Location Tracker Behavior

Description: This week we're going to answer only two questions. First, why hasn't Steve been saying anything about his work on SpinRite recently? And then second, what are all the details spelled out in the emerging specification for the detection of unwanted location tracking?

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-923.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-923-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. We've got of course the Picture of the Week, nice little cartoon about AI. We've got an update on his software project SpinRite. We'll talk a little bit about a software coding philosophy, even as it applies to his favorite language, assembly language. And then we're going to dive deep, as we did last week, into location tracker behavior, in particular the future spec that both Google and Apple want to adopt for these little Bluetooth location trackers. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 923, recorded Tuesday, May 16th, 2023: Location Tracker Behavior.

It's time for Security Now!, the show where we cover the latest security news with our champion, Mr. Steven Gibson. Hi, Steve.

Steve Gibson: Hey, Leo. It's great to be with you for Episode 923, where we're going to continue, we're going to pick up basically where we left off after last week's episode, talking about the behavior of location trackers. And it was my intention to do what I always do, or almost, "almost" being the operative word in this case because I didn't do it, which is to actually talk about some news in addition to our main subject.

Leo: Well, you had to prepare the way beforehand.

Steve: Well, yes. And the problem is doing this specification of AirTag and other location tracking behavior justice, I just - the deeper I got into it, the more I had to unwind it because the spec turns out to be really poorly written. It's all out of order. They refer to things in other places. And it's like, okay, wait, what? So anyway, and I ended up with 18 pages without ever talking about any news. So this is one of our - this is a pure deep dive.

We're going to answer only two questions this week. First, why haven't I been saying anything about my recent work on SpinRite? And second, what are all the details spelled out in this emerging specification for the detection of unwanted location tracking? On the other hand, I think everybody's going to find this very interesting and fascinating. And I will do the two previous weeks of news next week for our podcast. So, and of course we do have another good Picture of the Week. So I think another great couple hours of entertainment for our listeners, and a lot of information.

Leo: As you always do, Steve. Always.

Steve: So our Picture of the Week is a fun one, apropos of many of the things we've been talking about relative to AI. It's a two-frame cartoon. The first frame on the left has someone sitting in front of his computer. He's very excited. He's explaining to a friend standing behind him: "AI turns this single bullet point into a long email I could pretend I wrote." Very excited about that. And then in the second frame we have a gal sitting in front of her computer. Oh, I should mention, on the left-hand screen we just seen one little bullet point. On the right-hand screen, we see her screen is full. And she's explaining to her friend: "AI makes a single bullet point out of this long email I could pretend I read."

Leo: It's almost not even a joke.

Steve: I know.

Leo: It's actually true. Oh, my god, yeah. I think at Google I/O they practically showed this as a feature; right? Which is good. Let the AI handle the email.

Steve: Yeah. Besides, no one likes typing on those touch screens.

Leo: Yeah.

Steve: So just, you know, give it a clue. Give it a hint.

Leo: Let the AI do it.

Steve: Let it know how you're feeling, and it'll just go ahead and respond on your behalf. So, that's right. Okay. So before I get into our main topic, which is going to dominate the rest of the podcast, I did want to mention why I've been mute for the past couple of weeks about ongoing work on SpinRite. And that's not because of any slowdown in the action, but because of a major revamping of SpinRite's primary core data recovery code. As it happens, it sometimes occurs that software gets itself painted into a corner. Or a programmer paints themselves into a corner. Or maybe the software paints the programmer into a corner. I don't know. But paint and corners always seems to be involved.

This can occur when an inexperienced programmer tackles a task that they don't fully understand. You know, they jump right in and start writing code, and at some point they

realize that they can't get where they need to go from where they are. They're in a corner. And at the start of our careers, we coders, that's a common occurrence to many of us who love and live to code. It can also happen, however, when a sufficient quantity of new information arrives late in a software design cycle which requires that the code be modified in a way it wasn't designed to be, to incorporate the needs created by this new information. And since I checked, I did some subtraction, since I've been coding now for more than 60, six zero, years, inexperience is not my shortcoming.

What happened is that I - and SpinRite's first Alpha release - were wholly unprepared for the shocking myriad ways that today's and yesterday's highly distressed mass storage devices might and do fail. Everyone's heard me mentioning for many months that nearly all of SpinRite's currently 683 registered development testers have mostly been bored. Everything has always just worked for them. That's why I released an Alpha, as I thought, okay, here we go. It looks great. But while SpinRite is valuable to those whose drives are still healthy - it's proven through several decades to be particularly adept at helping to keep drives that way - it really comes into its own when it's working to resurrect the data on drives that are far from healthy, for whatever cause, you know, it got dropped by mistake, or just got too old, or the power failed in the middle of it writing some data or something.

So I was so close to being able to declare this work finished when a final couple of unexpected surprises landed. For example, though I have never seen this happen, a literal reading of the latest ATA-8 specification, which specifies the exact operation of mass storage drives, states that if an error occurs during a data transfer, the drive will abort the transfer and indicate the sector of the first error it encountered. Well, right. That's fine. That's what you'd expect. But it then goes on to say that nothing can be inferred about what data may have been transferred before that error occurred. Now, as far as I know, all drives will transfer all of the data preceding the erroneous sector. But the specification says they no longer need to. And who knows what technology might be used in the future, or how far behind in transferring its data a drive might be when it decides to stop due to encountering an error. And since SpinRite 6.1 obtains its breathtaking speed by transferring 16MB in a single request, that could be important.

So from a practical standpoint, this means that when an error occurs, to be safe, SpinRite needs to identify the trouble, then re-request all of the data up to, but not including, the trouble. Get all that data settled, then deal with the sector that caused the transfer to halt. That was the straw that finally broke the camel's back, as they say. It wasn't just that. I could have once again fixed that. It was the "once again" part. SpinRite's all-new core data recovery code, which I've been so pleased with, and which I have had every intention of moving directly from SpinRite 6.1 into SpinRite 7, had become a mess over the past six months as I had needed to keep poking and prodding it as we kept discovering new and surprising ways that drives could misbehave.

On top of that, SpinRite users had drives that were so badly damaged that they were hanging. So they wanted to be able to interrupt SpinRite at virtually any point to immediately get out and try another drive. SpinRite wasn't ever built to support that sort of emergency user escape because we weren't seeing that.

I was so close to being done that the last thing I wanted to do was to sacrifice that investment. But my code had become so clogged up with special cases, exception testing, and early-out, you know, like bailout code from inner loops, that it started to feel brittle rather than robust. So I scrapped it.

I recognized that the real investment that had been made was in what I had learned since that first Alpha-1 release. So I settled down with my favorite text outliner, and I completely reconceived SpinRite's entire data recovery system. I'm willing to confess this today because I completed that entire rewrite last Sunday night, two nights ago, and it's

been a long time since I have been so happy with any code. It is now immaculate. All of the worker code has exactly zero tests for any special case exceptions. I moved that handling into SpinRite's common IO abstraction.

There are two "top level" places in SpinRite during this data recovery, one where SpinRite is starting to work with a block of sectors, and the other where SpinRite is starting to work on a single sector. Immediately upon entering the top of either of those worker routines, I checkpoint the system's stack by saving the current stack pointer in a static global variable. Then, if anything happens that requires us to immediately surface from wherever we are, no matter how deep and how nested with routines we are, the IO abstraction system, through which all control passes on its way to and from the drivers, now has the ability to simply reset the stack to its previously saved state. When it returns, that will have the effect of unwinding and discarding all current call stacks and local variable storage. It immediately terminates anything that was going on, whatever it was.

That design expedient allowed me to have exactly zero tests or concerns for anything that might go sideways throughout the rest of the data recovery code. Now, as a consequence, it is breathtakingly clean because the data recovery code no longer needs to test for anything. And it now also adds support for the wacky ATA-8 spec in case drive designers ever do take the spec as literally as it says they could.

And there's a broader lesson here that has probably occurred to anyone who codes a lot. There really is an art to coding. And Leo, I know you appreciate this. And the act of solving a complex problem teaches the coder so much about how to approach and solve the problem that it's really only after they finally have their program running that they understand how it should have been done.

