departed Facebook in 2018 in protest over Facebook's handling of data-security practices surrounding the Cambridge Analytica fiasco and Russian interference in the 2016 U.S. presidential election. Alex had his eye on Russia's Facebook activity since July 2016 and wanted the company to go public with his findings. But Facebook's top execs including Zuckerberg refused.

Even though Stamos is already quite busy being an adjunct professor at Stanford’s Freeman-Spogli Institute, and a visiting scholar at the Hoover Institution, in a posting on Medium last Wednesday he said that he felt compelled to assist Zoom even though he is consumed by other commitments.

I want to share what Alex wrote because I was impressed by it and I think our listeners will find it interesting:

https://medium.com/@alexstamos/working-on-security-and-safety-with-zoom-2f61f197cb34

Last week, after I posted a series of tweets discussing the security challenges for Zoom and how they could respond, I got a phone call from Eric Yuan, Zoom's founder and CEO. We talked about the significant challenges his company was facing, both in responding to an incredible growth in users but also living up to the security expectations of the moment. He asked detailed and thoughtful questions of my experiences working at companies facing extreme crises, and I was impressed by his clear vision for Zoom as a trusted platform and his willingness to take aggressive action to get there. He asked if I would be interested in helping Zoom build up its security, privacy and safety capabilities as an outside consultant, and I readily agreed.

To be clear, I am not an employee or executive of Zoom and I don't speak for the company. I have refrained from any public comment on Zoom or discussions with journalists since my call with Eric, but in the interest of transparency I think it's important to disclose this work. I don't do a lot of consulting these days; I am generally quite busy with my role at Stanford and I'm proud of the work that team has been doing during this critical time for disinformation. This opportunity to consult with Zoom was too interesting to pass up, however, and I thought I would explain why I have embraced this challenge.

First off, Zoom has gone from being a successful mid-sized enterprise IT company to a critical part of the lives of hundreds of millions in the space of a couple of months. As my CV might suggest, I am attracted to difficult problems and this creates some doozies. As someone who has walked through the galaxy of blinking lights and deafening whir of tens of thousands of servers carrying the sessions of millions of users, I appreciate the effort it takes to build a product that scales. To successfully scale a video-heavy platform to such a size, with no appreciable downtime and in the space of weeks, is literally unprecedented in the history of the Internet. It has been clear to many people who have worked on production-scale systems that something special has been happening at Zoom, and the related security challenges are fascinating.

It's not just the technical challenges that I am interested in. In a time of global crisis, Zoom has become a critical link between co-workers, families, friends and, most importantly, between teachers and students. The morning Eric called me (and most mornings since) there were five simultaneous Zoom sessions emerging from my home, as my three kids recited the Pledge of Allegiance in their virtual morning assembly, my wife supported her middle-school students and I participated in a morning standup with my Stanford colleagues. Like many techies I have used Zoom professionally for a while, but I admit that there was still a bit of culture shock as my wife taped a daily calendar full of Zoom meeting codes to our eight year-old daughter's desk.

The adaptation of a successful enterprise collaboration tool into virtual classrooms, virtual doctor's offices and a myriad of other applications (including at least one virtual Cabinet Room) has created privacy, trust and safety challenges that no company has ever faced. As I told the computer science students in my Trust and Safety Engineering course this last quarter (the last two weeks of which were taught over, yes, Zoom) coding flaws and cryptographic issues are important, but the vast majority of real technological harm to individuals comes from people using products in a technically correct but harmful manner. Zoom has some important work to do in core application security, cryptographic design and infrastructure security, and I'm looking forward to working with Zoom's engineering teams on those projects.

Still, I'm certain that the real challenge, one faced by every company trying to provide for the diverse needs of millions seeking low-friction collaboration, is how to empower one's customers without empowering those who wish to abuse them. I encourage the entire industry to use this moment to reflect on their own security practices and have honest conversations about things we could all be doing better. This is possibly the most impactful challenge faced by the tech industry in the age of COVID-19, and together we can make something positive out of these difficult times and ensure that communications are safer and more secure for all.

Zoom creates a CISO Council
Last Wednesday, the day before Alex Stamos' blog posting on Medium, Zoom's CEO Eric Yuan also announced that Zoom had formed a CISO Council and an Advisory Board to collaborate and share ideas regarding how to address the videoconferencing platform's current security and privacy issues.

