Wi-Fi Protected Access v3

Description: This week we discuss the interesting case of a VirusTotal upload - or was it? We've got newly discovered problems with our 4G LTE and even what follows; another new EFF encryption initiative; troubles with Spectre and Meltdown in some browsers; the evolution of UPnP-enabled attacks; an unpatched WordPress vulnerability that doesn't appear to be worrying the WordPress devs; and an early look at next year's forthcoming WPA3 standard, which appears to fix everything!

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-670.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-670-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. We're going to talk about VirusTotal and the interesting case of a virus leaked before its time. Also, what WPA3 will mean to you, and when we'll get it. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 670, recorded Tuesday, July 3rd, 2018: Wi-Fi Protected Access v3.

It's time for Security Now!, the show where we protect you and your loved ones online with the help of our Explainer in Chief, Mr. Steve Gibson of the GRC, Gibson Research Corporation, GRC.com. Hi, Steve. Good to see you.

Steve Gibson: Yo, Leo. Great to be with you again, as always.

Leo: Of course, of course. And today we have a fabulous show lined up for you. I shall only be interrupting it three times. I'm going to let you just take it over.

Steve: This is the pre-Fourth of July episode of Security Now! #670. And it was announced at the beginning of the year that this was going to happen, and then early last week it finally did. But these are not my favorite people, as we all know. This is the Wi-Fi Alliance. And we've often said that they haven't yet gotten everything right, yet it's one of the most important protocols, arguably, WiFi security, probably increasing in importance fast. And it's just been a lesson in how not to develop really important protocols; whereas the fundamental architecture of the Internet was all done academically in full view. I mean, there's this amazing trail of public discourse where problems were resolved.

Unfortunately, this Wi-Fi Alliance is a closed group, nothing in public. I was excited this morning as I began digging into the details of WPA3, where there were links to the specification. It was like, oh, my goodness, like they've changed. It turns out, after filling out a form telling them who I am, where I am, what's going on, I mean, basically just here's all of my contact information, please, please, please let me have the specification, I get like this seven-page header with nothing in it.

Leo: We wouldn't want you to know how it works, Steve.

Steve: And, I mean, it had to, I mean, the WPA3 spec has to be a thousand pages, or hundreds. And then I thought, oh, well, okay. And then there was one on opportunistic encryption specifications. I thought, okay, maybe I'll do better. So I immediately went back and got the same form, so they didn't remember me after I'd just bent over, so I did it again. It's like, okay.

Leo: You did this all for us, Steve. Thank you.

Steve: I really want this. So did it all again, clicked the link, seven pages. You know, basically an abbreviated table of contents and some acronyms defined. It's like, ugh. So anyway, same story. But the good news is, if we give them a lot of benefit of the doubt, what they're saying about WPA3 is all good. I mean, I'm bullish about the promise. Unfortunately, there's no ability whatsoever to, like, verify any of this. But we're going to talk about what it is that they say they've done. But it doesn't affect us yet, either, because it's expected late 2019. So, okay. Because of course everything has got - it's got trademarks on everything, and certified everything, and it's all locked down, and they're in control, and they're going to put their WPA Wi-Fi Alliance stamp on everything, and that takes time, you know, to make those stamps. So someday WPA2, which has been broken - and I'll remind people about that later, with the KRACK attack. And of course everything else earlier is badly broken. Someday we're going to have an improvement, and it's good.

So in the meantime we've got an interesting case of a VirusTotal upload mistake, or was it? Newly discovered and reported problems with our current 4G LTE cellular system - worse even than we knew, and we already knew that was a problem - and even apparently what follows it, the 5G system. The EFF has launched another encryption initiative, a detailed look into which will probably be the topic for next week because it's big and I want to do it justice, but we'll talk about it.

We've got some troubles with Spectre and Meltdown as it exists now in browsers post-mitigation of those problems. Sort of the predictable - but sure enough, things that get predicted often happen - evolution of attacks leveraging the Universal Plug and Play features of today's so prevalent routers. An unpatched WordPress vulnerability that got a lot of press last week, but doesn't appear to be worrying the WordPress devs because they've known about it since last November. And then we'll talk about the WPA, what we can of the WPA3 standard, at least the hope and the promise that they're offering, and why it looks like it's a good thing. So I think a fun and interesting podcast to get everybody ready for their Fourth of July barbecues tomorrow.

Leo: Sounds like it.