A couple hours ago, purely by coincidence, during MacBreak Weekly's pre-show, Alex Lindsay told the story of the four months he first took to solve a complex program in 3D graphics, which in this case was rotating a 3D pyramid on a Commodore 64. This was obviously a while ago.

Leo: That's a long time, yeah.

Steve: Shortly after he finally had it working, after four months, the machine crashed, and he lost all of his work. You know, maybe it was the cassette tape that we used back then...

Leo: Yeah. [Sound effects]

Steve: ...to save and load our programs. In any event, after recovering emotionally from the loss, Alex told the story of how he reimplemented what he originally had in just three weeks.

Leo: And it was much better, probably, because that's - you know?

Steve: Exactly, that which had taken him four months to do the first time. And, he said, it was only after that, that he then realized how it should be done. So he rewrote it again in four days.

Leo: Yeah.

Steve: And the way Alex expressed it was perfect. He expressed it as three phases. He said: "Do it. Then do it better. Then do it right." And he said that's like his mantra now. That's the way he approaches things because he recognizes things that are hard, you know, the reason we're engaged by them is that they're hard.

Leo: Right.

Steve: But they're not boring because we're learning from them.

Leo: Right. I love it. That's one of the reasons I use Lisp - I'm not recommending it, well, I mean, I would, but no one will ever do it - is because it has that kind of iterative style built into it. It's a very natural way to work in Lisp, especially because of the REPL and the way the debugger works. You can actually change a program while it's running. So it's very easy to - it's a good tool for experiment. This is how I write. I'm not writing real code like you or Alex. But I like to experiment and try stuff and write a first iteration that I know isn't going to be the final version. And then you're not attached to it. It's easier for you to kill it.

Steve: And you're able to get in.

Leo: Exactly.

Steve: It's often just getting in.

Leo: Just the first line is the hardest, yeah, yeah. That's right, yeah.

Steve: So to wrap this up, in my present case, you know, in retrospect, because again, the other reason we code is we're like, okay, what happened? How did this happen? When that first little innocuous exception was needed, the quickest answer, which seemed right at the time, was to simply add a couple of tests for it. And when it turned out to be needed in other places, I had already established a template for how to, quote, "solve" that need, in air quotes. So I applied the same tests elsewhere. Then when another type of exception emerged, I did the same since by then I'd established a precedent.

But after six months of that, my code had become a mess. I knew how to solve it, which is what I just did. But right up until the moment that I finally bit the bullet, it was more important to me to be almost finished. The good news is that now I'm almost finished again, although the gang in the .dev group will have a lot of fresh virgin code to pound on while we verify that it's working exactly the way we want.

Leo: "Almost finished," when a programmer says that, means something completely different. I just want to point out.

Steve: Exactly. And it's why, you know, my dear wife is saying, "So, honey, how much longer do you think?" And I just say, you know, "I have no idea." I would tell everybody if I did. It's not like I'm holding out.

Leo: Tell them what Michelangelo told Pope Julius when he was painting the Sistine Chapel. "It's done when it's done. Get out of here. I'm working."

Steve: Right. So anyway, this time I will have a beautiful legacy, which SpinRite 7 will be able to inherit from SpinRite 6. And just as I was reading this, I realized that the other cool thing about this is that I have now sequestered all of that special case crap, that exception handling, in one location. So if any additional, like, need for special casing occurs, it's got a happy place to live.

Leo: Also a good idea, yeah.

Steve: And it won't ever muck up the rest of my code.

Leo: Higher level languages I think kind of facilitate that a little bit better. It's probably really tempting with assembly language to just do it when you think of it, stick it in there and it's all over the code base.

Steve: It's three bytes. It's three bytes.

Leo: But you're disciplined, and I'm sure that you'd do that. But assembly language doesn't give you any - it's like bowling without bumpers. You really can't - you could throw a gutter ball pretty easy.

Steve: Well, there is in high-level languages now, because the need for this has been recognized, it's called "structured exception handling," where you create a block of stuff where you know there may be a problem. And if there is, then it's like without constantly having to test for it throughout your code, you've told it where you want to go in the event that that happens. And that's a little bit like what this is, though as you said, in assembler I have the advantage of knowing the structure of the stack, knowing all of what's on it.

I did have to make sure that my code, there wouldn't be any side effects from just aborting, like at any time, the inner loops of my code makes a call for some IO with the drive which may never come back. And that's essentially the power that I gave the IO abstraction layer is, if it sees a problem, it just discards all of the stack. And when it returns from it, it goes back to the top level.

Leo: Yeah, that's another thing Lisp does really well, what they call their common Lisp condition system, but it has exception handling built into it. And it's very powerful. It's a really - it's kind of, I mean, I'm sure that what they're doing in other languages is kind of inspired by it. It's really quite amazing.

Steve: I would love to know whether McCarthy put that in in version 1.

Leo: He did not. No, he did not. And that's what's - I look forward to the day when you don't have anything to do. And then I have a friend who says, don't learn golf until I can teach you. Don't learn Lisp until we can talk about it. I can't teach you a thing. But Lisp is kind of amazing. So McCarthy was really taking symbolic logic and algebraic expressions and turned it into a programming language.

But the common Lisp now has evolved like 30 or 40 years of evolution since McCarthy thought of that in 1956 and is really, I think, where a lot of academic research and a lot of academic thought about what's the right way to write code kind of ended up in common Lisp and its successor Scheme. And Racket's a very good example of that. But the common Lisp condition system is kind of famous for exactly that, taking all the errors and making sure that you can handle them. And then you don't have to worry about errors because everything will be okay. We'll see.

Steve: Okay. Last week, I opened our look at the forthcoming standardization of consumer location tracking technology by writing: "It seems that any powerful new technology gets used for both the benefit and detriment of society. In other words, it's a mixed blessing. And so is the case for AirTags, those popular and handy Bluetooth LE (Low Energy) dongles that are all about their location."

Now, as for popularity, I should note that Apple announced their $29 tags two years ago last month in April of 2021. So in the past two years, they alone have sold more than $1 billion dollars worth of AirTags, amounting to around 55 million of those little buggers. And Tile, another tracking tag company, told Wired magazine last year that they had sold about 40 million of their popular little devices. So we know that consumers love the concept; and, when tightly integrated with smartphones, they also love their ease of use. A single Apple account can be associated with up to 16 AirTags.

As our listeners also know, we ended last week with the controversial tidbit that the forthcoming "Detecting Unwanted Location Trackers" IETF specification requires that all qualifying devices be registered in an online database that could be queried by law enforcement. Leo's immediate and certainly understandable reaction was that this would be a deal breaker for the technology. I then posed this question over in GRC's Security Now! Newsgroup to gauge the feelings of those there, and many people felt the same way, often quite vocally.

I would argue that nothing is going to stop the sales of these handy little things for use in the legal and ethical tracking of people's own property. I doubt that anyone using them to keep track of their car keys or backpack will care. And I believe that the specter of a registration database - actually, I hope that the specter of a registration database - that can be queried by law enforcement when probable cause has been established is what's needed to help curtail the abuse of what really is a very powerful consumer technology.

So I think it's perfect that Leo and I may have differing feelings about this and that we'll be able to discuss the pros and cons which will make for a more interesting and balanced podcast for our listeners. It's already clear that the issues are not entirely black and white. But one thing that did arise from my posting into GRC's newsgroup was that there was significant mis-presumption about many aspects of this technology. That's understandable since this technology has not yet been elucidated. Which is why we're here today.

One example of mis-presumption is that people assumed that Apple themselves would be aware of the location of everyone's AirTags and could track them. The technology that Apple has deliberately designed makes that explicitly impossible. And it has always been possible for anyone discovering an AirTag to directly query the device to obtain the last

four digits of its owner's phone number. This is done to help people determine whether a Tag they may discover might belong to someone obvious, who they know.