Yuan wrote in his announcement: "I am truly humbled that — in less than a week after announcing our 90-day plan — some of the most well-respected CISOs in the world have offered us their time and services. This includes CISOs from VMware, Netflix, Uber, Electronic Arts, HSBC, NTT Data, Procore, and Ellie Mae. The purpose of the CISO Council will be to engage with us in an ongoing dialogue about privacy, security, and technology issues and best practices — to share ideas, and collaborate."

What's next for Zoom?
As I noted last week, no one has yet taken a deep look into their crypto, and someone should. What we've seen is that their ECM mode (electronic code book) probably leaks repeating patterns from the plaintext. And if they have more deeply rolled their own crypto solutions, and if Zoom is headed toward becoming the world's teleconferencing platform, then one thing Zoom's CEO Eric Yuan ought to do -- and perhaps Alex Stamos or one of the CISO's will recommend it -- would be to open their system to an independent security audit. There were a number of stories last week about major companies banning all use of Zoom -- SpaceX and Google among them. I understand the emotional reaction to the initial security troubles. But most of the headline making was from high profile Zoom bombing, which was mostly the result of lax security measures on the part of the conference organizer. Due to the massive rapid uptake of Zoom, this was many conference hosts' first experience using Zoom. It should surprise no one that mistakes were made. And Zoom has already fixed those defaults.

That said, an independent security audit is now what Zoom needs.

Browser News

Chrome 81

On schedule -- or, on Coronavirus revised schedule -- Google released Chrome #81 last week.

As expected, Chrome is becoming less tolerant of mixed-content images. As we know, mixed content means that whereas the webpage is delivered over a TLS authenticated and encrypted connection, passive images are explicitly using HTTP rather than HTTPS. I say “explicitly” because if an image's URL leaves off any protocol specification the browser will default to using the same protocol (HTTPS in this case) as the underlying webpage. Until Chrome 81, mixed content was being allowed because it was regarded as “passive” content. Images, audio, video and objects are considered to be passive content, whereas scripts and iframes are considered to be active content so they were already rigorously blocked from mixed-content loading. But Chrome 81 changes this. From now on (unless this results in too much breakage), Chrome will override any explicit HTTP:// and replace it with HTTPS://. And if the asset cannot be delivered over HTTPS, well, that's too bad. The image won't be loaded and will show as “broken” on the webpage.

NFC: Chrome 81 also brings us “Web NFC” support. This will allow webpages to read and write to NFC tags when they are close to the user's laptop or computer. NFC uses near field RF technology (actually, NFC stands for Near Field Communications) having a range of 2 to 4 inches. So... nearly in contact with each other. The initial release of this API supports a widely compatible universal data interchange format known as NDEF which stands for NFC Data Exchange Format. It's a lightweight binary message format which is cross-NFC-tag compatible. At the moment apps are able to access a device's underlying NFC. We see this, for example, with Apple Pay on iOS. Having native NFC support in Chrome will make NFC available to a broad range of web sites. (What could possibly go wrong?)

32 security problems of varying degrees of severity were also fixed. None were Critical, three were rated high and the rest were about half and half medium or low. So not much to see there.

Firefox 75

We were just last week talking about the need to update Firefox to 74.0.1 to eliminate a pair of 0-day "Use After Free" memory vulnerabilities that were found being deployed in the wild in targeted attacks. Now we have Firefox 75. Firefox has upped the ante with their background telemetry collection. Firefox has been collecting telemetry for awhile and they have a page explaining what and why: https://support.mozilla.org/en-US/kb/telemetry-clientid

> Firefox collects telemetry data by default. We collect this to help improve the performance and stability of Firefox. Telemetry data is made up of two data sets: interaction data and technical data.
>
> Interaction data includes information about your interactions with Firefox to us such as number of open tabs and windows, number of webpages visited, number and type of installed Firefox Add-ons and session length, as well as Firefox features offered by Mozilla or our partners such as interaction with Firefox search features and search partner referrals.

Technical data includes information about your Firefox version and language, device operating system and hardware configuration, memory, basic information about crashes and errors, outcome of automated processes like updates, safebrowsing and activation to us. When Firefox sends data to us, your IP address is temporarily collected as part of our server logs. IP addresses are deleted every 30 days. If you are interested, you can read more about how we handle your data in our in-depth documentation about our data platform.

