Security Now! #923 - 05-16-23
Location Tracker Behavior

This week on Security Now!

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?

And even though this is a pure deep dive episode, we do have a timely picture of the week:

What's not to love?

A.I. TURNS THIS SINGLE BULLET POINT INTO A LONG EMAIL I CAN PRETEND I WROTE.
A.I. MAKES A SINGLE BULLET POINT OUT OF THIS LONG EMAIL I CAN PRETEND I READ.
TOM FISH BURNE
© marketoonist.com

SpinRite

I’ve been mute for the past couple of weeks about the ongoing work on SpinRite, not because of any slowdown in the action, but because of a major revamping of SpinRite’s primary core data recovery code. It sometimes happens that software gets itself painted into a corner. Or that 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 seem to be involved. This can occur when an inexperienced programmer trackles a task that they don’t fully understand. 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. At the start of our careers, that was a common occurrence to many of us who love and live to code. It can also happen when a sufficient quantity of new information arrives late in a software design cycle which requires that the code be modified to incorporate the needs created by this new information. Since I’ve been coding for more than 60 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, in fact, fail. Everyone has heard me mentioning for many months that nearly all of SpinRite’s 683 registered development testers have always been bored. Everything has always worked for them. While SpinRite is also valuable to those whose drives are still healthy – it’s proven through several decades to be 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.

I was so close to being able to declare this work finished when a final couple of unexpected surprises landed. For example, although I’ve 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. That’s all fine. But then it 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 an error. And since SpinRite 6.1 obtains its breathtaking speed by transferring 16 megabytes at a time, this 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 that data all settled, then deal with the sector that caused the transfer to halt. That was the straw that finally broke the camel’s back. 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 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 time to immediately get out and try another drive. SpinRite wasn’t ever built to support that sort of emergency user escape.

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 code from inner loops that it felt 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 the entire rewrite Sunday night and it's been a long time since I've been so happy with code. It is 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, 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 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 then 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, 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 could be done.

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

Purely by coincidence, during today's MacBreak Weekly pre-show, Alex Lindsay told the story of the four months he first took to solve a complex problem in 3D graphics, rotating a 3D pyramid on a Commodore 64. Shortly after he finally had it working, the machine crashed and he lost all of his work. Maybe it was the cassette tape that we used back then to save and load programs. In any event, after recovering emotionally from the loss, Alex re-implemented in just three weeks what had taken him four months to do the first time. And it was only after that, that he then realized how it **should** be done, so he rewrote it again, in four days. The way Alex expressed it was as three phases: *"Do it. Then do it better. Then do it right."*

In my present case, when that first little innocuous exception was needed, the quickest answer 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 "solve" that need, 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 true mess. I knew how to solve it, which is what I just did. But right up until I 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. But this time, I really **will** have a beautiful legacy which SpinRite 7 will be able to inherit.

Location Tracker Behavior

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 the 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 2012. 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 told WIRED last year that they had sold 40 million of their popular little devices.

So we know that consumers love the concept and, when tightly integrated with smartphones, they also love the ease of use. A single Apple account may be associated with up to 16 AirTags.

As our listeners 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 can 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 buggers 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 that can be queried by law enforcement when probable cause has been established is what's needed to help curtail the abuse of this quite powerful consumer technology.

I also think it's perfect that Leo and I may have differing feelings about this and that we'll be able to discuss its 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 discover might belong to someone obvious, whom 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 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 absolutely 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 that means for its users and abusers.
https://datatracker.ietf.org/doc/pdf/draft-detecting-unwanted-location-trackers-00

Once I had fully absorbed this detailed, 22-page specification, I was disappointed by what was missing. 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 other details spelled out. I wanted that, because I wanted to be able to present a detailed walkthrough of the technology that Apple has developed to ensure the privacy of the various parties. Instead, what we have is a small subset of that whole and I have been unable to locate anything more complete. This document does provide us with the common behavior that's required from small tracking devices to enforce their owner's privacy and the privacy of anyone they are 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, it appeared that most of this has already been implemented by Apple. So, this felt 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 model, "location" is determined by others in a crowd-sourced model.

Whereas fancy and more expensive traditional GPS-style trackers use both GPS satellite positioning and built-in cellular telephony for reporting to headquarters, these BLE tags rely upon having their periodic broadcasts picked up and their position relayed by other bystanders' smartphones. This means that enlisting all Android and potentially other devices into one big happy crowd-sourced family will significantly improve the tracking 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 she noted that while he was near their sparsely populated home, she didn't get much updating of his position with any of the non-GPS trackers. But when he ventured into "the city" (presumably New York City), 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 AitTag he was carrying and 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.