Steve: Our Picture of the Week came from my archive of photos, since there was nothing particularly pertinent for this week. But I got a kick out of it. It was some conference, a security conference sponsored by, because you could just see this at the bottom of the screen, Oracle, Red Hat, ING, and Google. And it's a big slide that says the first law of software quality. And it reuses the famous expression e=mc2. In this case they redefined "e" to be errors and "mc" to be more code.

Leo: More code, more problems, as the kids would say.

Steve: Yes. And of course this is something our listeners are so familiar with me going on about endlessly, the idea that the more code you have, the more opportunities for a mistake there is. Less code, fewer errors. So anyway, errors equals more code squared, meaning yes, and it does. It goes up. And we've also talked about this, the fact that it goes up exponentially, because all the code you have is interacting with all the other code you have. So as you add code, the opportunities for interactions goes up nonlinearly, much more quickly than the code you're adding would suggest. So anyway, keep it small. Keep it simple.

Leo: Yup.

Steve: Okay. So this was an interesting story that I got a kick out of. And at first it looks like it may have been a mistake. But upon further reflection, I'm wondering. It involves the discovery on VirusTotal of a previously unknown, in other words zero-day, exploit. So, okay. So we've never talked about VirusTotal before, so we should do that. VirusTotal...

Leo: Yeah, who are these guys? What...

Steve: Excuse me?

Leo: Who are these guys when they're at home?

Steve: So VirusTotal is a very cool system. And I'm surprised it's never come up somehow in nearly 12 years of this podcast because I have used it from time to time. I know that a number of our listeners know of it. Simon Zerafa, a friend of the podcast, is frequently checking things for me. And as we know, a lot of our antiviral systems have, by necessity, needed to become heuristic. There's this cat-and-mouse game where they're trying to stay current, and they have so-called "virus signatures" which cue them that there may be a problem. But viruses can morph, and do.

And so increasingly, antiviral systems have been using signals to tell them that this could be malicious, rather than knowing it's malicious, because they previously saw exactly the same thing. And this is why, from time to time, as a software publisher myself, I'll get reports that, like, oh, the DNS Benchmark has malware in it. And it's like, what? First of all, the DNS Benchmark hasn't changed in, you know me, I think I finished it in 2010, so eight years. I mean, nothing has happened to it. And it's digitally signed, and the signature is still valid, so no one has horsed around with it on my server. And it's like, there's nothing wrong.

But suddenly some AV tool just decides, oh, this looks suspicious. Let's warn people. And so, okay. How do you know, like what does an end-user do in that situation? Well, VirusTotal is there for us. And it ought to be in everybody's toolkit, everybody who's listening to this podcast who doesn't already know about it. And VirusTotal is the domain name. I don't know if it's dot org or dot com. But if you put "VirusTotal," all one word, into Google, it'll take you right there.

Leo: It's owned by Google, we should mention, if that makes a difference.

Steve: I didn't know if it was. I thought it was, but when I was digging into it, it's something that looked Irish or something, CSLI, or I don't remember what the acronym was. But I also thought it was owned by Google. But when I dug into it, it looked like it had a different owner.

Leo: Well, it's owned by a Google - an Alphabet subsidiary. I guess that counts.

Steve: Okay. Right, right, right. Yes. And so, okay. So what it is, is sort of social networking comes to antivirus and malware research. An individual who suspects something on their machine might be, like, hinky, can upload it for free, the service is completely free, to VirusTotal. And that service, that web-based service runs that thing, that file that was uploaded, through over 70 antivirus scanners that are shown by name. And so it's like all of the AV engines of the major players in the industry are accessible to VirusTotal and are interested themselves in seeing new things. So it's sort of a win-win-win-win-win. Everybody wins because the end-user gets to say, uh, my computer started acting weird after I downloaded this thing that automatically sends love letters. Is it safe?

So somebody questioning some software can upload it to VirusTotal. And what you get is this really gratifying, comprehensive report of, by engine, by AV scanner, what each of them thought of what you just submitted. So it's just this cool system. And part of the terms and conditions of doing so, which you implicitly agree to, is that you're turning over the intellectual property, I mean, you're alleging that it's yours to offer for analysis; and that, in doing so, you're explicitly letting other people look at it for this purpose.

So what happened was that, back in March, ESET, which is one of the well-known AV players, was closely examining a PDF which had been uploaded to VirusTotal and suspected that they might have found a potential exploit for an unknown, to them, Windows kernel vulnerability which was hidden within the workings of that PDF, which somebody put up on VirusTotal. So they forwarded this possible discovery to Microsoft, who subsequently confirmed the presence of two previously unknown zero-day exploits in that PDF. One was leveraging a flaw in the JavaScript handling of Adobe Acrobat and reader.