We do not know about your specific interactions with Firefox. We just know the number of tabs a user had opened and how long they were opened.

Entering "about:telemetry" into your Firefox URL will display your browser's telemetry collected information, and it's entirely possible to opt out of providing this feedback. They say: "You can opt out of sending any Firefox telemetry information at any time. If you opt out of sending telemetry data, we will also treat this as a request to delete any data we previously collected. Data will be deleted within 30 days after you opt out."

Under the Menu: Options / Privacy & Security / Under "Firefox Data Collection & Use" uncheck the various checkboxes.

So, what's been added to Firefox 75? Mozilla decided that they wanted to know what a user's browser choice was even if they were not using Firefox -- presumably after a user has changed their default away from Firefox. So Firefox 75 now contains a separate process that's launched by the Windows task scheduler once a day to "ping" a report back to Mozilla. They explained...

With Firefox 75, we're launching a new scheduled task for Windows that will help us understand changes in default browser settings. As with all other telemetry related changes here at Mozilla, this scheduled task has gone through our data review, a process designed with user choice and privacy at its core.

- We're collecting information related to the system's current and previous default browser setting, as well as the operating system locale and version. This data cannot be associated with regular profile based telemetry data. If you're interested in the schema, you can find it here. https://github.com/mozilla-services/mozilla-pipeline-schemas/pull/495/files#diff-48f14d6bdea5bf803f8b8cff5f018172 (I checked it out, and it is innocuous.)
- The information we collect is sent as a background telemetry ping every 24 hours.
- We'll respect user configured telemetry opt-out settings by looking at the most recently used Firefox profile.
- We'll respect custom Enterprise telemetry related policy settings if they exist. We'll also respect policy to specifically disable this task.

So now everyone's on the same page with Chrome 81 and Firefox 75. And remember that since Chrome's jump to 81 was delayed, Google still plans to skip 82 entirely and deliver 83 on its regular schedule. (And they are still imagining that 83 might have TLS v1.0 and v1.1 eliminated, but I'll be surprised if that happens this year.)

Security News

Android Apps Again in the Crosshairs

The research was conducted by researchers at Ohio State University, New York University and the Helmholtz Center for Information Security (known as CISPA). The paper that resulted was titled: "Automatic Uncovering of Hidden Behaviors From Input Validation in Mobile Apps."

The sheer volume of Android apps submitted to the Google Play store means that an automated first-pass screening system is the only possible means for even trying to put a dent in the task of discovering those that might have a hidden purpose. Otherwise (and even so) it's possible to simply get lost in the crowd.

> Abstract—Mobile applications (apps) have exploded in popularity, with billions of smartphone users using millions of apps available through markets such as the Google Play Store or the Apple App Store. While these apps have rich and useful functionality that is publicly exposed to end users, they also contain hidden behaviors that are not disclosed, such as backdoors and blacklists designed to block unwanted content. In this paper, we show that the input validation behavior—the way the mobile apps process and respond to data entered by users—can serve as a powerful tool for uncovering such hidden functionality. We therefore have developed a tool, INPUTSCOPE, that automatically detects both the execution context of user input validation and also the content involved in the validation, to automatically expose the secrets of interest. We have tested INPUTSCOPE with over 150,000 mobile apps, including popular apps from major app stores and pre-installed apps shipped with the phone, and found 12,706 mobile apps with backdoor secrets and 4,028 mobile apps containing blacklist secrets.

They noted that the problems they discovered were not just theoretical. Later in their paper they wrote: https://web.cse.ohio-state.edu/~lin.3021/file/SP20.pdf

> Nor are such cases hypothetical: by manually examining several mobile apps, we found that a popular remote control app (with 10 million installs) contains a master password that can unlock access even when locked remotely by the phone owner when the device is lost. Meanwhile, we also discovered a popular screen locker app (with 5 million installs) uses an access key to reset arbitrary users' passwords to unlock the screen and enter the system. In addition, we also found that a live streaming app (with 5 million installs) contains an access key to enter its administrator interface, through which an attacker can reconfigure the app and unlock additional functionality. Finally, we found a popular translation app (with 1 million installs) contains a secret key to bypass the payment for advanced services such as removing the advertisements displayed in the app.

