WiFi 6

Description: We begin this week as we often do on the third Tuesday with a look at the previous week's Patch Tuesday; and, in this case, a troubling new trend is emerging. We look at the DoH support coming soon to Windows 10, and at a little known packet capture utility that was quietly added to Windows 10 with the October 2018 feature update. We'll spend a bit of time on yesterday's DOJ/FBI press conference, and then take a look at a problem that Microsoft appears to be having a surprising time resolving. We'll take a look at face masks thwarting automated public facial recognition, and Utah's decision to roll their own contact tracing and locating app. And we'll wind up with what I hope will be an interesting walk through the history of Ethernet, from the beginning of wired to the evolution of the many confusing wireless protocols.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-767.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-767-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. We'll look at last week's Patch Tuesday. Believe it or not, the third highest number of vulnerabilities last Tuesday, only topped by March and April. What's going on with Windows? Steve will talk about it. And then a deep dive into the technology of WiFi 6 and why it might be worth waiting for WiFi 6E. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 767, recorded Tuesday, May 19th, 2020: WiFi 6.

It's time for Security Now! with Steve Gibson. He's over there. Now he's there. He's there. Actually...

Steve Gibson: I'm over here, Leo.

Leo: What people don't know is, they see you there, but I see you there, there, there, there, there, there. I have seven or eight Steves around me at all times, many screens.

Steve: That's more than you need.

Leo: Plenty of Steve. Hi, Steve, good to see you.

Steve: Good to be back.

Leo: Episode 767.

Steve: Yup.

Leo: Flying into the future.

Steve: Yup. And we are, as we will see, a few days, I think three days, shy of a major anniversary.

Leo: Oh.

Steve: The 47th anniversary of the invention of Ethernet.

Leo: Oh, Bob Metcalfe.

Steve: Yes. I want to talk about WiFi 6, not for any particular reason except that there was no, like, outstanding news of the week. And I just sort of thought, okay, I've had it on my list of things to kind of get to because this whole 802.11a/b/g/n/ac, all of this is - oh, my god. Yeah, it's just a mess. And so I thought, let's sort of take a walk through history, looking at where this started, and look at the major milestones as we've gone along to look at the evolution of this, wrapping up with, well, some interesting recommendation. Because just last month the FCC approved another chunk, actually it's a 1200 MHz bandwidth chunk up at the 6 GHz band, separate from 2.4 and 5, which is what WiFi now uses, which the industry's going to immediately jump on because it represents "next."

And so the question is, if you were about to go to 6, what are the things to consider about that? And maybe it makes sense to wait till you get a base station, an access point that is able to take advantage of the next stuff. So anyway, we're going to finish with that.

But we've got, as we often do on the third Tuesday of every month, take a look at the previous week's Patch Tuesday and, in this case, maybe a worrying new trend which is emerging. We'll also take a look at the DoH support coming soon to a Windows 10 near you. That is, natively, in the OS itself. So not just the browser, but everything. Also, a little-known packet capture utility that was quietly slipped into Windows 10 back with the October 2018 feature update, which I will talk about how you can use it to verify what type of DNS your Windows 10 system is using after you configure it to use DoH, see if it actually is.

We're going to spend a bit of time on yesterday's DOJ/FBI press conference, where they're moaning more about Apple, and also take a look at a problem that Microsoft appears to be having a surprising time resolving. And as I was researching this, I kind of got this, well, I wouldn't put it as, I don't know, it wasn't a sinking feeling. But it's like, okay, this doesn't seem to be Microsoft up to their usual game. Also we've got - I got a kick out of this - face masks are thwarting automated public facial recognition in the U.K.

Leo: Good.

Steve: Oh, boohoo, uh-huh. And Utah's decision to roll their own contact tracing and locating app. We're going to take a look at it. And why I end up thinking, you know, if it were available in California, I think I would probably step up and do it, yeah.

Leo: Oh, really? Good, good.

Steve: And then we'll wind up talking about - I have a little brief update on SpinRite, of course, because I'm working on that full-time now. And then we're going to start from Bob Metcalfe on.

Leo: Wow. From Bob - you're going to skip Token Ring, I hope.

Steve: Yes. We're not doing Token Ring.

Leo: Actually, relevant to that Utah story, any minute now, maybe not today, but I think it'll be today, we'll get the next generation of iOS that will include that new API for that kind of contact tracing. So that should be kind of interesting.

Steve: Oh, and Leo, I forgot to mention we have the Painful Pun Picture of the Week for our listeners.

Leo: I'm looking at it right now.

Steve: Actually for our viewers. But anyway, our listeners will get it, too. So this was photoshopped, but it was pretty clever.

Leo: See if people get the pun. See if they get the pun.

Steve: It shows a guy wearing a white mask. And of course on the front of it - well, not "of course," but in this particular mask the front of it has stenciled "255.255.255.0." So of course that makes it a subnet mask.

Leo: Oh.

Steve: Oh.

Leo: And a very restrictive one, I might add.

Steve: Okay. Well, a Class C subnet mask, yes. You really probably did, when you think about it, you want a Class A subnet mask.

Leo: Oh, yeah. That's what we're going for.

Steve: If you're going to be trusting your health to it. So of course this guy got the best he could, I guess.

Leo: Yes. This is just IPv4, I mean, no big deal.

Steve: Yeah, exactly, yeah. So last Tuesday's patching round was not the biggest ever. But it was the third largest in Microsoft's history.

Leo: What?

Steve: Weighing in with a whopping 111 CVE tracked bug fixes.

Leo: They did 113 last month. This is crazy.

Steve: Yes, Leo, and 115 the month before.

Leo: Wow.

Steve: Sixteen of this month's 111 were rated critical, and all but one enabled remote code execution by an attacker. In a refreshing change of pace, however, none were zero-day flaws. We've been having those recently. Not this month. But think about this. If things have been seeming worse recently, and they have been, it turns out it's not our imagination. Because as you noted, 115 bugs were fixed in March, making it the number one largest number of bugs in Microsoft's history, and 113 a month after in April, that is, last month, with 111 this month.

Leo: Crikey.

Steve: So the past three months - March, April, and May - were the first, second, and third most patched bugs in Microsoft's entire history. And although this month's bugs spanned 12 different Microsoft products, from Edge to Windows and Visual Studio to .NET, nearly half were problems within Windows itself.

Leo: So it's a big code base. How big are these bugs? I mean, are they just little errors or...

Steve: Well, and that's really - that's a perfect question. So I'm glad that these oversights were being fixed. And I'm glad that whatever it is that is wrong doesn't appear to be affecting my own work when I'm using Windows 10 because I sit in front of Windows 10 every evening. That's what I have in my location with Lorrie. So perhaps

these are all just exploitable edge cases. And since I was wondering exactly that, I took the time to read each and every one of this month's 111 descriptions.

Leo: Oh, god. Oh, my god.

Steve: Yes, dear podcast listener, I did that for you.

Leo: How painful.

Steve: I made a note of what it was that was wrong and was fixed. Every one of the 111 problems was exactly one of the following: remote code execution vulnerability.

Leo: Not good.

Steve: Denial of service vulnerability.

Leo: Not good.

Steve: Elevation of privilege vulnerability.

Leo: Very not good.

Steve: Cross-site scripting vulnerability.

Leo: Terrible.

Steve: Memory corruption vulnerability.

Leo: Oh, my god. They're all bad.

Steve: A spoofing vulnerability.

Leo: Oh.

Steve: Or an information disclosure vulnerability.

Leo: They're all terrible.

Steve: Well, they're all bad. But among those 111 problems there was not a single one that was not a vulnerability. So this suggests that it's not that Windows 10 is not working. I mean, there are problems, like we've seen with upgrades having to be backed out of, and things going wrong because of some screwy AV system they've got or something. As we know, for the most part, it works. And I'm sure that most of us listening to this podcast, although we had a bunch, you know, thinking about the crowd that we had in Boston, Leo, sort of a representative demographic of the podcast, there was a range of ages. There were young bucks and old-timers.