The other turned out to be a potent, never before seen elevation of privilege in the Windows kernel. And as we know, elevation of privilege is like a key component of forming a powerful exploit chain because you're wanting to get - you wanted to elevate yourself to system privilege so you can get up to more mischief. And as we're always saying, and as is commonly known, not running as a privileged user is a good way of protecting yourself and your system since the things you can do and the things that can be done on your behalf are dramatically curtailed.

So, okay. So what was believed was that this malicious PDF was likely in the early development stages since the exploits were fully functional, yet the PDF did not contain a

malicious payload. There was just sort of a proof-of-concept stub, as is often the case, for example, calc.exe, the little Windows calculate gets launched, which is a safe thing that is often used as a proof of concept to say, look, we just ran something that you didn't explicitly run. So it was like that. It was like all of the machinery was there for an unknown exploit, but it wasn't itself malicious. It was like the development stage.

So what was assumed was that the exploit's developers, well, what was initially assumed was that the developers made a mistake by uploading this because this proof of concept got away. But in thinking about this further, I wonder if that's the case. Some reporting of this in the industry suggested that it was a mistake. In fact, quoting from some of the coverage I have in the show notes, it seems someone could have combined both the zero days to build an extremely powerful cyberweapon. But they had intentionally and mistakenly lost the game, as it was written, by uploading his or her under-development exploit to VirusTotal. And certainly that could be the case.

Microsoft wrote: "The sample does not contain a final payload, which may suggest it was caught during its development stages. Even though the sample does not contain a real malicious final payload, the authors," Microsoft wrote, "demonstrated high-level skills in vulnerability discovery and exploit writing."

And so there's another way to process this same set of facts. Given that there's a 60-day response Window from the VirusTotal upload to the patch - oh, I forgot to mention that this is the story, this is the back story behind two of the fixes from last Patch Tuesday, from the second Tuesday of May's updates. There was an Adobe zero-day patch and vulnerability and fix, and there was one for Microsoft's kernel. These are those.

So it took two months to go from the event of the upload on VirusTotal - oh, and by the way, this PDF, because as I said it was zero-day, it looked clean. VirusTotal gave it a clean bill of health. Looks like you just uploaded a regular PDF, says VirusTotal, when in fact something induced ESET to look more closely. And they said, whoa, this is not benign. So imagine that there's a 60-day window, and that the bad guys know this, know that the whole process, the cycle takes about two months. So this could have also been a carefully and deliberately calculated gamble on the part of the attackers because, if they were as good as Microsoft admits they were, it seems unlikely that they would have made such a simple mistake.

Leo: But why would they upload it then?

Steve: Well, because based on their target - so this kind of a PDF would be part of a phishing attack where the malicious payload would be bound into a phishing PDF aimed at someone. And it might have been more valuable to them to determine whether it would set off any alarms with the recipient relative to the size of the length of time they knew they had to use it. So, for example, if they developed it, gave it a benign payload, sent it to VirusTotal, and it was immediately recognized, then that would tell them that this was not - they didn't have zero days. They had something that somebody already knew about.

Leo: And now the clock is ticking, they have three months to use it.

Steve: Exactly. And in the world we live in today, this is what it's come down to is you can imagine, if a target was valuable enough, and the probability of a successful spearphishing infection at that target was high enough, in an environment where raising an alarm by its detection would be sufficiently detrimental, then they could have made a

cold calculation. It's like, well, because they don't need 60 days to launch their spearphishing. They only need one day. But so it might have been, I mean, they would know that sooner or later these were going to be found, like this is what happens. Zero-days get seen in the wild, they get detected, they get found, they get added to everybody's AV heuristics.

So these zero-days have a limited life always. And they're only valuable while they're not yet known. And so but there is this 60-day window. We saw it. From March to May this thing was not recognized by any AV and patched against being used. So anyway, it's interesting that, I mean, where now we're in this spearphishing mode where we're deliberately targeting specific individuals to get into, you know, like state-level sponsored cyberwar, where it's very valuable to establish a foothold if you can, when someone comes up with a zero-day, they need to know if it's going to have a chance to succeed. And 60 days, that's plenty of time for them.