For those who may not be familiar with RFC-style terminology, I should preface this by noting that the spec uses all-caps qualifiers so that its reader understands how various requirements should be taken. Specifically, the capitalized phrases used are "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL".

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 in the real world. For large accessories, such as a bicycle, these best practices are merely RECOMMENDED. Accessories are considered easily discoverable, and thus recommended but **not** required, if they meet one of the following criteria:

The item is larger than 30 cm in at least one dimension. (That's about one foot.)
The item is larger than 18 cm x 13 cm in two of its dimensions. (that's about 5x7 inches)
The item is larger than 250 cubic cm in three-dimensional space. If it was a cube, that would be 6.3cm on a side or about 2.5 inches on a side.

So, the "detecting unwanted tracking" aspect of this specification is meant to protect from itty-bitty trackers like Apple's AirTags which are discs 1.25" in diameter x 0.31" thick. If it's bigger than a bread box, it doesn't need to rigidly follow this 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 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 sale feature. Wondering whether that had always been done, a quick Google revealed a story from last month titled: "Why a Bike With Built-In 'Find My' Capabilities Is Total Genius" The articles three bullet points said: "Velotric's new Thunder 1 VST e-bike has a built-in AirTag feature. You can track your bike just like you can track your iPhone and your AirPods. 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 an annual 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 anti-theft AirTag tracking so that the lazy thief who wants a free ride will think twice and just walk away instead.

So, there's every indication that for such items, especially when they become cross-platform for compatibility with both iOS and Android, we're going to begin seeing "AirTag Tracking" become a competitive marketing feature.

Okay, moving on, I want to next share some formal definitions from the specification. This is more than just defining terms since a lot of reading between the lines is possible:

Throughout this document, these terms have specific meanings:

- *The term **platform** is used to refer to mobile device hardware and associated operating system. Examples of mobile devices are phones, tablets, laptops, etc.*
- *The term **owner device** is a device that is paired to the accessory and can retrieve the accessory's location.* [So, an "owner device" is typically a smartphone.]
- *The term **non-owner device** refers to a device that may connect to an accessory but is not an owner device of that accessory.* [So, an "non-owner device" is someone else's smartphone.]

- *The term **location-tracking accessory** refers to any accessory that has location-tracking capabilities, including, but not limited to, crowd-sourced location, GPS/GNSS location, WiFi location, cell location, etc., and provides the location information back to the owner of the accessory via the internet, cellular connection, etc.*
- *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 (BT) advertisement payload that is advertised when an accessory has recently, is currently, or will in the future provide location updates to its owner*
- *The term **unwanted tracking (UT)** 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.*
- *The term **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 (LE) range.*
- *The term **platform-compatible method** refers to a method of communication between the platform and the accessory/accessory manufacturers to exchange information, including, but not limited to, BT GATT protocol, BT advertisement, HTTP, etc.*

Also, 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 as “accessories.”

So... Tags can be in one of two major modes: Near-Owner Mode or Separated Mode.

Each Tag periodically emits a broadcast hoping to be heard by some passing smartphone. 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 with which it has previously been paired, the owner’s smartphone notifies the tag that its owner is nearby. This places the tag into its “Near-Owner Mode” which, among other things, suppresses its location broadcasts.

The specification states *“The accessory 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 altitude.

Conversely, the specification states *“The accessory SHALL transition from near-owner mode to separated mode if it has physically separated from the owner device for a duration no longer than 30 minutes.”* Since BLE’s range is limited, it makes sense for a tracking device not to immediately flip into “Separated Mode.” But this says that it must do so within 30 of being away from its owner.

One of the definitions we just noted above was for “location-enabled state.” The specification explains it further, 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 the 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 indicates if the accessory is connected to or nearby one or more of the owner’s devices. Near-owner state can take on two values, either near-owner mode or separated mode. Near-owner mode is denoted as the opposite of separated mode.

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 the separated state. The reasoning behind this recommendation is that unwanted tracking detection relies on the Bluetooth LE 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 (UT) alerts.

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. 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 than adds:

The accessory SHALL broadcast the location-enabled advertisement payload if 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 sort 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 have smartphones and there’s also a tag somewhere that belongs to one of them. If they are all traveling together, the tag is also traveling with the non-owners. So if that tag were to be emitting its location-enabled advertisements, the non-owner’s phones would believe that they were being tracked and would generate false positive alerts. So the proximity of the owner places the tag into “Near-Owner” mode and thus suppresses these overt tracking announcements.