Now, I recognize that querying an AirTag you have found is very different from some evil database in the sky. We'll get back to that by the end of our technical walkthrough. To me, and I may be alone in this, which is fine, the idea that casual consumers should be entitled to the absolute private tracking of others, which is illegal in many jurisdictions, seems against society's best interests.

Okay. Before we get any deeper into the pros and cons, let's learn how this technology operates and what it means for users and abusers. Once I had fully absorbed this detailed 22-page specification, which I mentioned was a hodgepodge, I mean, it is - there's typos. There's grammatical errors. I was wishing that I had an editor open on it rather than just reading a PDF because...

Leo: That's kind of discouraging.

Steve: Yeah, I mean, it's rough, and all out of sequence. You're having to, like, wait a minute, why are we talking about that here when it should go over there? So, you know, let's hope they get this fixed. But more importantly, I was very disappointed by what was missing from it. I was hoping that we were going to get a full working technical specification for the operation of Bluetooth Low Energy trackers with all of the crypto and the other details spelled out. I wanted that because I wanted to be able to present a detailed walkthrough of that level of the technology which Apple has developed to ensure the privacy of the various parties. Instead, what we have is a small subset of that whole. And I've been unable to locate anything more complete. I don't think that's been published yet.

This document does, however, provide us with the common behavior that's required from small tracking devices to enforce their owners' privacy and the privacy of anyone they may be near. There's still plenty of interesting stuff to talk about, but we're going to need Apple or Google or someone to publish another specification to fully satisfy our curiosity.

What today's specification does is outline how location tracker accountability is created. And it's also about enforcing the privacy of the users of location trackers. In other words, users should not misuse tracking power, and there should be fair accountability if they do. But neither should their legal tracking in any way compromise the privacy of what they are tracking. As I got more deeply into this, and brought myself up to speed about the features and operation of Apple's current AirTag technology, that is, what we've been using for the last two years, it appeared that most of this has already been implemented by Apple.

So this began to feel more like Apple finally taking the covers off of some of the existing AirTag technology that they had already developed as a means of encouraging industry-wide standardization. For all of these low-power, short-range Bluetooth and NFC tags, this is important since, in this entire model, "location" is determined by others in a crowd-sourcing model.

Whereas fancy and expensive, technically you need to have a cellular account which costs something every month, traditional GPS-style trackers, they use both GPS satellite positioning and built-in cellular telephony for reporting back to headquarters. Compared to that, all of these little Bluetooth Low Energy tags, you know, BLE tags, rely upon having their periodic broadcasts picked up and their position relayed by other bystanders' smartphones. This means that enlisting all of Android and potentially all other devices

into one big happy crowd-sourced family - pardon me. I've been suppressing a cough there.

Leo: Let it go. Let it all out.

Steve: This means that enlisting all Android and potentially other devices into one big happy crowd-sourced family will significantly improve the tracking experience and responses for everyone. One New York Times reporter, writing last year about this exploding industry in consumer tracking devices, planted multiple sets of three different types of trackers on her husband with his full consent and knowledge.

And she noted that, while he was near their sparsely populated home, she didn't get much updating on his position with any of the non-GPS trackers. But the moment he ventured into "the city," presumably New York, the tracking information available exploded and became absolutely real-time. That was due to the fact that so many people were carrying iPhones which were autonomously pinging the AirTag he was carrying, or rather it was pinging them, and they were all simultaneously reporting its location as being near to them at that moment. And since iPhones are adept at knowing where they are, the tag's location could be inferred from that.

For those who are not familiar with RFC-style terminology, I should preface what comes next by noting that the spec in RFC style uses all-caps qualifiers so that its reader understands how various requirements should be taken. Specifically, the capitalized phrases used are "MUST" - whose implication is obvious - "MUST NOT," "REQUIRED," "SHALL," "SHALL NOT," "SHOULD," "SHOULD NOT," "RECOMMENDED," "NOT RECOMMENDED," "MAY," and "OPTIONAL." So keep that in mind.

And with that in mind, it was interesting to see the section on Applicability where the specification states that these best practices are REQUIRED (in all caps) for location-enabled accessories that are small and would not be easily discoverable on their own in the world. For large accessories, such as a bicycle, these best practices were merely RECOMMENDED, as opposed to REQUIRED. Accessories are considered easily discoverable, and thus recommended but not required, if they meet one of the following three criteria: The item is larger than 30 centimeters in at least one dimension. Okay, so that's about a foot. If any dimension is larger than a foot, it's recommended but not required. Or the item is larger than 18x13 centimeters in two dimensions. That's about 5x7 inches. Or the item is larger than 250 cubic centimeters in three-dimensional space. If it was a cube, that would be 6.3 centimeters on a side, or about 2.5 inches on a side.

Leo: So if it's the size of a refrigerator, you don't have to do it.

Steve: If it's bigger than a breadbox, Leo, that's right.

Leo: And that's because everybody knows it's there.

Steve: That's right. It's not going to, you know, it's like...

Leo: Not hiding.

Steve: Why is R2D2 following me? Well, at least you know it is.

Leo: Yeah.

Steve: So, you know, this detecting unwanted tracking specification is meant to protect from itty-bitty trackers, you know, like Apple's AirTags, which are disks 2.25 inches in diameter by about .3 inches thick. As I said, if it's bigger than a breadbox, it doesn't need to rigidly follow the specification.

One of the people who responded to my GRC posting noted that he had a pair of very nice electric bikes stolen from their garage, and he was wishing that they had been trackable. With the rapidly growing popularity of this technology, I think it's foreseeable that such high value, desirable, and inherently mobile objects, such as a very nice bicycle, will incorporate tracking as a sales feature.

Leo: Oh, there already are for ebikes, yeah.

Steve: Exactly. Wondering whether that had already been done...

Leo: It has.

Steve: A quick google revealed a story from just last month which was titled "Why a Bike with Built-in Find My Capabilities Is Total Genius."

Leo: Apple showed it, I think, when they showed off AirTags. They were talking about third-party uses of this technology.

Steve: Yes, yes. So, and the article had three bullet points. It said: "Velotric's" - bad name, Velotrics.

Leo: I think it's Velotric's.

Steve: Okay, that's somewhat better.

Leo: Because bicycles were originally called Velopeds. So it's an electric Veloped.

Steve: Velopeds and Velotric's. Got it.

Leo: Yeah, electric Veloped.

Steve: Yeah. Anyway, so it's their new...

Leo: You sound skeptical.

Steve: Their new - I don't. I was calling that router the MikroTik router for a long time.

Leo: Yeah, for a long time, yeah.

Steve: I don't think they appreciate that.

Leo: I don't know how they pronounce it.

Steve: MikroTik's sounds like something you take medicine for.

Leo: Yeah.

Steve: No, not good. So anyway, this thing is the Thunder 1, the ST ebike, has a built-in AirTag feature, says you can track your bike just like you can track your iPhone and your AirPods. And finally it said: "Find My is potentially more private and much cheaper than a dedicated cellular GPS tracker." And to that I would add that having it built in, it can run off the ebike's battery, so avoiding the annual otherwise needed battery replacement. And more importantly, assuming that the technology is integrated into the ebike's electronics, it cannot be discovered and removed.

The final point is that hopefully the ebike's console will prominently advertise the fact that it incorporates built-in antitheft AirTag tracking so that the lazy thief who wants a free ride will think twice and just walk away instead. So there is every indication that for such items, especially when they become cross-platform for compatibility with both iOS and Android, that we're going to begin seeing AirTag tracking becoming a competitive marketing feature for these kinds of devices where a little bit of incremental price or cost increase justifies the cost.

Okay. So moving on. I want to next share some formal definitions which the specification uses. It'll help in understanding what comes later, although I'm going to also be deobfuscating this as we go. So this is more than just defining terms since a lot of reading between the lines is also possible. So the spec says: "Throughout this document, these terms have specific meanings." The term "platform," which we sort of have a sense for, in this case, in this document, it's used to refer to mobile device hardware and associated operating systems like phones, tablets, and laptops. So the platform is what we're carrying in our pockets, not the AirTag, which they insist on calling an "accessory."