What they realized was that secret backdoors or other behaviors are often accessed through the front door. So they ran their static code analysis tool "InputScope" against the input handling logic of Android applications. And they found a significant number of designed-in misbehavior. And these were not obscure apps. They took the top 100,000 most popular on Google Play, plus 30,000 apps pre-installed on Samsung devices, and 20,000 taken from the alternative Chinese market Baidu.

The study examined two issues – what proportion of apps exhibited secret behaviours and how these might be used or abused.

Of the 150,000, 12,706 exhibited a range of behaviours indicating the presence of backdoors (secret access keys, master passwords, and secret commands) plus another 4,028 that seemed to be checking user input against blacklisted words such as political leaders' names, incidents in the news, and racial discrimination.

Looking at backdoors, both Google Play and apps from alternative app stores such as Baidu showed roughly the same percentage of apps falling into this category, 6.8% and 5.3% respectively.

Interestingly, for pre-installed 'bloatware' apps, the percentage showing this behaviour was double the other sources at around 16%. This finding supports a public open letter sent to Google's Sundar Pichai in January by Privacy International that criticised the way pre-installed apps were often not scrutinised for privacy and security problems. As a separate Spanish study last year documented, the provenance of pre-installed apps is often shadowy, based on commercial tie-ups between phone makers that the end user would not be aware of.

The team took a closer look at 30 apps, picked at random from apps with more than a million installs, finding that one installed with the ability for someone to remotely log into its admin interface. Others could reset user passwords, bypass payment interfaces, initiate hidden behaviours using secret commands, or just stop users from accessing specific, sometimes political content.

In Sophos' coverage of this research they wrote: "Perhaps the biggest consequence from the study is simply how many Google Play apps exhibit these behaviours. While the Play Store is large, the fact that several thousand [from among the top 100,000] apps have hidden backdoors hardly inspires confidence. And there is currently no easy way, short of the sort of weeks-long analysis carried out by the researchers using a dedicated tool, to know which apps operate in this way."

Sophos concluded: "That's not so much a backdoor as a blind spot, another problem Google's sometimes chaotic Android platform could do without."

I don't mean to pick on Android. But Google boasts the incredible size of its installed base. All that honey attracts flies.

Sandboxie goes Open Source

And speaking of Sophos, Sandboxie, which has for more than 15 years been a Sophos project, has been released into the open source community for its continued maintenance and evolution. Security Now! Episode #172, dated May 27th, 2008 was titled "Sandboxie" where we took a deep dive into its operation. While it was originally intended to sandbox web browsers, its use has expanded into a general purpose application isolation wrapper. https://www.sandboxie.com/

Miscellany

I discovered “Scientific American” magazine in my high school years. It was incredibly influential for me, since its science writing was pitched at just the right level for me. Toward the back of every issue, was a monthly column called "Mathematical Games" written by Martin Gardner. And Gardner was quite influential, too. Wikipedia noted that Gardner's “Mathematical Games” column became the most popular feature of the magazine and was the first thing that many readers turned to. In September 1977 Scientific American acknowledged the prestige and popularity of Gardner's column by moving it from the back to the very front of the magazine. The column ran from 1956 (the year after I was born) to 1981, with sporadic columns afterwards and was the first introduction of many subjects to a wider audience.

And the point of this bit of history is that it was his October 1970 column, published early in my sophomore year of high school, where Gardner introduced his readers to John Horton Conway's amazing "Game of Life." We’ve spoken of Conway's Game of Life many times through the years of this podcast. Conway's creation was incredibly elegant in the simplicity of its rules, and the complexity of its results. And it rather clearly divided all people who were exposed to it into two camps: Those who thought it was the coolest thing they had ever seen, and those who thought that the first group might benefit from medication. Needless to say, I was a member of the first camp and the game consumed me for quite some time.

Wikipedia has a terrific page about Conway's Game of Life. Anyone who is listening to this podcast who doesn't already know what I'm talking about MUST go look at the Wikipedia page for “Conway’s Game of Life”. All of our desktop PCs and smartphones have implementations of Conway's Game of Life. You will no longer be bored while waiting for this pandemic to subside.