But those old-timers among us certainly recall those days using the early versions of Windows when it would just lock up at any time, without any apparent cause, and always without warning. Some of the losses that resulted were so traumatic that to this day, today, I'm not kidding you, decades later, I'm still hitting CTRL+S for Save...

Leo: Yes, that's right, yes.

Steve: ...before I ever switch away from anything I'm doing.

Leo: Oh, yeah. Habitually. Every minute.

Steve: I'm still not willing to trust that I'm going to be able to come back.

Leo: But you're right. Those things don't happen as often. The hard crashes, the data loss.

Steve: I would say I spend most of my time in front of Windows 7. What I'm noticing is that the video driver crashes. But it's like everything freezes for a minute, and I go, okay, what's this? And then everything goes blank. And then it kind of comes back. Well, it couldn't used to do that. I mean, if the video driver crashed, it was blue screen. Now it's like, ooh, ow, hold on a second. And then it, like reconstitutes itself.

Leo: Well, they rearchitected with NT, and they really do protect Ring 0 much better than they used to. It's a lot harder to bring the system down.

Steve: The one thing we still have today, I would argue that my incessant CTRL+S'ing is probably, from a rational standpoint, it's really not that necessary. But CTRL+C, the thing we have now is that that doesn't always take. And so I noticed that, if I want to make sure that I copy something to the clipboard, now I'm hitting CTRL+C a bunch of times, just because one doesn't seem to be enough to give Windows the idea that, oh, no, he's serious. He really does want this on the clipboard. It's like, okay, fine. But anyway, in the case of the myriad vulnerabilities that we've seen, certainly in this past month, we have - I guess I'm feeling like Windows 10 works. But you really, as you were saying, noting the nature of these vulnerabilities, you really don't want to expose it to too much incoming because that's not going to have a happy ending.

Leo: Do you think it's because there are more security people trying to find these bugs than ever before? That must be some of it.

Steve: I do think so. And Leo, they just - they, Microsoft, will not leave it the eff alone.

Leo: Well, that's true, too.

Steve: Just stop effing with Windows. Let it settle. Let it stabilize.

Leo: But they're always adding stuff. They can't leave it alone. And that also speaks to the poor quality of the code base because everything's interdependent. So you change something here, and suddenly your wallpaper can't resize or something bizarre; right?

Steve: Exactly. Something, some obscure, like, okay, I can't set dark blue anymore. What?

Leo: Yeah. That stuff should not be interdependent. It really shouldn't. But, you know.

Steve: I did have a really, really good friend who got his comp sci degree from MIT. He was at Berkeley for a while. And he went to Microsoft. And as he was leaving, he said, "Oh, this is gonna blow. It's gonna blow. Run a bypass, quick."

Leo: Oh, there's patch upon patch upon patch upon patch.

Steve: Oh, boy. Yeah. So get somebody to filter your email externally, if possible. Use a well-curated web browser. If you're going to go to anywhere sketchy on the Internet, do that in a VM. VMs are easy now. They're free. We've been talking about various VM solutions. Pass anything that you download while you're in the VM through VirusTotal before you even consider moving it out of the VM. And then revert any changes that you made to the VM while you were using it.

Leo: Sure. Uncle Joe's going do that.

Steve: That's not completely necessary. But in the past three months, just the past 90 days, 339 vulnerabilities...

Leo: Incredible...

Steve: ...have been patched. And a number of them were being exploited at the time that they got fixed. So, as I said, it works. But you just don't want to subject it to too much incoming because, yeah, not good.

Leo: I mean, do you think if you wrote something from scratch - it's just such a hairball of a thing.

Steve: Oh, Leo. There is no one on Earth who now understands it. And that's a problem.

Leo: That's a problem, yeah.

Steve: At least we have Linus, who still is sitting on top of and incubating Linux, the Linux kernel. But there's nobody. This thing is out of anyone's ability to comprehend. And of course that's part of the problem. We were talking recently about how there is a problem with, when we have a big project, you're inherently needing to rely on other people's code. You're grabbing libraries for this and that, Node.js here and some other module there, in order to glue together a solution.

So what that means is that you don't understand. You didn't write all that. You don't understand it all. You're assuming, hoping that these various pieces each behave themselves. That's Windows now because just once upon a time - we talked about this, I still shed a tear - when I actually knew what my files were. We have computers now, we just sort of scroll through the endless System32 and the "x by x," whatever that is that's now 25 gigs of something. And it's just like, okay, I just give up. I don't know what any of this is. Just please don't have it hurt me.

Leo: I think some of this is a legacy of - there was a period of about 20 years where we were infatuated with some - there were some very bad fads in coding. Like object-oriented coding. And I think we were infatuated with this stuff. And I think C++, I think some of this is the legacy of some just bad technologies that are going away now, frankly.

Steve: And frankly, I mean, to Microsoft's credit, legacy is their middle name.

Leo: Right. You can't dump it.

Steve: Once upon a time they were getting a graphical user interface to run in a system with 640K.

Leo: Right, exactly, yeah.

Steve: And so that's where the DLL was born. That crime against civilization came from the fact that you had no choice back then to share code. So the idea was that you'd have these blocks of code that apps would - basically an app was just a script that was calling functions in external dynamically linked libraries. And there was only one instance of this DLL in RAM so that it wasn't taking up space. To their credit, they got an amazing system to function in a zero resource environment. The problem is then, like, they couldn't leave, like, okay. That was version 1 of the DLL. Oh, but we're going to add a few more things and make it version 2. But then of course it broke compatibility with version 1. So then remember how you would install some new program, and things that were already installed stopped working? So that happened...

Leo: All the time.

Steve: ...because of DLLs colliding with each other.

Leo: All the time. I got calls so often on the radio show, and the answer would be, oh, you installed the DVD burning software. That DLL clobbered the DLL that your word processor was using. And so that's why you can no longer get email. And it was like, oh, so frustrating.

Steve: At least we left IRQs behind.

Leo: Yeah, I know.

Steve: I mean, half of your career, Leo, was IRQ conflicts.

Leo: No kidding, it really was. So depressing. I think it's getting better. I really do.

Steve: Well, no. I mean, I totally agree with you. I never, I mean, I'm not on the bleeding edge of anything, but my systems never lock up or crash. They just go. Sometimes they get wounded, but they seem to heal themselves. It's like, oh, okay. That was good. That's a little bit of magic. So, yeah, I just...

Leo: It's amazing.

Steve: Yeah. They are huge lumbering machines.

Leo: They are.

Steve: So, okay. Speaking of Windows 10, here I'm complaining that Microsoft doesn't leave it alone, and then the next thing I talk about, this happened now several podcasts, it's like, ooh, listen to this new feature that Windows 10 is getting.

Leo: You see? You see? You see?

Steve: Okay. So DoH...

Leo: They do it for you, Steve.

Steve: ...which is to say DNS over HTTPS test drive is appearing for Windows Insiders.

Leo: That's really interesting. That's surprising.

Steve: Yeah. I know. And they really jumped on this quickly. Without indicating when they would be giving this a wider release, Microsoft is letting Windows Insiders test drive DNS over HTTPS in the Insider Preview Build 19628. So you need to be on the Fast Ring. You need to be all on the jiffy spiffy quick, last night's build, hold your breath and boot. But the key number is 19628 and subsequent.

Last Wednesday Microsoft wrote: "If you've been waiting to try DNS over HTTPS (DoH) on Windows 10, you're in luck." Luck was actually not involved, but fine. "The first testable version is now available," they wrote last Wednesday, "to Windows Insiders. If you haven't been waiting for it and are wondering what DoH is all about, then be aware this feature will change how your device connects to the Internet and is in an early testing stage. So only proceed if you're sure you're ready." On the other hand, it doesn't do anything to you. You've got to go ask for it.

So Microsoft first indicated their intent and their interest when we talked about this last November. And as opposed to our browsers that are only offering this for browsing, this is at the OS level. So all browsers, without any configuration, everything else you do, email, any other utilities that are doing DNS lookup, they all get protected. So this will be native for all of Windows' DNS queries. And where DNS is of course ubiquitously available from every ISP over the UDP protocol to port 53, DoH is not yet universal.