The term "owner device" is a device that is paired to the accessory and can retrieve the accessory's location. You know, so a smartphone. Then the term "non-owner device" refers to a device that may connect to an accessory, but is not an owner device of that accessory. The term "location-tracking accessory" refers to any accessory that has location-tracking capabilities, including but not limited to crowd-sourced location, GPSS, WiFi, cell tower, et cetera, and provides the location information back to the owner of the accessory via Internet, cellular connection, and so forth. So that sort of just broadly location-enabled tracking accessories, non-AirTag-ish things, and also of course AirTags.

The term "location-enabled state" refers to the state an accessory is in where its location can be remotely viewed by its owner. The term "location-enabled advertisement payload"

refers to the Bluetooth advertisement payload that is advertised, which is the term, you know, broadcast, when an accessory has recently, is currently, or will in the future provide location updates to its owner. And I have no idea how it knows it will be able to in the future, but okay. The term "unwanted tracking" refers to undesired tracking of a person, their property, or their belongings by a location-enabled accessory. The term "unwanted tracking detection" refers to the algorithms that detect the presence of an unknown accessory traveling with a person over time.

"Unwanted tracking alert" refers to notifying the user of the presence of an unrecognized accessory that may be traveling with them over time and allows them to take various actions, including playing a sound on the accessory if it's in Bluetooth Low Energy range. And there's a lot more we'll be getting to. And finally, the term "platform-compatible method" refers to a method of communication between the platform, which, you know, is the thing we're holding in our hand, our pad or whatever, and the accessory/accessory manufacturers to exchange information, including but not limited to Bluetooth, advertisements, HTTP, and so forth. So that's sort of a dumb term. I don't know why it's even there, but it is.

Okay. So I find it awkward to refer to tags as "accessories," which is what the spec does. So I will tend to use the simpler and clearer term "tag." But when I'm quoting the specification I'll use its language of referring to these tags which are "accessories."

So tags can be in one of two major modes, and this affects their behavior. They can be in so-called Near-Owner Mode or Separated Mode. And the tag knows. Each tag periodically emits a broadcast hoping to be heard by some passing smartphone or Bluetooth-enabled device. And later we'll see that by "periodically" we mean every half second to two seconds. So these things are quite chatty. If the tag is within range of the owner's smartphone, and if actually any of the owner's multiple devices that are on the same account, with which it has previously been paired, the owner's smartphone notifies the tag that its owner is nearby. So that's what places the tag into its Near-Owner Mode is it does a ping, and the owner's device says yeah, you know, you're here. That, among other things, suppresses its location broadcasts.

The specification states: "The accessory SHALL" - again, all caps, SHALL - "transition from Separated to Near-Owner mode if it has reunited with the owner device for a duration no longer than 30 minutes." But I suspect this transition is typically immediate because there's no reason for it not to be. The specification is likely leaving some latitude there. "Conversely," the specification states, "the accessory SHALL transition from Near-Owner mode to Separated Mode if it has been physically separated from the owner device for a duration no longer than 30 minutes." Okay, now, since BLE's range is limited, it makes sense for a tracking device to not immediately flip into Separated Mode. But this says that it must do so within 30 minutes of being away from its owner.

One of the definitions we just noted above was for "location-enabled state." The specification further explains that by saying: "The accessory SHALL maintain an internal state that determines when its location is, or has been, available to the owner via a network. This state is called 'location-enabled state.' Misuse of location-enabled accessories can occur when the owner's device is not physically with the accessory. Thereby, the accessory SHOULD maintain a second internal state, denoted the 'Near-Owner state'" - which is being a little redundant here - "which indicates if the accessory is connected to or nearby one or more of the owner's devices. Near-Owner state can take two values" - you know, true or false - "either Near-Owner Mode or Separated Mode. Near-Owner Mode is denoted as the opposite of Separated Mode." They're being very clear about this.

When the device is in "location-enabled state," it is broadcasting a payload containing the most recent location it has been able to obtain. This payload is, not surprisingly, called

the "location-enabled payload," and the spec says: "It is RECOMMENDED that the location-enabled payload is only advertised when the accessory is in its Separated state. The reasoning behind this recommendation is that unwanted tracking detection relies on the Bluetooth Low Energy advertisements emitted while in the location-enabled state to determine if an unknown accessory is traveling with someone who is not the owner. If the location-enabled payload is advertised only in the Separated state, that minimizes false-positive unwanted tracking alerts."

Okay, well, now, that's sort of obvious. The location-enabled payload contains the most recent location information that the accessory tag was able to obtain from some nearby smartphone, which is what let it know where it was. So it would make no sense for a tag to be broadcasting its location when it's nearby and therefore in its Near-Owner Mode. The spec adds: "The accessory SHALL broadcast the location-enabled advertisement payload if the location is available to the owner or was available any time within the past 24 hours. This allows unwanted tracking detection to operate both between and beyond the specific moments an accessory's location is made available to the owner.

It's this so-called "location-enabled advertisement" that is the primary indication of tracking. That's the Bluetooth advertisement that sets off alarms in nearby phones. The idea being that the accessory tag has somehow obtained a location fix within at least the past 24 hours. And being a good little tracker, it's hoping to find someone who will dutifully relay its location back to its owner. In other words, this is tracking. It's not necessarily unwanted tracking, but it's definitely some form of tracking.

This should be disabled when the device is with its owner, since it makes no sense to be broadcasting its location in that situation. The false positive unwanted tracking alerts referred to earlier, that are being eliminated by the suppression of location-enabled advertisements, occurs, for example, if two or more people, each having smartphones, are together, and somebody also has a tag somewhere that belongs to one of them. If they're all traveling together, the tag is also traveling with the non-owners. So if that tag were to be emitting its location-enabled advertisements, then the non-owners' phones would believe that they were being tracked and would generate false positive alerts. So the fact that you have the proximity of the owner placing the tag into its Near-Owner mode, and thus suppressing these overt tracking announcements, is a good thing. And that's where you get your false-positive suppression.

And speaking of these Bluetooth Low Energy advertisements, as I mentioned before, they are quite frequent. The specification's "advertising policy" formally specifies an announcement interval of between half a second and two seconds. So, you know, lots of this happening. Bluetooth Low Energy devices have MAC addresses, and therein lies a problem. In the spec, we're wanting to prevent the tag from being tracked by an adversary. So on one hand, there's not having the tag being with someone whose owner is using it to track. But then there's also just the tracking of the tag itself. That wants to be prevented. So here's what the spec has to say about that.

"The Bluetooth Low Energy advertisement payload SHALL contain a resolvable and private address for the accessory, which is the 6-byte Bluetooth LE MAC address. The address MUST be private, and it MUST rotate periodically and be unlinkable. Otherwise, if the same address is used for long periods of time, an adversary may be able to track a legitimate person who is carrying the accessory. A rotation policy aims to reduce this risk.

A general approach to generate addresses meeting this requirement is to construct them using a pseudorandom function, taking as input a secret key established during the pairing of the accessory and either a counter or a coarse notion of time. The counter or coarse notion of time allows for the address to change periodically. The secret key allows

the owner devices to predict the sequence of addresses for the purpose of recognizing its paired accessories.

"An accessory SHALL rotate its resolvable and private address on any transition from Near-Owner state to Separated state, as well as any transition from Separated state to Near-Owner state." So, like, as soon as it realizes it's no longer with its owner, bang. Immediately the MAC address changes. And similarly, as soon as it is back within the region of its owner, the MAC address changes again. But it also changes autonomously. When in near-owner state, the accessory SHALL rotate its resolvable and private address every 15 minutes. This is a privacy consideration to deter tracking of the accessory by non-owners when it is in physical proximity to the owner. Since it is nearby, the owner device is able to maintain synchronization so that it's able to recognize and remain paired with its known accessories.

"When in a Separated state, the accessory SHALL rotate its resolvable and private address every 24 hours. This duration allows a platform's unwanted tracking algorithms to detect that the same accessory is in proximity with it for some period of time, when the owner is not in the tag's or accessory's physical proximity."