And speaking of these Bluetooth LE advertising announcements, they are quite frequent. The specification’s “Advertising policy” specifies an announcement intervals of between 0.5 and 2 seconds.

Bluetooth LE devices have MAC addresses and therein lies another problem. In this case we’re wanting to prevent the tag from being tracked by an adversary. Here’s what the spec has to say about that. I’ll share it then we’ll discuss it:

The Bluetooth LE 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 Pseudo-Random Function (PRF) 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 purposes 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.

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 for some period of time, when the owner is not in physical proximity.

So this seems well thought out. Devices rotate their 6-byte MAC addresses on a schedule as directed by a secret key. This is analogous to the TOTP one-time passwords we’re 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 upon its periodic broadcasts will be able to determine any tag’s next MAC address.

It’s also worth noting that the devices have no actual native MAC address. Unlike the anti-tracking technology we’ve seen developed for smartphone Wi-Fi, there is not one rotating spoofed MAC used when roaming and another actual physical fixed device MAC address used when associated or paired with homebase. So these tags are all simply using the traditional

6-byte, 48-bit MAC as a short-term rotating ID. So there's a galaxy of them, all occasionally changing their identifier following a predictable pattern that only their owner can predict. Another intriguing aspect of this specification is what non-owned devices are required to do to reveal themselves. The spec calls this "Non-Owner Finding" and it has several components. The spec says:

Once a user has been notified of an unknown accessory traveling with them, it is REQUIRED 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.

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 be configured to detect a change in orientation of ±10° along any two axes of the accessory.*

After some number of hours between 8 and 24 chosen randomly from a uniform distribution of 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 any motion within 10 seconds. [So it's sampling its position at that interval, presumably to conserve power.] *If motion is not detected within the 10 second period the accessory MUST stay in this state until it exits separated state.*

If motion is detected within the 10 seconds between samples, the accessory MUST play a sound. After any first motion is detected, the movement detection period is decreased from 10 seconds to half a second. 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 conditions: Motion has been detected for 20 seconds at the half a second sampling rate, or 10 sounds have been played.

If the accessory is still in its separated state at the end of the 6-hour backoff, the unwanted tracking behavior MUST restart. Any Bluetooth LE connection from a paired device MUST reset the separated behavior and transition the accessory to connected state.

The accessory MUST include a sound maker (for example, a speaker) 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 5 seconds each time.*
[The specification goes into loudness measuring details.]

So this gives us a system where, after a tracking device has been separated from its owner for an uncertain interval, randomly and uniformly chosen between 8 and 24 hours, its motion detector will activate. At that point it will begin taking readings of its angular orientation in three axes every ten seconds and if it finds that it has 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, this will have effectively 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.
All of this occurs whenever a tag is separated from its owner for between 8 and 24 hours. It's

about requiring anything that can track to deliberately reveal itself periodically... and 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 AitTag 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 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 only making and receiving calls, texting and taking pictures. I don't know whether this person was ever actually 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.

Anyway, there is 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 else stealthily would choose. After that initial period of 8 to 24 hours of separation from all of its owner's devices has lapsed, 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 for more.

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 range of a tag, whether a tag's owner or non-owner, is also able to remotely cause a 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 tag to sound off to aid its location.

Tags also contain a wealth of queryable information. This includes:

- A unique 8-byte 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 issuer can be determined.
- The manufacturer's name, up to 64 bytes.
- The model's name, also up to 64 bytes.
- An 8-byte "Accessory category" indicator. Only the first of the 8 bytes is presently defined and many values for that byte are already defined. They include things like "Finder", which has a value of '1'. And also Other, 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.

  I wanted to run through those because it 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 many higher-end consumer products – like that eBike – which are prone to misplacement, loss or theft, or which might need to be located with urgency, such as an inhaler, may eventually incorporate this consumer location technology as a sales feature.

- The last item of queryable data is a 4-byte, 32-bit value which enumerates a tag's capabilities. Only four bits are currently defined. Those are:
  - Bit 0 - Supports play sound
  - Bit 1 - Supports motion detector
  - Bit 2 - Supports serial number lookup by NFC, and
  - Bit 3 - Supports serial number lookup by BLE.

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 (e.g., button press, gesture, removal of battery, etc.).