So the first question anyone has is what DoH provider does Microsoft use? Now, I went digging because I was curious about this. Turns out there is a proposal in the works for adding a DoH type to DHCP. So just as our systems today are able to auto configure to our providers' old school traditional DNS, when a DoH type becomes supported by ISPs and routers and PCs, then the same kind of auto-configuration can occur when an ISP chooses to support DoH with their own server. But until that time, it's up to the user to decide which external DoH provider to use.

And in Microsoft's case, three providers are currently supported to be used as DoH resolvers: Cloudflare, Google, and Quad9. The way this works is kind of clever. Windows needs to be configured to use one of these as a DNS server, sort of through the normal DNS settings. DoH is disabled by default in this preview build. You need to do some, believe it or not, registry tweaking. This doesn't surface at the UI at all. I've got the specifics in the show notes.

You go to, for those propeller heads among us, HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services. Under that is Dnscache, and under there is Parameters. So in the Parameters key under the Dnscache key under Services, you create a new DWORD named "EnableAutoDoh" and set its value to 2 for some reason. I don't know what 1 does, but 2 is what we're told to do. This enables DoH.

And so as long as you have your Windows machine configured to use Cloudflare's normal DNS or Google's normal DNS or Quad9's normal DNS, when Windows is restarted, rather than sending normal DNS to one of those three providers, it sends DoH DNS, DNS over HTTPS. So it's kind of clever. I liked how, at the UI level, you just say, okay, I want to use Cloudflare's DNS, Google's DNS, Quad9's DNS. And then if you also have it set to EnableAutoDoh, when Windows realizes that it's about to make a DNS query to one of those three, it goes, oh, and instead establishes - and probably does it persistently because you want that, in order to be processing many DoH lookups per second - it establishes an HTTPS connection to that provider, and then that's its tunnel through which the entire OS then resolves its DNS queries.

Oh, and if you want to set up a DoH server that isn't already on what Microsoft calls their so-called "auto-promotion list," meaning that those are the DNS queries to an IP that are auto-promoted to DoH because for example maybe your corporate Intranet supports it, although there's really arguably not a big need for DoH over a corporate Intranet

because it's really only when it goes outside of your control; or if your ISP started to support DoH and you wanted to use it, just because sort of in the spirit of not aggregating all DNS down to a very few providers, you can do that, there is a command line.

Those who have messed with a command line are familiar with netsh. So you say netsh - and again, in the show notes, the details - space dns space add space encryption space server= and then the IP address that you want Windows to notice and then convert from DNS to DoH, and then the so-called "dohtemplate." So you do IP address space dohtemplate= and then that server's URL template. They call it the DoH-URI-template, and that's all of the DoH servers have it. It's https:\\, you know, who knows what it is, like doh.cloudflare.com or whatever. I just made that up. That's not what it actually is for Cloudflare. And so that's the HTTPS URL to which a persistent connection is made to then establish the tunnel. And you can type something similar in order to query whether the IP address got registered: netsh dns show encryption server= and then the IP address you're querying, and it'll let you know if it has registered that IP in its auto-promotion list.

So anyway, I'm delighted that Windows jumped on this as quickly as they did, or that Microsoft jumped on this for Windows 10 as quickly as they did. It just makes it a no-brainer. You don't have to worry about, I mean, I guess our browsers are going to begin doing it, so they'll be establishing their own tunnels, ignoring what Windows is doing. But at least then that means that the rest of what Windows is doing will also be DoH to the provider of your choice, based on how you've configured Windows; or even to your own local ISP at such point as they start supporting it.

So how do you know it's working? The standard answer would be to use some sort of network sniffer. Microsoft has a network monitor that you can download separately, or of course the perennial favorite is Wireshark, which continues to evolve and move forward. But an intriguing option is to use Windows 10 little-known built-in and completely undocumented packet monitor which is pktmon. If you're in front of a Windows 10 machine, open a command line or any form of command prompt and type "pktmon" ENTER, and you will be treated with a little help screen showing you the list of available verbs to follow pktmon with. And you can add help to the end of any of those and get help on those, and basically explore the command tree of a built-in packet monitor.

It was quietly added with zero fanfare, as I noted back in the October 2018 Windows 10 feature update. And with this month's forthcoming Windows 10 2004 - boy, is that a confusing number - feature update, it will be gaining a little bit more capability. Right now it can run counters, which you can then examine. Or it can log packets to a log file, which can have its format converted. With Windows 10 2004 feature update, you'll be able to have it dump its captures directly to the screen, which will be very cool. It'll just give you a very quick real-time display of stuff happening.

So, for example, if you were to start - say that you wanted to find out whether in fact your system was still issuing any old-school UDP. You might start by saying "pktmon filter remove" to clear any old filters that you might have around. Then you'd say "pktmon filter add -t UDP," saying that I want to capture the UDP transport, and then " -p 53" for port 53, which we know is DNS. So that would add a filter to capture all traditional DNS over UDP queries. Then you say "pktmon start," which will initiate the filter and begin running counters, which will be incremented for any packets that match the filters. If you wanted to, you could do "pktmon start --etw," which will tell it to start the filtering and capture packets into a pktmon.etw file. And there are other parameters for all these things. So you can, like, give it a different filename if you wanted.

So what I did last night when I was playing with this was that I then had my browser - I poked around. I went to an NY Times page thinking, okay, that's going to be full of DNS

queries. It's going to be pulling crap from 200 different domains. And then I came back to the command window and said "pktmon stop." And I got a dump - I have it in the show notes for anyone who's curious - a dump of all of the capture events which occurred for UDP port 53 during the period of time that things were running. So it's just very cool. It shows you all of the different stages of drivers in your various stack. You can type "pktmon comp list" to list all the registered components in your system's network driver stack, which is amazing. There's just a bunch of stuff there.

But anyway, it's a terrific little built-in tool. You don't need to add anything else to your Windows 10 system. Everybody's got it since the October 2018 update. And it can quickly let you know, just for any kind of purpose, what your network is doing. If it captures your imagination, BleepingComputer's Lawrence Abrams has reverse engineered a few additional tips and tricks. As I said, there's no documentation anywhere. Microsoft did refer to it in their DoH article, which is probably what put both Lawrence and me onto it. Lawrence took the time to go down every command path. And anyway, he did find out a bunch of additional things. So anyway, just a little hidden gem in Windows 10. Very cool built-in utility. Because, again, you could install Wireshark and go that route. But this is already there.

So the DOJ and the FBI during a press conference yesterday criticized Apple yet again over encryption. And boy, this was sort of - it was a weird comment that I'll highlight here. I just didn't want to, I mean, we know what it's about. I just didn't want to fail to touch on the fact that William Barr, who of course is the current head of the U.S. Department of Justice, announced - and I thought this was interesting - that FBI technicians had finally, after four months and "much expenditure of taxpayer dollars," managed to crack and gain access to the two locked iPhones belonging to last December 6th Pensacola naval air base shooter Mohammed Saeed Alshamrani.

During the press conference, FBI's director Christopher Wray criticized Apple for not helping its investigators to unlock the two phones. Wray said the entire process of cracking the terrorist's two iPhones took four months and "large sums of taxpayer dollars." So they were both harping on this.

Leo: Where did that money go, do you think?

Steve: Exactly, Leo. Since actual cracking itself doesn't cost anything more than the time and skill, either the FBI techs were extremely well paid during this arduous past four months, or more likely the FBI went outside to purchase the golden keys for those two iPhones. And remember that one of the two phones had been shot by Alshamrani. I was amazed when we first talked about this that they were able to get it going again. It's like...

Leo: He must have missed.

Steve: You could shoot an iPhone, and it still lights up? That's amazing.

Leo: He missed the memory.