So this seems well thought out. Devices rotate their 8-byte MAC address on a schedule as directed by a secret key. This is exactly analogous to the TOTP one-time passwords many of us are using today, though with six 8-bit bytes rather than only 6 digits. By knowing each device's secret key, all future MAC addresses can be determined, and no one tracking a device based on its periodic broadcasts will be able to determine any tag's next MAC address.

It's worth noting that the devices have no actual native MAC addresses. Unlike the anti-tracking technology we've developed for smartphone WiFi, there is not one rotating spoofed MAC address used when roaming and another actual physical fixed MAC address used when associated or paired with a WiFi home base or access point. So these tags are all simply using the traditional 8-byte, 48-bit MAC as a short-term rotating ID. And there's a galaxy of them, you know, a hundred million of them out there now and growing, all occasionally changing their identifier following a predictable pattern that only their owner knows. Another intriguing aspect of this specification is what non-owned devices, that is, the tags that are in their Separated state, are required to do to reveal themselves. The spec calls this "Non-Owner Finding," and it has a number of components.

The spec says: "Once a user has been notified of an unknown accessory traveling with them, it is REQUIRED that they have the means to physically locate the accessory. This is called non-owner finding of the accessory. These capabilities are both REQUIRED and RECOMMENDED and reflect hardware to be incorporated into the accessory to enable non-owner finding." And now we'll explain it.

"Motion detection: The accessory SHOULD include a motion detector that can detect accessory motion reliably, for example, an accelerometer. If the accessory includes an accelerometer, it MUST [all caps] be configured to detect a change in orientation of plus or minus 10 degrees along any two axes of the accessory. After some number of hours between eight and 24, chosen randomly from a uniform distribution" - so that would be an average of 16 hours, so no fewer than eight, no more than 24, but an average of 16 - "the accessory being away from its owner and in a Separated state, the accessory's motion detector will be enabled. While the motion detector is enabled, it must be able to detect motion within 10 seconds." Okay, so it's sampling its position at 10-second intervals, presumably to conserve power. The spec says: "If motion is not detected within the 10-second period, the accessory MUST stay in this state until it exits Separated state."

Okay. So accessory leaves the owner. After some time between eight and 24 hours from then, the motion detector is activated. And at 10-second intervals, it starts sampling its angular position in space. The spec says: "If motion is detected within the 10 seconds between samples, the accessory MUST play a sound. After any motion is detected, the movement detection period is decreased from 10 seconds to half a second." Okay, so it starts sampling much more quickly after first detecting any motion. "The accessory MUST continue to play a sound for every detected motion. The accessory SHALL disable the motion detector for six hours under either of the following two conditions: Motion has been detected for 20 seconds at the half-a-second sampling rate, or 10 sounds have been played."

Okay, so after this thing has become motion sensitive, which after an average of 16 hours of being aware from its owner, then any motion of more than 10 degrees on two axes will cause it to make a sound, and will also cause it to shorten its sampling to half a second, and to continue making sounds if it continues being moved, until either 20 seconds have passed or 10 sounds have been played. After that point it then goes into a six-hour, they call it a "back off," so it goes to sleep for six hours.

The spec says: "If the accessory is still in its Separated state at the end of the six-hour back off" - which has gone silent, essentially, it's not going to just keep beeping or screaming, whatever it's going to be doing - "the unwanted tracking behavior MUST restart." So again, after six hours of quiet time, if it is again moved, it starts making sounds. "Any Bluetooth LE connection from a paired device MUST reset the Separated behavior and transition the accessory to connected state." In other words, the instant it's back with its owner, then it's paired, and it shuts this down, and it goes back into near-owner state.

And finally they said: "The accessory MUST include a sound maker, for example, a speaker of some kind, to play sound when in Separated state, either periodically or when motion is detected. It MUST also play sound when a non-owner tries to locate the accessory by initiating a play sound command from a non-owner device when the accessory is in range and connectable through Bluetooth LE. The sound must be loud, and the sound must be played for a minimum of five seconds each time." And in the spec it goes into a detailed specification of the measuring of loudness the devices are able to have.

Okay. So this gives us a system where, after a tracking device has been separated from its owner for an unknown interval of time, randomly chosen between eight and 24 hours, its motion detector activates. At that point it begins taking readings of its angular orientation in three space every ten seconds; and if it finds that it's been rotated by more than 10 degrees through any two axes between successive position samples, it will emit a clearly audible sound for five seconds. And if any qualifying movement is detected within those 10 second intervals, essentially that will have roused it so that it will start sampling twice per second to allow it to make its noise much more responsively to someone who may be attempting to discover its location by moving things around.

All of this occurs whenever a tag is separated from its owner for between eight and 24 hours. It's about requiring anything that is small that can track to deliberately reveal itself periodically, and in doing so using simple sound which does not require any technology where the tag is located.

Note that a lot of attention has been given to detecting unwanted tracking tags with another smartphone. But not everyone has a phone that's smart enough to do so today. Today, unless that AirTag Apple app is launched and running on an Android phone, no Android-carrying user would be able to detect an unwanted nearby tag. This spec will be changing that soon for new Android devices. But we know that there will remain many

non-upgraded Android devices in use for years, and there are also non-smart cellular phones.

I was recently listening to a talking head on some show suggesting that one way to keep young people away from the perils of social media would be to equip them with only a dumb phone capable of making and receiving telephone calls, texting, and taking pictures with its camera. Now, I don't know whether this person has ever actually been around any young kids, but that would be a tough sell when they're surrounded by their peers who are gleefully deep into the social Internet. So good luck with that. But still, the point being not everybody is carrying a cell phone.

So there's clearly a need to expose trackers through some low-tech means, and sound is the obvious choice. This is not what someone wishing to track someone stealthily would choose, right, because the device is going to give itself away. After that initial period of eight to 24 hours of separation from all of its owner's devices, any tag that remains separated will generate attention-getting sounds whenever it's significantly moved. And once it has done so 10 times or for 20 seconds of movement in its faster sampling mode, it will go quiet for six hours, after which it will again reawaken and notify of any movement.

The obvious weakness in this system is that tracking requires radio, but not sound. So arranging to enclose a tag inside some sort of acoustic suppression container which is transparent to radio might defeat the tag's audible "Help me! I've become separated from my owner!" cries for help. In addition to physical movement which will trigger sounding, any nearby device within radio, you know, Bluetooth LE radio range of a tag, whether a tag's owner or non-owner, is able to remotely command any tag to emit its sound. So if, for example, a suspected tracker is detected, the detecting smartphone's user interface for managing tags can request any unseen tags to sound off to aid in determining their location.

Tags also contain a wealth of queryable information. This includes a unique 8-byte token, a UUID, that serves as a unique identifier for the accessory make and model. The 8-byte value will be listed in a public registry so that the tag's issuing company can be determined. Tags also contain the manufacturer's name in an up to 64-byte field, so plenty of length. And also another 64 bytes for the model's name, so that can be made clear. There's also an 8-byte accessory category indicator. Only the first single byte of the eight bytes is presently defined, though eight have been set aside; and many of the values of that byte are already defined.

To give you a sense for how the publishers expect these tags are going to be used, the enumerations of the one byte that's been defined are things like a generic Finder. Then there's also Other, also Luggage, Backpack, Jacket, Coat, Shoes, Bike, Scooter, Stroller, Wheelchair, Boat, Helmet, Skateboard, Skis, Snowboard, Surfboard, Camera, Laptop, Watch, Flash Drive, Drone, Headphones, Earphones, Inhaler, Sunglasses, Handbag, Wallet, Umbrella, Water Bottle, Tools or Tool Box, Keys, Smart Case, Remote, Hat, Motorbike, Consumer Electronic Vehicle, Apparel, Transportation Device, Sports Equipment, and Personal Item.

Leo: Well done. You just passed our cognitive test. You do not have dementia, Steve. Congratulations.