The accessory manufacturer SHALL provide both a text description of how to disable the accessory as well as a visual depiction (e.g. image, diagram, animation, etc.) that MUST be available when the platform 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 to reference. (Again, remember that "platforms" in this spec refers to smartphones, pads and similar devices with full user interfaces.) This URL must return a response which can be rendered by an HTML view.

So this is 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. But this will always require 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, this spec makes some clear distinctions about what can be done via NFC that cannot be done over Bluetooth.

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. 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 (e.g. image, diagram, animation, etc.) 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 look-up over Bluetooth LE SHALL be available for platforms to reference. This URL MUST return a response which can be rendered by an 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 or to learn its immutable serial number.

It is obviously required for the serial number since, unlike its MAC address which is nothing but a keyed pseudo random 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.

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 to query it. But we also just encountered the phrase *"decrypt the serial number payload."* 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 5 minutes upon user action on the accessory (for example, press and hold a button for 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 value 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" mode, and either presses a button on the tag or taps it for a Near Field exchange, that smartphone receives a URL 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 possession of a tag may use the tag to obtain its serial number electronically.

But remember that the serial MUST also simply be physically printed on the device. This suggests that this electronic fallback is provided in case a would-be malicious tracker thinks they're get clever by sanding off the device's exposed and obvious serial number to make it anonymous.

Since the serial number is an important piece of the whole, the spec further explains:

> *Serial number look-up 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.* [We'll get to that in a second, this spec has a lot of out-of-sequence content.] *Therefore, the serial number SHOULD NOT be made directly available to any requesting devices. Instead, several security- and privacy preserving steps SHOULD be employed.*
>
> *The serial number look-up SHALL only be available in separated mode for a 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 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 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 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 the decryption, and for the service to know which decryption key and protocol version to use.*

So possession of the tag, which would usually allow someone to simply look at it 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 with embedded decryption data that such a device can query to then obtain the device's plaintext serial number. This is what they meant by rate limiting and anti-fraud measures to prevent attacks. By asking a remote service for this information, its disclosure can be controlled.

And now we get to the so-called "pairing registry" where we ended our truncated discussion last week. To reiterate against the background of 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, e.g., phone number, email address.

Okay. So the process of beginning to use a new tracking tag is to pair it with the user's so-called "platform" device – typically a phone or a tablet, etc. And most users will, in turn, have their real-world identity and billing account 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 of "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 display so, too, is some deliberately obfuscated owner information that's obtained from this pairing registry. The spec says:

A limited amount of obfuscated owner information from the pairing registry SHALL be made available to the platform [meaning any 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. e.g., (***) ***-5555, or
- An email address with the first letter of the username and the domain name visible, as well as the entire top level domain . e.g., b********@i*****.com

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 separated from those accessories.

So, the idea being that someone might immediately react with *"Who the 'F' is tracking me?"* And this allows that individual to immediately obtain some information about the account and individual who is currently paired with a device while it's in its "separated" mode. It may often be someone who the person knows 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 of a few characters from their entire eMail address.
So who has access to the fully unobfuscated pairing registry?
Law enforcement. Paragraph 3.15.3, titled "Availability for law enforcement" 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 heavily obfuscated identity information. So one purpose for the pairing registry is to aid in immediately resolving any *"Who the 'F' is tracking me?"* style questions. 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 this 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 world wide standard. In any crowd-source model, bringing Android into the fold would make the system far more powerful for everyone.

So I did some further digging 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:

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.

Since that was a bit noncommittal, I dug a bit more. On February 22nd of **last** year, so 15 months ago, in their article titled *"An update on AirTag and unwanted tracking"* they make this VERY plain and explicit: https://www.apple.com/newsroom/2022/02/an-update-on-airtag-and-unwanted-tracking/

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.

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.

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 a surprise. But no one should have been surprised. It's not new.

This, of course, begs 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. 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 of the evidence we've seen during the 17+ 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 my bicycle or my luggage, I wouldn't hesitate 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 thought about AirTags before – even though we've covered them on this podcast whenever they made the news. But knowing now what I do, I am absolutely going to purchase some AirTags and never travel again without them in my luggage.

We know that they won't begin squawking until they've been separated from us for a minimum of 8 hours, or on average 16 hours. So, if they're in the belly of a plane 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 for $29. And I think that, being a bit sneaky, I would have one AirTag exposed and visible hanging out on an AirTag dongle with another 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 what we wind up with is a single global standard which, due to the crowd-sourced positioning nature of the technology, will make it much more powerful and attractive.

And next week... back to the news, which will start off with anything that happened during this past week.