Steve: In any event, the DOJ said that, following the FBI's success, they were able to link Alshamrani to an Al Qaeda branch active in the Arabian Peninsula, which I guess is no surprise because those were his people. William Barr said: "We now have a clearer

understanding of Alshamrani's associations and activities in the years, months, and days leading up to the attack." And I should clarify that this guy was sort of like on base for some training here in the U.S.

Leo: He was a Saudi national, right, to be trained, yeah.

Steve: Yes. Yes, yes, exactly. Anyway, however, FBI director Wray said that the investigation could have advanced sooner if Apple had helped the FBI's technicians. Wray said that despite public pleas from both President Trump and Attorney General Barr, Apple did not cooperate in the investigation. But then, in an immediate contradiction, Christopher Wray continued: "Apple made a business and marketing decision to design its phones in such a way that only the user can unlock the contents, no matter the circumstances."

Leo: Shocking. Shocking.

Steve: "Apple's desire to provide privacy for its customers is understandable, but not at all costs." So first he's complaining that Apple refused to cooperate with this specific case, despite pleas from top administration officials. And with the next breath he's complaining that Apple chose to design their phones in such a way that it was impossible for them to comply. So I think we need to read this nonsense as the continuing drumbeat toward the inevitable collision of consumer encryption technology and legislation, which will somehow criminalize the use - or the sale, probably, because I doubt you're going to blame the user - of subpoena-proof encryption.

We might imagine that this could take the form, for example, of some spiffy penalty legislation where, for example, any manufacturer of a consumer electronics device produced for sale within the United States will have 30 days to comply with a lawfully issued subpoena to decrypt and provide all data contained within the device to law enforcement. After which a significant fine, who knows, perhaps some percentage of that company's annual revenue, so that it sort of auto scales to the size of the company, would be levied against the company for each additional day beyond the initial 30 days that the device's decrypted contents are not provided to law enforcement. I'm just making that up. And of course that would be for devices sold, beginning to be sold after some certain date so that companies had some opportunity to change the way they have decided to have their devices operate.

I did like the EFF who, as always, weighed in, noting that the very fact that the FBI was able to crack the iPhones to obtain all of its information argued against the need for any change to the existing status quo. So thank you, EFF, for always being there.

Leo: Oh, boy.

Steve: So anyway, again, I mean, they just, like, they wanted to announced their victory and to also pound Apple over the head and also drive another stake into the whole encryption debate.

So when is a fix not a fix? All too often, Leo, it's when path traversal attacks are involved. Yes, we were just talking about this, last week or a couple weeks ago, you and I, about the original sin of hierarchical directory design.

Leo: Oh, yeah, ..\.., yeah.

Steve: With its, yes, exactly, its inherent "dot dot" to refer to the parent directory, which allows you to traverse back up the hierarchy and then back down. And of course that's not itself as much of a problem, well, except that then security constraints were placed on hierarchy, cleverly, but still ouch, as it turns out, on nodes in the hierarchy such that descendents of a security policy that exists at a branch in the branching directory hierarchy are then inherited by all of the objects downstream of the branch. So it's very powerful. You could argue it's very cool. And Windows inherited the same architecture, I mean, because it is very powerful. But oh, my god, has it been a source of security problems. And as the saying goes, what could possibly go wrong?

In this week's installment of what did actually go wrong, we learn that when Microsoft fixed their reverse RDP (Remote Desktop Protocol) attack problem last July, and then tried to do it again this past February, it was never really fixed, at least not for everyone. So backing up a bit, recall that last summer several attacks against Microsoft's RDP protocol came to light. There was a serious authentication bypass against the server, which led to last summer's spate of attacks against vulnerable RDP servers. And also there was, separately, a path traversal vulnerability that could compromise an RDP client which made the mistake of remotely connecting to a malicious RDP server. And I noted at the time that that one seemed significantly less worrisome since someone using RDP to access a server other than their own seemed less likely.

So this all came to light after Microsoft first patched the vulnerability as part of its July 2019 Patch Tuesday update last summer. But later it turned out that Microsoft only thought that they had resolved the problem. Researchers at Check Point were able to bypass that patch simply by replacing the backward slashes in the path with forward slashes. The researchers explained that the July patch can be bypassed because of a problem that lies in its path - and here's a word, well, I'm blanking on the word because it's canonicalization.

Leo: Yes.

Steve: A mouthful.

Leo: Canonicalization. That's it.

Steve: The path cannot - I can't say it. Canonicalization.

Leo: There you go.

Steve: Path canonicalization...

Leo: There should be a better word than that. That's terrible.

Steve: Well, people like them. Yes, well, so it's path - the actual API call is PathCchCanonicalize, which is used to sanitize file paths. This allows, that is, the failure for them to properly canonicalize paths...

Leo: Very well done.

Steve: ...allows a malicious attacker to exploit the clipboard synchronization which exists between an RDP client and its server to allow the server to drop arbitrary files in arbitrary paths on the client machine. Thus the clipboard redirection feature, while connected to a malicious compromised RDP server, allows the server to use the shared RDP clipboard to send files to the client's computer to achieve remote code execution of any code that the server wants to shove down the client connection. And although the Check Point researchers had originally confirmed, and quoting them: "The fix matches our initial expectations," back when it appeared in July, it appears that didn't actually fix the problem. The patch can still be bypassed, as I mentioned, by replacing the backward slashes, you know, file\to\location, in any paths with forward slashes, just turning them into leaning right slashes, which are, as we know, the path separators used in Unix-based systems.

The researchers explained: "It seems that PathCchCanonicalize, the function that is mentioned in Windows' best practice guide on how to canonicalize a hostile path, simply ignored the forward-slash characters." They said: "We verified" - I know, Leo. It's just amazing. "We verified this behavior by reverse engineering Microsoft's implementation of the function, seeing that it splits the path into parts by searching only for '\' and ignoring '/.'"

After this was pointed out, Microsoft acknowledged the incomplete fix and repatched the flaw three months ago in its February 2020 Patch Tuesday update. But in the latest chapter of this ongoing saga, a Check Point researcher observed that Microsoft addressed the issue by adding a separate workaround in Windows while not actually repairing the underlying cause of the bypass issue in the PathCchCanonicalize function. The upshot of this is that, while the workaround apparently works fine for the built-in RDP client in Windows operating systems, the patch will not protect and does not protect any third-party RDP clients - and there are many - against the same attack that relies upon the vulnerable Path Canonicalize sanitization function developed by Microsoft.

Check Point wrote: "We found that, not only can an attacker bypass Microsoft's patch, but they can bypass any canonicalization function check that was done according to Microsoft's best practices," which uses that API call. "As a result, a remote malware-infected server could take over any client that tries to connect to it."

And as I was doing the research in this latest bit of annoyance, I had the sense, and I'm sure our listeners do, and I know you do, Leo, because I've heard you moaning, that someone was not really paying attention over at Microsoft. We heard last year that they were really taking a much-needed long look at RDP and their other exposed server protocols, and that sounded like all good news. But the persistent incomplete "solving," in quotes, of this problem made me aware that as buggy, vulnerable, and patch-needy as Windows has become, at least we've always had the sense that the developers at Microsoft were quite highly skilled. And frankly, they need to be, to keep Windows moving in a more or less straight line. If that should ever change, Windows is done for. And you've got to wonder about someone just like not changing, not fixing the actual functions. Like they didn't read the whole memo.

Leo: Or they were in a hurry; or, yeah.

Steve: They just read the beginning of it.

Leo: It seems sloppy.

Steve: Yeah, yeah. I mean, and sloppy will be the death of Windows, Leo.

Leo: Oh, yeah. Oh, yeah.

Steve: The only thing that they've got going for them is they've got highly skilled developers because they have built, I mean, the term is "a house of cards." And as long as you're placing each card very carefully onto the existing monstrosity, okay. But if someone comes along and doesn't appreciate the delicacy of this thing, boom.

Leo: Boom.