Steve: And had I been able to do that from memory, Leo, I would agree with you. So anyway, that gives us some sense for what the people behind this are thinking about the future and about the wide potential for this location tracking technology. I suspect that there's every chance that, as I said, that many higher end consumer products like that

ebike, which are prone to misplacement, loss, or theft, or which might need to be located with some urgency, such as an inhaler, may eventually incorporate this consumer location technology as a sales feature.

The last item of the queryable data is a 4-byte, 32-bit value which enumerates a tag's capabilities. Only four bits are currently defined. Those are supports sound, supports motion detection, supports serial number lookup by Near Field, and supports serial number lookup by Bluetooth. There's also some interesting specification about deliberate disablement. The spec says: "The accessory SHALL have a way to be disabled such that its future locations cannot be seen by its owner. Disablement SHALL be done via some physical action, for example, a button press, a gesture, or the removal of the battery.

"The accessory manufacturer SHALL provide both a text description of how to disable the accessory, as well as a visual depiction, for example, image, diagram, animation, et cetera, that MUST be available when the platform, you know, the thing that, you know, the smartphone, is online and OPTIONALLY when offline. Disablement procedure or instructions CAN change with accessory firmware updates.

"A registry which maps product data to an affiliated URL supporting retrieval of disablement instructions SHALL be available for platforms for reference." You know, again, remember that "platforms" in this spec refers to smartphones, pads, and similar devices with full user interfaces. Then they said: "This URL must return a response which can be rendered by an HTML view."

So this says that, if someone discovers an unwanted tracking device, that 8-byte registered product ID data, which can always be retrieved directly by querying the device over Bluetooth, will, among other things, point to a URL which returns HTML that any smartphone can render to obtain clear and updated instructions from the device's manufacturer about how to manually disable the tracker. And this is all in, you know, capital MUSTS. But this always requires physical access to the device. At least at this point in the evolution of the specification, it cannot be done remotely over radio. It's foreseeable that this might be allowed over NFC, since that requires essentially physical proximity, and it might be simpler. As we'll see, the spec does makes clear distinctions about what can be done via NFC and what can only be done over Bluetooth. Or rather Bluetooth and only under NFC.

All devices are also serialized. The spec explains: "The serial number SHALL be printed and be easily accessible on the accessory. The serial number MUST be unique for each product ID. The serial number payload SHALL be readable either through NFC tap or Bluetooth LE." So again, that can be done at a distance. "The serial number payload SHALL be readable either through NFC tap or Bluetooth LE." Note that this does not mean the serial number itself. I'll get to that in a second.

They said: "For privacy reasons" - oh, they're going to say it. "For privacy reasons, accessories that support serial number retrieval over Bluetooth LE MUST have a physical mechanism, for example, a button, that SHALL be required to enable the remote 'Get_Serial_Number' opcode command. The accessory manufacturer SHALL provide both a text description of how to enable serial number retrieval over Bluetooth LE, as well as a visual depiction - again, image, diagram, animation, et cetera - that MUST be available when the platform is online and OPTIONALLY when offline. The description and visual depiction CAN change with accessory firmware updates. A registry which maps Product Data to an affiliated URL that will return a text description and visual depiction of how to enable serial number lookup over Bluetooth LE SHALL be available for platforms for reference. This URL MUST return a response which can be rendered by HTML view."

So we have the same set of requirements for enabling the retrieval of a device's serial number remotely over Bluetooth as we do for disabling the device. In both cases, some

sort of physical action must be taken with the device by an individual to prove that it is in their physical possession - either to disable it and/or to learn its immutable serial number. It's obviously required for the serial number since, unlike its MAC address, which is nothing but a keyed pseudorandom sequence, the serial number never changes. And if it were available remotely without contact with the device, it could be used for long-term tracking. But as I noted before, NFC, which is, as its name suggests, a Near Field contact technology, is allowed greater latitude.

The spec says: "For those accessories that support serial number retrieval over NFC, a paired accessory SHALL advertise a URL with parameters. This URL SHALL decrypt the serial number payload and return the serial number of the accessory in a form that can be rendered in the platform's HTML view."

Okay. So first of all, NFC. If the tag and user device both support NFC, then obtaining the device's serial number is as simple as tapping the smartphone against the tag which you have in your possession to query it. But we also just encountered the phrase "decrypt the serial number payload." Okay. So here's what that's about:

"The 'Get_Serial_Number' opcode is used to retrieve serial number lookup payload over Bluetooth LE or NFC. This MUST be enabled for five minutes upon user action on the accessory," you know, the button press for, for example, 10 seconds to initiate serial number read state. "When the accessory is in this mode, it MUST respond with 'Get_Serial_Number_Response' opcode and Serial Number Payload operand. For security reasons, the serial number payload returned from an accessory in the paired state SHALL be encrypted. A registry which maps Product Data to an affiliated URL which will decrypt the serial number payload and return the serial number SHALL be available for platforms to reference. This URL MUST return a response which can be rendered by an HTML view."

So this tells us that the data contained within the public registry is encrypted, and that only the tag device itself contains the information required to decrypt its own publicly stored serial number. When a user encounters a tag that's in its Separated state, and either presses a button on the tag or taps it for a Near Field exchange, that smartphone receives a URL from the tag which contains the address of the lookup and also the decryption keying information needed to decrypt the publicly stored and encrypted serial number. Thus only someone who is in physical possession of a tag may use the tag to obtain its own serial number electronically.

But remember that the serial number must also be physically printed on the device. This suggests that this electronic fallback is provided in case some would-be malicious tracker person thinks they're getting very clever by sanding off the device's exposed and obvious serial number to make it anonymous. Well, that won't work.

Since the serial number is an important piece of the whole, the spec further explains: "Serial number lookup is required to display important information to users who encounter an unwanted tracking notification. It helps them tie the notification to a specific physical device and recognize the accessory as belonging to a friend or relative. However, the serial number is unique and stable, and the available partial user information can further make the accessory identifiable." And we'll get to that in a second. This spec has a lot of, as I said, out-of-sequence content.

The spec says: "Therefore, the serial number SHOULD NOT be directly available to any requesting devices. Instead, several security and privacy preserving steps SHOULD be employed." It says: "The serial number lookup SHALL only be available in Separated mode for a previously paired accessory. When requested through any long-range wireless interface like Bluetooth, a user action MUST be required for the requesting device to access the serial number. Over NFC, it MAY be acceptable to consider the close proximity as intent for this flow.

"To uphold privacy and anti-tracking features like the Bluetooth MAC address randomization, the accessory MUST only provide non-identifiable data to non-owner requesting devices. One approach" - oh, and by the way, it doesn't want to be repeatable, either; right? Because that would also be like having something that's not changing. So whatever it is that the device provides has got to be changing. "One approach," they say, "is for the accessory to provide encrypted and unlinkable information that only the accessory network service can decrypt. With this approach, the server can employ techniques such as rate limiting and anti-fraud to limit access to the serial number. In addition to being encrypted and unlinkable, the encrypted payload provided by the accessory SHOULD be authenticated and protected against replay.

"The replay protection is to prevent an adversary using a payload captured once to monitor changes to the partial information associated with the accessory, while the authentication prevents an adversary from impersonating any accessory from a single payload. One way to design this lookup encryption," they wrote, "is for the accessory to contain a public key for the accessory network server. For every request received by a device nearby, the accessory would use the public key and a public key encryption scheme to encrypt a set of fields including the serial number, a monotonic counter or a one-time token, and a signature covering both the serial number and counter or token.

"The signature can be either a public key signature or symmetric signature, leveraging a key trusted by the network server which MAY be established at manufacturing time or when the user sets up the accessory. Some additional non-identifiable metadata MAY be sent along with this encrypted payload, allowing the requesting device to determine which accessory network service to connect to for decryption, and for the service to know which decryption key and protocol to use."