I bring all this up, because tragically, last Saturday April 11th, COVID-19 claimed the life of John Horton Conway. Wikipedia writes:

John Horton Conway was an English mathematician active in the theory of finite groups, knot theory, number theory, combinatorial game theory and coding theory. He also made contributions to many branches of recreational mathematics, most notably the invention of the cellular automaton called the Game of Life. Conway spent the first half of his long career at the University of Cambridge in England, and the second half at Princeton University in New Jersey, where he held the title John von Neumann Professor Emeritus. On 11 April 2020, at age 82, he died of COVID-19 at his home in New Jersey.

SpinRite

I am finally making very good progress toward the next SpinRite. I've had a surprising number of challenges getting to where I've finally been for the last 4 days.

Since I plan to be spending a lot of time during this development cycle, working on a combination of SpinRite v6.x releases and the Beyond Recall utility, I wanted to invest in the creation of an efficient development environment so that I would not be spending a lot of time in repetitive and unnecessary overhead.

The lack of support for 16-bit code by today's 64-bit OSes, coupled with the fact that my toolchain depends upon a number of 16-bit components proved to be more troublesome than I expected. I also needed to link the DOS testing targets, which only know about SMBv1 to file shares on Win10, which strongly resists having anything to do with that original SMBv1. I also spent a week trying to get the Open Watcom remote debugger to work. It's the only remote debugger that can still debug a DOS target machine. It's incredibly finicky, but I finally managed to get it working, only to discover that it does not handle "included" code in source files, so I could not use it. I finally ended up returning to my tried and true pure 16-bit tools. But I did finally achieve my goal, in spades. I now have a very high-efficiency development and testing environment established and working.

And using that, I've been writing code for Intel's AHCI controller spec when in AHCI mode. Prior to this, the work that I had been doing was writing to the "Legacy mode" of the controller which, back in 2004, all systems still had. And the next SpinRite will know how to talk directly to all of that hardware at full speed without the BIOS. But most modern systems are now operating in AHCI mode and some of the newest hardware has dropped support for legacy mode altogether since it's no longer needed by any OSes.

So, once I got my working environment established four days ago, I plowed into the AHCI controller spec and began writing code. What I found is that Intel's AHCI controller is an amazing piece of engineering. A week ago it seemed large, mysterious, daunting and opaque. Today, I can report that it is clear and obvious to me. I've obtained the mindset of the engineers who originally defined it and it all makes perfect sense. I cannot wait to put it through its paces! The moment I'm finished with this podcast I am going to eagerly return to coding and I expect to have some first test code for our testing gang to play with shortly.

Virus Contact Tracing

How the technology works and what it does, in detail:

The Master "Tracing Key" is unique per device, 256-bits and is obtained from a high entropy source. The Tracking key NEVER leaves the device.

The Daily "Tracing Key" is a half-size (128-bit) key deterministically derived from hashing the master Tracing Key with the UNIX epoch day number (the number of days since January 1st, 1970). Therefore, this "Tracing Key" changes once per day as determined by the Master Tracing Key. But, being on the other side of an HMAC-based key derivation function, the Master Tracing Key is never revealed and the daily sequence cannot be predicted. However, as we'll see this is important, the device that contains the Master Tracing Key CAN, itself, recreate any previous day's Daily Tracing Keys that it may choose to.

This is important since, if the user of the device should subsequently test positive for COVID-19, the user can instruct their device to upload -- to a regional health server -- a block of previous days' daily tracing keys for the period during which they may have been contagious. But we're getting a bit ahead of ourselves. We'll come back to this is a minute.

A so-called "Rolling Proximity Identifier" is a 16-byte (128-bit) value sent out as the beacon from all participating devices. As we have discussed several times in the past, to prevent Bluetooth tracking, all modern devices already randomize their Bluetooth MAC addresses. Those Mac addresses change on a random schedule between once every 10 to 20 minutes, so on average every 15 minutes. This system's "Rolling Proximity Identifier" changes in sync with the MAC address.

When the user has enabled Contact Tracing, every time the Bluetooth MAC address changes, a new "Rolling Proximity Identifier" is generated. Since the Bluetooth MAC address and the Rolling Proximity Identifier change synchronously -- since no single identifier straddles MAC addresses and no single MAC address straddles identifiers -- it's not possible to link and track Contact Tracing by MAC address.

The value of the 128-bit "Rolling Proximity Identifier" is also fully deterministic. It is derived from the Daily Tracing Key and a 10-minute window number from the start of the day.