Steve: So it turns out LFR is unfortunately an abbreviation we've going to be needing to learn. It stands for Live Facial Recognition. Its initial trial program used in London has been controversial, with the biggest problem being, aside from just the creepy Big Brother aspect of being monitored and surveilled without your explicit consent, the biggest problem is that, in the best of times, even before everyone was wearing COVID-19 masks, or subnet masks, the system was causing more trouble than it was worth, due to its extremely high palse fositive - boy - false positive rates...

Leo: I have those palse fositives. Yeah, I hate those.

Steve: For any of you who have improperly canonicalized this sentence, coming in as high as 90% in two recent LFR (Live Facial Recognition) deployments in which over 13,000 faces were scanned. Six individuals were stopped as a result of a match, five of whom had been misidentified by the system.

Leo: Ooh, that's not good.

Steve: That's not good. Civil rights groups say there is no clear legal basis for scanning the faces of potentially millions of citizens in the hope of catching a few people, with fears that a wholesale rollout could contribute to a shift toward a surveillance state model adopted in countries such as China. This was all happening in London. And I can say, I mentioned this during our Thursday panel last week, that it was with a little bit of surprise, but then I of course realized that my own well-trained iPhone looks back at me with a great deal of puzzlement when I hold it up to my masked face. It's like, uh, no. You're going to have to do better than that or type in your InstaCode by hand.

Leo: Who are you? You are? What's your subnet?

Steve: Yeah. So as long as wearing face masks is considered polite, bad guys wishing to avoid any chance of automated recognition can appear to be acting with full social responsibility by wearing a mask when they're getting about some business which is probably not socially responsible. And I guess it must come down to the question of whether people have, as we know the term is, a "reasonable expectation of privacy"

when they're out in public. And so law enforcement argues that, well, if you're in public, you don't have an expectation of privacy. Although, Leo, what's the law with taking pictures of people? Because sometimes we see people's faces blurred out when they're in public settings.

Leo: Yeah.

Steve: Clearly to protect their identity.

Leo: So photographers are allowed to take pictures of people in public without permission. But I remember when I worked at NBC that they wanted releases of anybody that would be shown on camera on NBC, regardless of where the location was. And they said, even if they're turned around, if your mother could recognize you, then we want a release. But that may have just been lawyers being overly proactive. And it is the case that responsible photographers, we always tell people, ask permission. Don't just go around shooting. But I don't think technically you have to. A mall, that's not a public place. Some sidewalks in front of buildings may not be public. But a truly public place, yeah, you could take pictures. You have no expectation of privacy.

Steve: Oh, that's interesting. I'm surprised that a mall would not be considered public.

Leo: Oh, yeah, that's not. That's privately owned. So you can be kicked out of a mall. You can be kicked out of a lot of places - train stations, airports. I've been kicked out of the best.

Steve: I don't want to know how you know that, Leo.

Leo: They don't, they really don't like you taking pictures. And if you're taking pictures at a power plant from the street, from a public area, you will get stopped. And, I mean, you have a legal right; but at the same time I don't know how much you want to assert that.

Steve: Well, and we know, for example, that there are blackout zones in certain areas of the globe where satellites are not allowed to be surveilling.

Leo: Well, yeah. Yeah, this only applies to the United States. Your rules may vary.

Steve: So Utah has done what I expect lots of states to do. They have chosen to roll their own COVID virus contact tracing app. To me it makes sense. They call it "Healthy Together," and it's on the App Store, both iOS and in Google Play.

Leo: So it doesn't - it's available now. That means it doesn't use the API, the Google/Apple API.

Steve: Correct. It does not.

Leo: They have their own thing.

Steve: It was created by a startup called Twenty Holdings, who is best known for their - well, they're not very well known. But if anyone knows them, I'm sure their mother, it's "Twenty - Hang Out With Friends." It's a social app which allows users, in their words, to "See who's around, see who's down, and hang out."

Leo: Oh, dear. Oh, dear.

Steve: So the company already, I'm sure this is what happened, they already had a platform for enabling physical in-person connections. So they thought, hey, we already have a contact dating app. We can easily convert it into a contact tracing app.

Leo: Sure.

Steve: So their Healthy Together app uses everything it can get its hands on, including physical location data, including GPS, WiFi access point proximity, cellular phone tower triangulation, and Bluetooth. Its goal is to pinpoint its opt-in users' locations and their location history and ID coronavirus breakouts and hotspots. And I've got to say, for those who are civic-minded and who do not mind the privacy implications of explicit historical tracking, since that's clearly what's going to be necessary for dealing with outbreaks, that is, at the state health services level, they want more than just individual, oops, I may have been near somebody who was positive. So am I really going to self-quarantine for 14 days? For me, I think this makes a lot of sense. There's nothing I'm doing in my life that's the least bit controversial. I mean, it's going to see this boring loop between my two work locations at this point. And my one tank of gas lasted I think two months, last time I filled.

Leo: Yeah.

Steve: So I personally would not hesitate to add that to my phone for the duration. Of course you're free to delete it any time you want to. If I were to become infected, and my location and time history could be played back to determine when and where that occurred, and everyone else who was also present in the same group at the same time could then be interviewed and checked, that would prove to be...

Leo: Fine. Yeah, yeah.

Steve: ...of vital use for health. And if, as it looks like, we are going to be reopening, and people are not going to be practicing the kind of safety that they need to based on some of what we're seeing currently with high-density restaurants and no one wearing masks, then I would argue that the responsible thing to do, the flipside of that, is okay, then opt into something like this so that your state can zero in on outbreaks and proactively notify you that, hey, you know, you were here at this time. Thank you for letting us know because now we're able to let you know you need to be very careful. Like we'd love to

have you come in and just get swabbed to see whether you might have the virus, but be asymptomatic, assuming that you're not showing symptoms.

The point is you really need something like this. They have a site, it's coronavirus.utah.gov/healthy-together-app. And it says of the Healthy Together beta app, they say: "Protect yourself and your family. Utahans are working to slow the spread of COVID-19. We can work together to protect our family members, friends, health workers, and our communities. The Healthy Together app helps you assess your symptoms, find the nearest testing center" - of course that means they know where you are, nearest testing center - "view test results, and learn what to do after you've been tested for COVID-19."

So they have four bullet points there up at the top: "Assess your symptoms. Use the symptom checker to see if you need to be tested. Two, find the nearest COVID-19 testing center. Testing centers are located across the state. Three, learn what to do after you get tested. Get your test results and instructions for care. And, four, location data. Find COVID-19 hotspots to focus public health efforts." And so they gave themselves a bit of a Q&A.

Question: "How does the Healthy Together beta app help protect me and my family and slow the spread of COVID-19?" Their answer: "The Healthy Together beta app helps you assess your symptoms, find the nearest testing center, view test results, and learn what to do after you've been tested for COVID-19. We can work together to slow the spread of COVID-19 and protect our family members, friends, health workers, and our communities. If authorized by the user, the app can also provide location data to public health workers, providing them with a faster and more accurate picture of where and how the virus is spreading within our community to focus public health efforts."

Question: "What happens to my data?" Answer: "Protecting your data is of utmost concern to the State of Utah; and the developer, Twenty, to ensure the privacy and security of the data, will follow these principles and limitations: Using the app is strictly opt-in and voluntary. You own your data and can delete it at any time. Only data required to combat COVID-19 will be shared with public health officials. Location data is automatically deleted after 30 days. Symptom data is automatically de-identified after 30 days. The developer will comply with state requirements for data security and encryption. You can decide what data you would like to share, for example, Bluetooth data, location data, or contact lists."

Question: "Who has access to my data? Is the data shared with any third parties? What details are shared?" Answer: "Your data is secure, and you are in full control of what you choose to share. Only data that is useful to combat COVID-19 will be shared with public health officials. While the state will have access to your symptom data, location and Bluetooth data will only be released to the state should you test positive for COVID-19." Now, that's a little limiting, in my opinion. Maybe you can share it otherwise because I'd like to be notified if I was in an area at a specific time when they believe the result of this demonstrates there was active virus in the air at that time, or on contact surfaces.