So possession of the tag, which would usually allow someone to simply look at its serial number, is essentially duplicated electronically. The tag won't directly inform even a nearby device of its serial number. But it will provide a URL, once a button is pressed or Near Field has been used, with embedded decryption data that such a device can then use to query and then obtain the device's true plaintext serial number. Again, it was also printed on there, so it's not a big deal. But this allows it to be obtained electronically. By asking a remote service for this information, its disclosure can be controlled. And that's a cool part, too; right? So since someone is having to query a server using a URL that provides the data, you can do rate limiting, and you can do various anti-fraud things.

Okay. And so now we get to the so-called "pairing registry," where we ended our truncated discussion last week. To reiterate against the background of what we now understand, the spec says: "Verifiable identity information of the owner of an accessory at time of pairing SHALL be recorded and associated with the serial number of the accessory, for example, phone number, email address, and so forth."

Okay. So the process of beginning to use a new tracking tag is to pair it with the user's so-called "platform" device, you know, phone, a tablet, whatever. And most users will, in turn, have their real-world identity and billing account information associated with their device. So this pairing associates the tag's globally unique serial number with the user's real-world identity. The point of this being to create an accountability link from the tag to the real-world user with whom the tag has been paired.

Later, under the heading "Persistence," the spec says: "The pairing registry SHOULD be stored for a minimum of 25 days after an owner has unpaired an accessory. After the elapsed period, the data SHOULD be deleted.

We covered in detail the steps required to obtain, decrypt, and display a device's serial number. But when that serial number is displayed, so too is some deliberately obfuscated owner information that's obtained from the pairing registry. In other words, when you go

through getting the serial number electronically, you also get some obfuscated information from the pairing registry. The spec says: "A limited amount of obfuscated owner information from the pairing registry SHALL be made available to the platform" - the user's device - "along with a retrieved serial number. This information SHALL be part of the response of the serial number retrieval from a server which can be rendered in a platform's HTML view.

"This MUST include at least one of the following: the last four digits of the owner's telephone number," you know, asterisks for everything except -5555 in their example; "or an email address where the first letter of the username and the first letter of the domain name are visible, as well as the entire top level domain." So as we said last week, s****@g**.com, if my email address were steve@grc.com, which it hasn't been for several - for quite a while.

Leo: So don't send him email at that address.

Steve: Yeah, that won't help. Okay. Elsewhere, under the heading "Privacy Considerations," the spec says: "In many circumstances when unwanted tracking occurs, the individual being tracked knows the owner of the location tracker. By allowing the retrieval of an obfuscated email or phone number when in possession of the accessory, this provides the potential victim with some level of information on the owner, while balancing the privacy of accessory owners in the arbitrary situations where they have become separated from those accessories.

So, you know, the idea being that someone who might immediately react with, like, "Who the 'F' is tracking me," this allows the individual to immediately obtain some information. It's just a matter of pressing a button on your phone. The phone will get the serial number, make the query, the remote server decrypts it, returns it in HTML to the phone, and you will get the serial number and some information that hopefully provides you with some relief. It's like, oh, that's where that tracker went or whatever. So it may often be someone that you know or can recognize.

So we have a pairing registry which associates the real-world identity of the pairer with any of the tracking tags that they have paired. Even if a third party discovers or obtains one of these tags and uses the system to get as much information about the tag's owner as possible, the absolute limit of that information will be heavily obfuscated identity information such as the last four digits of the registered phone number, an email address and so forth. So who has access to the fully unobfuscated pairing registry? And the answer is law enforcement. But there are some caveats. Paragraph 3.15.3, titled "Availability to law enforcement," simply reads: "The pairing registry SHALL be made available to law enforcement upon a valid law enforcement request."

Nothing in this technical specification indicates how high the evidentiary bar would be set for such law enforcement queries to be honored. But the point is that the identity of any tag-pairing party is tightly controlled for non-authorized access to this information. The only information that anyone possessing a tag they are not paired with can obtain is that obfuscated identity information. So one purpose of the pairing registry is to aid in immediately resolving the question of who might be tracking you. But since the pairing registry itself contains fully non-obfuscated identity information, it's clear that the reason for this is to deliberately and explicitly create end-user accountability for the use of what is quite powerful consumer tracking technology.

I said earlier that I had the sense that at least part of this was Apple formalizing and publishing the technology that they had already established and deployed in the interest of making it an industry-wide and worldwide standard. In any crowd-sourced model,

bringing Android into the fold would make the system far more useful for everyone. So I did some further digging. And actually it was earlier this morning. And what I found was that none of this is new, and that this law enforcement accessible pairing registry has always existed.

In one instance Apple writes - this is Apple: "AirTag, AirPods, and other Find My network accessories include features to guard against unwanted tracking. They should not be used to track people, and should not be used to track property that does not belong to you. Using these products to track people without their consent is a crime in many countries and regions around the world. If an AirTag, set of AirPods, or Find My network accessory is discovered to be unlawfully tracking a person, law enforcement can request any available information from Apple to support their investigation.

And since even that was a bit noncommittal, I dug some more. On February 22nd of last year, so 15 months ago, February 22nd of 2022, 15 months ago, in their article titled "An update on AirTag and unwanted tracking," they make this very plain and explicit. They said: "We have been actively working with law enforcement on all AirTag-related requests we've received. Based on our knowledge and on discussions with law enforcement, incidents of AirTag misuse are rare; however, each instance is one too many." This is Apple.

"Every AirTag has a unique serial number, and paired AirTags are associated with an Apple ID. Apple can provide the paired account details in response to a subpoena or valid request from law enforcement. We have successfully partnered with them on cases where information we provided has been used to trace an AirTag back to the perpetrator, who was then apprehended and charged. Law enforcement has shared their appreciation for the assistance we've provided in helping them find the source of unwanted tracking. We've identified additional improvements we can make in the information we share and the educational resources we provide, and we will be taking action, including making updates to our law enforcement documentation."

Well, the surprise in this specification, of a registry whose real-world identifying data can be made available under due process of law, caught us off guard. And I had picked up on it because the tech press was highlighting it as being a surprise. So actually no one should have been surprised. It's not a new thing. This does, of course, beg the question, who knows? And who would care if they knew? We know that Apple has sold more than one billion dollars' worth of AirTags, and that Tile has sold more than 40 million units of their own tags. No one appears to care about tag ownership tracking. But I suspect that not many people know or have stopped to wonder.

Leo: Yeah. I didn't know, or I might have thought twice.

Steve: Right. So one reason is that even if they have no idea how any of this works, they implicitly trust Apple. And Apple has never given us any reason to suspect that trust is misplaced. If I were to trust anyone to work hard to keep my information private, based upon all the evidence we've seen during the 17-plus years of this podcast, it would be Apple. So the question is, would anyone particularly care if they knew?

I know that if I had something of value that I needed to track, like a bicycle or my luggage, I would not hesitate now to pair those tags to my real-world identity, knowing that any query about the ownership of those tags would need to get past Apple's legal department first. And I have to say that I've never really seriously thought about AirTags before, but I have now. Even though we've covered them on this podcast whenever they've made the news. Now knowing what I know, I am absolutely going to purchase some AirTags, and I am never going to again travel without having them in my luggage.

We know that they won't begin squawking until they've been separated from us for a minimum of eight hours, or on average 16 hours. So if they're in the belly of a plane along with us, they're going to be remaining silent until we pick them up after a flight. And just knowing that wherever they might be, we'd have some way to know, seems like a bargain to me for 29 bucks each. And I think that, being a bit sneaky, I would have one AirTag exposed and visible hanging out on an AirTag dongle, which are available, with another tag deeply hidden and tucked away rolled up in a sock so that if anyone nefarious did want to make off with our stuff, they might stop looking after removing the obvious tracker.

I'm glad that we took the time to look into this deeply since I've never really thought about it. And I'm glad that Apple is making the effort to work with Google and their Android platform on this, so that we wind up with a single global standard which, due to the crowd-sourced nature of positioning, will make it much more powerful and attractive. And next week, back to the news.

Leo: I've got a hammer. And I've got an AirTag. Which, by the way, I've got to point out I don't see any serial number on this.

Steve: It's not micro-etched around the back?

Leo: No.

Steve: How about if you open it up and look in the battery compartment?