So, everyone who is voluntarily participating in this Contact Tracing system is emitting 128-bit identifiers which is ultimately derived from their Master Tracing Key, which, in turn creates a Daily Tracing Key, which, based upon the time of day with 10-minute granularity, creates a changing Proximity Identifier. And notice that since the MAC address changes every 10 to 20 minutes, and the Rolling Proximity Identifier is calculated on a 10-minute scale, no two sequential MAC addresses can ever have the same Rolling Proximity Identifier.

Okay...

So we have a system where participating devices are broadcasting an identifier which changes unpredictably once every 10 to 20 minutes, so on average every 15 minutes, and every participating device is also collecting all of the similar incoming 128-bit "Rolling Proximity Identifiers" from everyone nearby. For every Identifier received, it checks to see whether it already has seen that one, and if not it adds it to a list of the identifiers it has seen. Since there is no need to retain the received Rolling Proximity Identifier's forever, they can be and will be deleted from the receiving devices after they have aged-out at 21 days.

Now...

Someone who has been participating tests positive for COVID-19. So, at their discretion, to help the world and to help protect their friends and family, they choose to notify everyone whom they may have been in contact with during the days prior to their diagnosis. So they instruct their device to notify the world.

Their device, alone, contains their Master Tracing Key which, based upon the UNIX epoch day derives their daily "Tracing Key". So they go back some number of days and upload each day's Tracing Key during which they may have been contagious. We relabel these “Tracing Keys” as “Diagnosis Keys” once they have been associated with a COVID-19 positive individual.

So think about this: ALL of the Rolling Proximity Identifiers which were emitted by that user's device, on any given day, were derived solely from that daily Tracing Key and the time of day. That means that anyone else who obtains those daily Tracking Keys is also able to re-derive the same set of Bluetooth beacons that the COVID-19 positive user's device transmitted that day.

So, on the receiving end, every participating user's device periodically downloads all of the daily "Tracing Keys" from people in their region who have tested positive for COVID-19 and have chosen to anonymously share their status. For any new keys that the device has not already seen -- and presumably the download could be selective by day -- the user's device simply recreates the original "Rolling Proximity Identifiers" which are derived from the newly downloaded key and compares them to the list of stored Rolling Proximity Identifiers they have received while out and about.

If there are any matches, what the recipient user knows is that someone whose device theirs was close enough to, to exchange Bluetooth beacons has posted their Daily Tracing Keys to an online contact tracing server.

So, what can go wrong?

Moxie Marlinspike, who has earned and deserves everyone's respect as a crypto-savvy researcher and the developer of Signal, did a stunning job with Signal. And he apparently assumed that Apple and Google did less. He fired off a bunch of tweets which were all wrong and which, because he's certainly as capable of understanding this system as its developers and as we all now do, can only mean that he was working not from the system's specifications -- as I have -- but from a conversational description of the way it works... which necessarily omitted the clever details.

Specially, as we've seen, users are not uploading or downloading megabytes of beacon data. Thanks to the fact that a day's worth of Rolling Proximity Identifiers are deterministically re-derivable from a single 125-bit Daily Tracing Key, the system's bandwidth requirements are quite modest. And since a great deal of attention was obviously paid to the synchronization between Bluetooth MAC addresses and the Rolling Proximity Identifiers that they carry, his attacks about MAC-based tracking also miss the mark.

One non-malicious problem that could arise is that "radio proximity" is not a perfect match for "viral propagation proximity." Your device might well be exchanging beacons with the people in the apartment above, below and to the sides of yours. But that doesn't mean that they represent a source of viral contagion to you.

And on the malicious side, we don't yet know what will prevent fraudsters from hanging out in large crowds so as to have their devices spreading proximity beacons, and to then upload their Daily Tracing Keys to the community's health tracing servers. Since this is such an obvious problem which could cause such havoc, and since the rest of the system has been so well thought through, it must be that there will be tight controls placed upon who is able to post their daily keys to these public resources.

Overall, I'm very impressed by the system. It is clean, simple, not over-engineered and is rigorously privacy-preserving. No identifying information nor location data is ever involved.

The plan is to initially (and quickly) build only an API for a downloaded OS-specific App. But Apple and Google have indicated that they plan to set about building this contact tracing into their underlying OSes to increase its efficiency, reduce power consumption and make it an integral part of the system.

So I say Bravo Apple & Google. Details remain to be worked out, but we're starting with a very solid and well thought out foundation.