Anyway, question: "Which public health officials will have access?" Answer: "Utah has trained a team of contact tracers under the Utah Department of Health who will reach out to people who have tested positive for COVID-19 and have been potentially exposed to the disease. When you grant access to your location or GPS and Bluetooth data, members of this team will be able to access your data to help in the contact tracing process. The app will help these professionals identify transmission zones, contact patterns, and other vital information to inform their research. So vital epidemiological data."

Finally, question: "Why aren't you using just Bluetooth like Apple and Google are, or utilizing the API that Apple/Google built?" Their answer: "Bluetooth on its own gives a less accurate picture" - actually, as we know, deliberately nothing but contact information - "a less accurate picture than Bluetooth and GPS location data. The goal of Healthy Together is to allow public health officials to understand how the disease spreads through the vector of people and places, and both location and Bluetooth data are needed to accomplish that. Bluetooth helps us understand person-to-person transmission, while location GPS data helps us understand transmission zones. Having both of these important data points provides a more effective picture of how COVID-19 spreads. This data helps policymakers make the best possible decisions about how and where we begin to relax and modify restrictions as our community and economy begin to reactivate."

And I say hallelujah. Again, if there was something I could download for California, I would put it in my phone and turn it on and share my location data, which as I said is very non-exciting. But I would love to get a notification, if after the fact I'm informed that where I was at the time I was, people were getting infected. To me that would be useful and certainly worth the tradeoff. And again, it's not for life. It's not forever. All of that is self-expunging after 30 days, as it should be. That's correct design because nothing matters past that point. I just think it makes a lot of sense.

So as everyone knows, I saluted the technology from a crypto standpoint of what Apple and Google did. That's what we analyzed. Bruce Schneier thinks all of this is nonsense. But boy, voluntary location tracking, I don't know, Leo, you've probably see some of the news where already anonymized cell phone data has been used just to watch people. There was one where the aggregate of people who were on a certain day in a meatpacking plant, their phones were anonymously followed as they just dispersed literally across the country, and it was just fascinating to see how valuable that kind of location data is, even if you don't know who it is.

Leo: Right. No, we've seen a lot of that, actually. Presumably anonymous. But still, yeah.

Steve: Yeah. SpinRite. Work is continuing quite well on the project. Yesterday, before switching to work on this podcast, I posted my just-completed, full, from scratch, FAT-partitioned formatting code. It's just a simple, I think it was 8K before I signed it so that Windows would be happy. And that exploded it to 18K. But okay. People were saying, "Hey, Windows says it doesn't know who the publisher is." I'm like, ooh, I forgot to sign it. But anyway, I need SpinRite's thumb drive boot installer to be able to work on any old or new thumb drive the user might have around, regardless of that drive's history, because who knows what users will have.

SpinRite's current installer, which I wrote back in 2004, as I think I mentioned on the podcast last week, I was shaking my head because there was code in there for doing this on Windows 95. Anyway, it's showing its age and is in need of some rework. So I wrote a, from scratch, FAT12, 16, and 32 partition formatter that will just take any thumb drive it is given. It'll remove what's there, install a master boot record. That's the next piece of work I will start on this evening, and then put a FAT format partition of whatever size is necessary based on the size of the drive, and then install FreeDOS and the test code that we'll be using. And that should be finished pretty - that'll be finished shortly. And at that point everyone will be able, who's testing, to more easily boot the testing code on their own machines.

And then the new AHCI driver code that I've talked about before, I'm still - there's a couple little cases I'll get back to, some specific chipsets where it looks like it's still, as I mentioned before, there was some chipset, I can't remember now, where it just wasn't

supporting a couple bits in the spec, and I was expecting they all would. This one didn't. So I was like, okay, fine. I can work around that. So there will be a couple more things like that. Anyway, it's why I'm excited to put this code out as a really cool raw performance benchmark for all of this podcast's listeners. Everyone will be able to play with it. I'm sure we will find additional systems where there are some problems. I want to find them because I want to fix them. So anyway, we'll be able to involve everyone here before long. At the moment we're just working in the GRC spinrite.dev newsgroup.

Okay. I titled this "WiFi 6," the podcast, because that's where we're going to end up. But I thought it would be interesting to do a bit of historical framing to place WiFi 6 into the context, historical context, since as I mentioned the 47th anniversary of the invention of Ethernet - not the Internet, of Ethernet, although they pretty much were coincidental, as we'll see - three days from now on, where is it, May 22nd, 1973.

So Ethernet was invented by a guy named Bob Metcalfe. Nice guy. He and I were on a couple panels back in the day. We've talked a lot on this podcast about packet switching. Bob is one of the people who built some of the very first hardware. In 1970, while I was rapidly falling in love with assembly language on a PDP-8, and Bill Gates was playing with a newly installed teletype at his high school, which was hooked to a remote timesharing system, Bob Metcalfe was building an interface known as the IMP, I-M-P, which stood for Interface Message Processor. It linked a PDP-10 at MIT to the ARPANET, as it was known at the time.

Three years later, in 1973, after building a second IMP host interface at Xerox PARC, which is where Bob was, he was assigned the task of somehow extending the ARPANET into buildings full of PARC's personal computers, which of course at the time was the only place in the world where it had occurred to anyone that computers might actually be personal. It was on May 22nd of 1973, in three days 47 years ago, that Bob wrote the memo inventing Ethernet. So today, 47 years downstream, more than 1.2 billion new Ethernet ports are shipped every year. One third of them are wired; two thirds of them are WiFi.

So let's follow the path that leads from there to today's WiFi 6. Bob's wired Ethernet first appeared commercially in 1980 and was standardized by the IEEE, that's the Institute of Electrical and Electronics Engineering. It was standardized as "I Triple E" 802.3. And of course it would become a growing family of electrically connected Ethernet adapters over time. Three years later, in '83, we had 10 mbps, with 10Base-5, which used a thick coax cable, not very easy to work with.

In '85, two years later, the much more popular 10Base-2 switched to a much more manageable thin coax. That's the first one I used when I ran that horseshoe loop around the building. You had to do a horse - it was like a straight line. You had terminators on each end so that the signal that hit the end would not reflect back. It would just it would absorb it smoothly. And then you put T adapters to sort of T connect into the thin cable wherever you had a PC. So we had a sort of a facility early in GRC's day where a big U-shape allowed us to connect everyone's machines. And it was like, wow, this is amazing. In 1990, we got 10Base-T, where the T stood for "twisted." That was twisted pair, which initially ran at the same 10 megabits. Then five years later we saw the jump to 100Base-T.

That brings us to 1995, which was also known as Fast Ethernet because now 10 megabits was slow. And that of course gave us a tenfold jump to 100 mbps. In '98 we got 1000Base-X, which was for one gigabit over fiber optic cabling. And the next year engineers had figured out how to deliver the same speed over the much more convenient multiple twisted pairs, giving us 1000Base-T. In '97, when our 100Base-T was then a couple years old, engineers began looking at wireless. Whereas the family of wired Ethernet were all 802.3, the IEEE assigned 802.11 for their work on taking the same

time-proven Ethernet technology wireless. The very first 802.11 was mostly experimental, so this again, first wireless Ethernet.

It was quickly followed up two years later, in '99, by 802.11a. Whereas the first 802.11 operated in the 2.4 GHz band, and was only able to deliver around one to two megabits per second, 802.11a moved into the 5 GHz band with a physical, so it's like a technical maximum bit rate in the air, of 54 mbps. But it also relied upon a lot of forward error correction so that it ended up delivering an effective data rate kind of down in the mid-20 mbps range. The 2.4 GHz band was already and now to the point of being crowded with microwave ovens, Bluetooth, baby monitors, cordless telephones. Some amateur radio equipment operates there.

So moving 802.11a into the relatively unused 5 GHz band gave it a significant advantage. But the higher carrier radio frequency, 5 GHz versus 2.4, brings some tradeoffs, since a higher frequency means a shorter wavelength, and a shorter wavelength increases the signal's absorption by walls and other solid objects in their path. So again, sort of a tradeoff. And we'll see we're moving, sort of jogging back and forth between these two bands.