Leo: Yeah, but it's supposed to be visible; right? Screw that. [Hammering sounds]

Steve: Oh, oh, oh, oh.

Leo: I don't want anybody - this thing's as tough - oh, yeah, hide that. This thing is tough.

Steve: You know that we know where your phones are, Leo; right?

Leo: Yeah, but I have this on my keys. Yeah, that's already gone. [Hammering sounds]

Steve: Ooh, ooh. Okay, wait. Turn the hammer over and use the horrible nail pull side. Oh.

Leo: Whoa. I don't know where it went. It has gone flying. No, I don't want to carry that around.

Steve: So even after this, you still feel this way.

Leo: Yeah, because Apple's already said they're handing this information over to law enforcement.

Steve: You're carrying a phone around.

Leo: Yeah, I know, but that's a phone. This is attached - I carry this on my person. If I want to murder somebody, I've got to make sure...

Steve: Your phone is on your person.

Leo: Well, I won't carry the phone when I go murder the person. But it's on my keys.

Steve: You're going to bring an AirTag with you when you murder someone?

Leo: Yeah, because it's on my keys. I'm taking it off right now. It could be in my bag because I - but, you know, basically, yeah, I guess you're right. Your phone is doing the same thing, and I have the phone with me all the time.

Steve: And your car. You've got technology out the wazoo, Leo. Your headphones probably are tracking you.

Leo: I'm mad. Well, they didn't tell us, though. That's the other thing. I mean...

Steve: And you've registered your DNA with 23andMe.

Leo: Yeah, I knew that my...

Steve: Your biology is smeared all over the map.

Leo: I knew that my phone was being tracked; right?

Steve: Okay.

Leo: I knew that. They didn't tell me that they have - I had no idea that that...

Steve: Did you read the fine print? I'll bet it's in the fine print.

Leo: Fine print.

Steve: I'll bet it is.

Leo: But I think you're right, most people who carry AirTags don't know that that is broadcasting their exact location to all iPhones in the vicinity. And then we always...

Steve: Wait, wait, broadcasting the tag's location.

Leo: Yeah, and we were told that that was going to be private; right? But the fact that Apple is now, is apparently saving that information about your location and then offering it to law enforcement upon request, that's - I think that's a violation of what they told us.

Steve: Okay. So it's not the AirTag's location. That's encrypted. They don't get that.

Leo: Oh.

Steve: It's the AirTag's ownership. That's all we get.

Leo: Oh, I don't care about that.

Steve: Well, that's all they have.

Leo: Well, what good is that to law enforcement?

Steve: To know who paired the AirTag? That's very useful.

Leo: Oh, I guess if you're using it to stalk somebody. So you're not saying, okay, now, I think my AirTag actually is still okay. You're not...

Steve: Good, because it's...

Leo: You're not saying that...

Steve: First of all, you've just demonstrated how robust the little suckers are.

Leo: I tried to kill it, and I couldn't.

Steve: Wow.

Leo: So you're not saying that the AirTag is sharing my location.

Steve: No, absolutely not.

Leo: Because they were very clear about that, that it didn't do that.

Steve: Absolutely not. It is deeply encrypted, and only the paired - it is encrypted so that only the device which paired the AirTag is able to determine its location.

Leo: Okay. That's what they were - that's what I thought because they were very clear about that.

Steve: Yeah, only the owner of the AirTag is what Apple is able to reveal.

Leo: So when they hand this over to law enforcement, they merely said, oh, that AirTag that somebody found stuck in their tailpipe or whatever, that's owned by Leo. That kind of thing.

Steve: Yes.

Leo: Yeah. Well, that's fine. I don't have a problem with that.

Steve: Oh, good. Then this was all just a misunderstanding.

Leo: It's just a misunderstanding. I'm sorry, little AirTag. I didn't mean to hit you with a hammer.

Steve: Aw. I'll tell you, if that thing had any sort of a G-force sensor, you know, the Apple security people are going to be knocking at your door, saying, "What are you doing to our AirTag?"

Leo: I'm just seeing if Find My still knows where my AirTag is. Uh-oh. It hasn't seen it in four minutes. That's a bad sign.

Steve: Yeah, I think that's when the pounding began.

Leo: I think I might have killed it.

Steve: You know, probably. It might deliberately take that as a hint and just stop, like turn itself off.

Leo: Take the hint. Yeah, because it...

Steve: That might be one of the instructions. Well, you can press the button for 10 minutes, or you can just ask Leo to smash it for you with a hammer. That will disable its tracking function.

Leo: There you have it, ladies and gentlemen. I misunderstood. I feel bad now. You can have your hammer back, Burke. I went searching all over this - this is a good hammer, I might add. Some significant leverage.

Steve: It is AirTag approved.

Leo: I think I killed the AirTag.

Steve: It's AirTag certified. I hope you did.

Leo: Hasn't been heard from in five minutes.

Steve: Yeah.

Leo: I think I killed it. It's all right, I didn't want it anyway.

Steve: If it starts beeping after 16 hours, you'll know that some part of it is still alive.

Leo: Now, that's Apple. Presumably doesn't mean Tile or Chipolo or any of the other Bluetooth trackers are as private. But Apple has always said we're not - that location's not being shared anywhere.

Steve: That's a very good point, and I actually meant to put that in the notes so I wouldn't forget to mention it. That's why I'm using AirTags, not weasel tags. I do not want, you do not want to use a weasel tag on your stuff because weasel will know where you are.

Leo: And all that law enforcement gets is my email, my phone number.

Steve: Yes. That's all they have is the pairing registry data.

Leo: Yeah, yeah. Sorry, AirTag.

Steve: Yay.

Leo: Haven't heard from it in six minutes. I'm thinking it's...

Steve: It's not looking good, Leo. Not looking good.

Leo: It didn't smash it. But apparently the insides were a little munged.

Steve: Oh, I think it's, yeah, it's done.

Leo: You did it again. You scared the pants off of me. In this case not necessarily needed. I've put my pants back on. And thank you, Steve Gibson.

Steve: Well, the world has lost one AirTag. But it has 55 million other ones, busily beeping away.

Leo: I mean, honestly, that really is the thing to underscore is that everything you have is beaconing, as you point out.

Steve: Yes. There is no privacy. It's all an illusion.

Leo: Yeah. And furthermore - by the way, I don't know what's in front of you. Can you fix that, somebody, anybody? Steve is now peering at us through his address card. Hello, Engineering. I sent the whole engineering team off to find your hammer, and now they're gone. Okay. Fixed. And you did mention 23andMe. I just recently swabbed my cheek and sent it off to a company called Nebula. It does my whole gene sequencing, not just sampling.

Steve: Nice.

Leo: Does 100% of Leo's genome. So anybody who's looking for that, you'll find that online.

Steve: And now we know where.

Leo: Nebula.org. They've got the whole thing. Steve Gibson is at GRC.com. That's where you find him. You'll also find copies of this show. He's got two unique versions. Normally we'd have a 64Kb audio version. Steve's got that. But he also has a 16Kb audio version for people who want to shrink it down. What is that, one fifth, something like that, one fourth as big. He also has - that's my AirTag there on the floor. Could you...

PERSON: I'm not touching that thing.

Leo: Okay. We also have - he also has very nice human-written transcripts that Elaine Farris does such a good job with. And there's one more thing he's got, which is important. He's got the show notes. Now, we link to it in our show page at TWiT.tv/sn, so you can click there.

While you're at Steve's site, pick up a copy of SpinRite v6.0, the world's finest mass storage maintenance and recovery utility. As you heard, he's working hard on 6.1. When that comes out, you'll get that for free. But you've got to buy 6.0 now: GRC.com. You can leave Steve DMs at the Twitter. He's @SGgrc. Let me put that on the screen: @SGgrc.

Copyright (c) 2014 by Steve Gibson and Leo Laporte. SOME RIGHTS RESERVED

This work is licensed for the good of the Internet Community under the Creative Commons License v2.5. See the following Web page for details: http://creativecommons.org/licenses/by-nc-sa/2.5/