As a consequence of the fact that 5 GHz was okay, but causing some problems, the next move was to work to improve the data performance of the inherently more robust 2.4 GHz lower frequency band. That gave us 802.11b. And products starting 20 years ago, back in the year 2000, were based on 802.11b. It used a more advanced carrier modulation scheme to deliver a maximum effective bit rate of around 11 mbps. The products were inexpensive and plentiful, and WiFi really began to take off because it was like, okay, now this thing works. And since 802.11a operated at 5 GHz, and 802.11b operated at 2.4 GHz, it was feasible to create dual-band WiFi, known as 802.11a/b.

Three years after that, 802.11g delivered a third carrier modulation standard for the lower band 2.4 GHz. It managed to deliver the same 22 mbps throughput of the much more finicky 5 GHz 802.11a, while operating in the inherently longer range, but more congested, 2.4 GHz band. And as with 802.11a/b, all three modes then were often combined to yield 802.11a/b/g.

So at this point things had become a bit of a mess. Just evolution does that. So work was undertaken to pull all of the various amendments to the original 802.11 together to figure out what to do next. The result of that work emerged by the end of 2009, and that was 802.11n. And even though, and we talked about this on the podcast at the time because we were here then, the industry didn't actually wait for the publication of the formal standard. It jumped the gun by a couple years by following the first draft specification two years previous. There was just too much pent-up need. And everyone figured, well, okay, we hope when 802.11n is formalized, that we're still certified, or we will be able to get certified. Because right now we're just hoping this is what it's going to be.

802.11n combined everything. And it would later come to be retroactively labeled WiFi 4 by the Wi-Fi Alliance. So this is their attempt to begin, you know, they recognized, okay, this is an alphabet soup of confusion for the typical consumer. Let's drop all of this 802.11a/b/g/d/e/f and just say WiFi 4. So that's what 802.11 later became named as, retroactively. The other thing it introduced was the three-antenna MIMO, M-I-M-O, Multiple Input/Multiple Output system. And being that it pulled everything together, operates on both the 2.4 and the 5 GHz bands, although support for 5 GHz technically is optional in the spec.

Okay. Seven years ago, in 2013, this 802.11n, also known now as WiFi 4 standard, had its 5 GHz transmission channel bandwidth significantly widened from 40 MHz to 80 or 160. The wider the band, the higher the data rate you're able to use in that band. You

have more bandwidth. And the encoding of the data jumped from 64 QAM, which is Quadrature Amplitude Modulation, to 256 QAM. So it jumped from encoding six bits at a time to eight bits at a time. And since it's all just silicon, they also defined something known as Multi-User MIMO, MU-MIMO, all of which results in where we went next, 802.11ac, which was also later retroactively labeled "WiFi 5." So it took 802.11n and widened the channel bandwidth and improved the modulation scheme to be able to store eight bits in a time interval where before they could only store six.

But we should pause our history here for a minute to talk about this. The original Ethernet used a system, that is, to talk about this MU-MIMO, the Multi-User MIMO, the original Ethernet used a system which we talked about on this podcast years ago, back in our earlier tutorial phase, known as Carrier Sense multiple access with collision detection. That's what Bob invented back then. That was the essence of his genius. It was essentially a party line. And remember the old days, I think this predates both of us, Leo, but like the first telephones, you actually often had multiple subscriber lines, they were also known as, where you'd pick up the phone...

Leo: Party lines. Party lines.

Steve: ...yeah, to see if anybody was already using it. And if not, you would maybe, what, flash the hook switch in order to get the attention of the operator and then have her connect you somewhere. And if you picked the phone up, and there was already a conversation going on, politeness required that you not eavesdrop, but that you put the phone back down on its hook and then come back later.

Leo: Marge, I told her when I came in there that I really didn't like the way she was wearing her hair. And then...

Steve: Exactly.

Leo: Yeah. Really awful.

Steve: Exactly.

Leo: I'm talking here, Leo. Get off the phone.

Steve: How much longer are you going to be? You've been on for the last three hours.

Leo: Yeah, exactly, yeah.

Steve: There's still plenty of rural areas where they don't have enough carriage capacity to have anything but a party line, believe it or not.

Leo: Today, still?

Steve: I've heard from people who say, yeah, we still have a party line. Maybe that's - yeah. Isn't that amazing?

Leo: Wow.

Steve: Well, in a sense we all have WiFi, so we do have a party line still.

Leo: That's true, yes.

Steve: Bob's invention, his brilliant invention, was to figure out how to create a simple solution that allowed multiple nodes to share one medium. It was initially coax, then it went to twisted pair. Although the topology for twisted pair was not the same T connection, just sort of tie in. But the concept was essentially a party line. The idea was that someone wanting to send data to someone else would listen on the connection until the shared coax was quiet. Then they would transmit their data.

But it was entirely possible, especially on a busy shared network with many nodes, that multiple parties who were listening for a pause might start transmitting at the same time. Back when Ethernet used coax, such a collision would actually create a higher voltage swing on the coax because of two transmitters trying to do the same thing at the same time. That was readily detected by everyone on the line. So the parties who were responsible for that jam-up would wait a random length of time. They would back off a random length of time before retrying to send, also listening to make sure that the line was still quiet.

So the system was elegant, simple, clever. And later, when twisted pair wiring was used, there was not that overvoltage event. So the transmitting parties would listen to the line while transmitting to see whether they were able to reliably receive the message they had sent. If not, that meant that someone else had collided with them, transmitting and interfering. So again, each party would back off a random interval and retry.

So the point was Ethernet has always been a shared medium technology. When semi-intelligent Ethernet switches replaced simple repeater hubs, things got better for twisted pair, since the semi-intelligent switches could dynamically learn the network topology by building a table of which Ethernet MAC addresses were connected to which of their ports. So now, with that technology, rather than simply sending anything incoming back out of all ports, the incoming data would automatically be routed out of only one port, where the destination MAC address had previously been seen and was known to reside. So this hugely reduced packet collisions within large networks.

And that's the one problem with Ethernet - we see this also with the shared medium of the air, which we're all using now with WiFi - is that, if you think about it, and all kinds of academic research has been done about the shape of the curve of the collapse of Ethernet, when too many people are trying to talk at once, because what this elegant simple system lacks is a means of handling too many people. That is, it basically collapses. And you mentioned Token Ring at the beginning. That's the approach that IBM took where a virtual token circulates in a ring. The person holding the token has permission to speak. When they're done, they pass the token to the next station. That's the way IBM did this. But they didn't win.

Bob's simple system, which it sort of models the Internet in how it's, you know, you wouldn't think it works. It's simple. It's clever. You mean I'm just going to put this packet on the wire, and I don't know how it's going to get there, but it is? Uh-huh. That's

right. Similarly, you mean I just wait till nobody else is talking, and then I can talk? Uh-huh. Yeah. Of course, if everyone's trying to talk at once, you have a problem.

Okay. So the point is that switching helped a lot. So 802.11ac, also known as WiFi 5, introduced this MIMO technology. It was actually defined, and equipment was certified, in two rounds over time, that is, 802.11ac, the second round only more recently in 2016, which added the higher bandwidth capabilities of Multi-User MIMO, the widest of the channel expansion, to 160 MHz channel bandwidth and additional 5 GHz channels. It also doubled the number of spatial streams with four antennas versus three, which were in that first round.

And I thought I said somewhere here, I don't see it, where I explained that the way this MIMO technology worked was very cool. The access point initially had three antennas. It would use varying phase of its transmitted signal where the phases would, depending upon the angle of the receiver to the antennas, it would create null zones where the phases canceled and extra power zones where the phases summed. And so the access point would periodically poll the receivers, saying how do you hear me now, how do you hear me now, how do you hear me now, using different phases among the antennas. The receivers would send back a "how loud" that was.

And what that did was it allowed the access point to adaptively learn effectively where these receivers were, assuming, which is often the case, they are relatively fixed in an environment, and to then essentially do beam forming. That is, if it knew it was sending something back to a given receiver, it would look in its table to see what it may have learned about the receivers' self-reported optimal phase of the antennas and then use that phase essentially to beam this signal to that receiver, minimizing interference where the phases were not aligned and were out of phase.

So again, not something you want to wire up in the backyard. But once it's all on silicon, it doesn't cost anybody anything except some extra transmitters, and those are cheap now. So that brings us to 802.11ax, also known as WiFi 6. 802.11ax aims to quadruple a single access point's overall data exchange capacity, the benefit of which will most be heavily felt in heavily used multi-client environments because it's not like one client gets four times the throughput, no. It's that the system degrades far less quickly. As I said, you can imagine everybody talking at once. That's the failure case of Ethernet because you end up just with constant collisions and everybody backing off and trying again, then more people talking. So an individual gets 30% more performance out of an 802.11ax than WiFi 5.

So WiFi 6 gives about a 30% boost. But when you've got a lot of clients of an access point, then you begin to see an improvement. It also uses this so-called multi-user MIMO, which can be sort of thought of as spatial domain multiplexing. That is, it multiplexes the space around the access point by learning what individual users are doing to reduce inter-client interference. And then to this notion of the spatial domain multiplexing, "ax" significantly enhances what's known as frequency domain multiplexing with what's called MU-OFDMA, Multi-User Orthogonal Frequency Division Multiple Access. It's a fancy way of saying that the entire available spectrum that the access point has allocated to it is dynamically divided into a very much larger number of very much smaller individual subcarriers. And by very large number I mean 2,048 individual subcarrier channels.

So this allows a much higher level of individual attention to be given to each client. Clients are essentially able to receive not only a signal, a radio signal aimed at them, which the access point learns, but they're also able to receive their own allocation of subcarriers that will inherently not collide with other clients in the area. So essentially it's managed to take what would look like a huge shared medium and create both essentially

physical beams and radio channels within that region in order to divide that monolithic space up into the ability to serve individual clients.

So this is the kind of thing which in huge auditoriums and stadium settings you would want tons of these access points spread around and expect to get much better performance. Except you need to have a compatible client. Of course the system works with down spec clients with no problem. But you don't get a lot of these features because they are negotiated between the access point and the client on the fly, unless you've got matching WiFi 6-compatible clients.

Oh, and one other thing, too. There's a significant power savings available for mobile clients, thanks to something known as "Target Wake Time" (TWT), where the access point and the client are able to negotiate silent periods which allow the client to shut things down and not worry.

And then in a final piece of news, just last month the FCC delivered some very good news to the WiFi industry overall by agreeing to open and make available a significant chunk of new bandwidth, so-called "unlicensed bandwidth," in the 6 GHz band. This is an additional 1200 MHz worth of WiFi spectrum which will add 14 80 MHz channels and seven 160 MHz channels. So we can expect to see further reduced interference, even lower latency, gigabit speeds, and higher capacity for simultaneously managing many more devices.

And I forgot to mention that one of the things that WiFi 6 also does is it expects to dramatically cut client latency by 75%. And anybody who's ever tried to use satellite Internet, I know that Elaine had satellite Internet for a while where she was, it's just - it's really a pain. Latency is just a thrill killer in Internet use. So WiFi 6 should improve that, too.

I was getting ready to make the move myself to WiFi 6. But since I don't yet have many other WiFi 6 client devices, and since last month's chunk of new unlicensed bandwidth means that there will be another generation of access points coming along, I think I may wait until the newer access points, which include the 6 GHz band, hit the market. But I don't expect those devices soon, since it'll likely require some tooling up of new silicon and some radio design from scratch. So anyway, depending upon your need for WiFi 6 today, you might choose rationally to wait or just say, hey, I want those features now. I think, Leo, you said you were going to jump to 6.

Leo: I have 6 now with my Orbi. Yeah, it was very, very expensive. And I guess it's better. The problem is so few WiFi 6 devices. So the iPhone 11 is. You have that. The MacBook Air, the new MacBooks are not. My Dell XPS, the brand new one, the 2020, is.

Steve: Actually, I have an iPhone 10. Do I have 6 or 5?

Leo: No, you have 5.

Steve: Oh, yeah. And then I have a whole bunch of old iPads.

Leo: Yeah, and I don't think even the newest iPad has WiFi 6. So it's silly to get it. It's a huge expense. And as you say, 6E is coming. So I would defer until you - and then you'll have enough. It's really - I think 6E is really aimed at IoT devices

because it's such a high frequency that it's not going to go through walls at all. So I think it's really intended more to handle the vast number of IoT devices we now find in people's houses than anything else. I know. I'm really, you know, I'm - you know what we're doing? We're getting a guy come out, going to put in Ethernet, Bob Metcalfe's brilliant invention, in every single room.

Steve: I'm totally with you, Leo. I was thinking about that while I was reading this. And it's useful to have WiFi where you need portability. But my own network here, I'm 100, well, you know, I mean, you really don't want any podcasters to be trying to do this with WiFi.

Leo: No, because of the collisions, because of that collision-based system, yeah.

Steve: Yeah. Nothing is better than wired.

Leo: Now, I think 6 - did you notice whether that collision detection is still in 6? I get the feeling that they were trying to do stuff to avoid that.

Steve: Well, they really are. So that whole spectrum allocation technology...

Leo: Oh, okay, that's the idea is we'll just be on different frequencies, and it won't matter.

Steve: Exactly.

Leo: Of course, there aren't that many frequencies. Even though it says zero to 11, there's really only three bands you can operate in, in those 11 frequencies, or 12 frequencies. So it's not like we've got all the spectrum everywhere. But it'll be nice to have more channels, have 6E. It'll be good. You know what, I love it when you do the explainers, to be honest with you. So I hope there's no security problems next week, and we can find another topic...

Steve: Who knows what I'll come up with to talk about.

Leo: Yeah, yeah, a lot of fun. Steve Gibson's at GRC.com. That's his website. That's where you'll find SpinRite, of course, the world's best hard drive maintenance and recovery utility. Recommended it on the radio show on Sunday, actually. A guy had a perfect example of a drive that SpinRite could help. You can also find lots of free stuff there, including ShieldsUP! to test your router and all sorts of information and a lot of little rabbit holes you can crawl down and find out things about including SQRL. There's a SQRL down that rabbit hole. And Vitamin D, you know, I just read another study that said Vitamin D seems to be a strong indicator in your chances of surviving COVID.

Steve: They keep coming out, yeah. They're seeing real correlations between Vitamin D status and how you do if you get infected.

Leo: Yeah, yeah. So I'm taking my 5,000 IUs every morning, thanks to you, as I always have. Anyway, it's all there, GRC.com, including, by the way, this show in two unique formats. He has a 64Kb audio. We've got that, too, at TWiT.tv. But he also has a 16Kb audio. Sounds a little scratchy. Sounds like Alexander Graham Bell. But it's small. And that's its advantage. There's also another very small file format, text, a very nice transcription. Elaine does those. And those are both available at GRC.com in the Security Now! area.

We have 64Kb audio. We have video, as well, if you want to watch. All of that's at TWiT.tv/sn. It's on YouTube, as well. A lot of people do that. You can listen on your favorite voice-activated device. Just say "Echo" or "Google" or whatever, "Play Security Now! podcast." You'll hear the latest episode. We do the show on Wednesdays. I'm sorry, Tuesdays.

Steve: Tuesdays.

Leo: Used to be Wednesdays. Tuesdays, 1:30 Pacific, 4:30 Eastern, 20:30 UTC. So if you're around at that time you can watch us do it live at TWiT.tv/live. On-demand downloads available at TWiT.tv/sn. Best thing to do, subscribe. That way you'll get it the minute it's available, audio or video. Just find your favorite podcast application and sign up today. It'll cost you nothing, despite the word "subscribe," which I think has always scared people off, like you're paying for it. But you're not.

Steve, thank you so much, sir. We'll see you next week on Security Now!.

Steve: Right-o.

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/