So just I wanted to take the opportunity to talk about VirusTotal for any of our listeners who aren't aware of it. It's a very cool system, the idea that you can easily send something there and have it checked. In the early days of my work on SQRL - actually when I added the installer facility to SQRL. SQRL contains - it's sort of a self-installer. It's the same size EXE, didn't grow very much. But when you run it, it notices it hasn't been installed. And it doesn't actually need installation, but users are used to that, and they're going to download it into their browser's download folder, and you don't want to run it from there. So we needed a process for moving it into a final place and kind of getting it configured in the system, although it also was doing that for quite a while without any help.

But something about my addition of the installer, because I'm messing with the registry, like registering it for autostart optionally and a few other things, something that I did set off Windows Defender so that it was slowing it down for a while. That's gone now, so it learned to trust SQRL, which is nice. But several times I uploaded it when I was trying to figure out what was it that I did that caused it to suddenly get examined? And the point was it slowed things down because Defender was jumping in there and saying, okay, wait a second, let's take a look at this.

And so I was playing with different types of EXE compressors, seeing if there's anything I could do that wouldn't upset Windows Defender. It must have been something fundamental about the fact that I was doing installation-y things where some heuristics got triggered. Now they don't slow it down at all. But I was using VirusTotal myself, sending copies, like various tests of the SQRL EXE, up to see if anybody else was upset by it. It never set off any alarms. But it's like, well, okay. I don't know. And it finally just kind of went away on itself.

But for what it's worth, VirusTotal is cool. And I wouldn't be at all surprised if this was not a mistake on the part of these guys. If they are that sophisticated, they know the world cannot respond instantly. But what's more important to them is knowing if they do actually have a zero-day, or if what they have discovered - I mean, because really, how would you know if this was already known? You would know that your exploit was effective on systems that you had, so you would know that the systems you had today were not patched for it. Thus it's an exploit. But you wouldn't know if any third-party AV that you don't have might not know about it.

And remember that all of our AV now is phoning home. So even letting third-party AV sniff this work in process, that could let the cat out of the bag, by having the AV go check in with the mothership and go, hey, we just discovered somebody working on a new zero-day. So I could see they, like, stayed offline, developed this in a cleanroom environment, knew that none of the OSes they had were patched against it, but they didn't know about the rest of the world. And so the decision was, rather than tip off

individual AV, let's just wait till we're ready, put a null payload in, send it to VirusTotal, see if any bells go off. And, if not, immediately take the weaponized version and launch it at our target. So as I said, such is the world we're in these days.

Leo: Really interesting, yeah.

Steve: Yeah. And again, there is a response time window, and spearphishing has, what a few hours of response. So it's very likely, if that interpretation of the facts is correct, this strongly suggests that someone somewhere was probably a victim since not a single alarm went off at the time. And if that person had reason to open the PDF, that system probably got compromised.

Leo: Wow.

Steve: And somebody's mission got accomplished.

Leo: Now, it's all fixed now, though; right?

Steve: Yes, yes.

Leo: Okay.

Steve: At the beginning of May is when both of these zero-days were addressed.

Leo: Why 60 days to fix it? Why does it take so long?

Steve: Well, you know, we've seen instances, like where Microsoft, remember, was told of something in January, and then they said - they finally, after 90 days, came back to, it wasn't Google, it was - we were talking about it just a couple weeks ago. Shoot. The Zero-Day Initiative guys gave them 90 days. And then so finally Microsoft says, oh, we were unable to duplicate the proof of concept. And the Zero-Day Initiative guys said, well, we sent you a proof of concept three months ago. And Microsoft said oh, yeah, really? And they said, well, here it is again. And so they gave them a couple more weeks and then went public with it because it was like, hey, you guys have used up your time. So I just think everybody's probably busy, and they're having to juggle these things, and it took 60 days.

Leo: Well, even if it takes two days, it's enough.

Steve: Yes, good point.

Leo: Right, yeah.

Steve: Good point. Okay. So a group of researchers took another hard long look at the LTE cellular network. This is of course 4G that we're all using. LTE stands for Long-Term Evolution.

Leo: In other words, nothing.

Steve: Or we're still waiting for it, yes. We're still waiting. And that of course replaces the 3G, which was GSM, which stood for Global System for Mobile. So these guys have basically - they've just released a research paper which pounds yet another nail in the already well-nailed coffin of LTE. They'll be delivering the paper at the IEEE Symposium on Security and Privacy in 2019, so at least half a year from now. But it's available today, titled "Breaking LTE on Layer 2." I'll talk about layers in a second. But I've got a link in the show notes to the whole paper, which is very interesting for anyone who wants to get into more details than I will as appropriate on the podcast.

But in their abstract they said: "Long Term Evolution (LTE) is the latest mobile communication standard and has a pivotal role in our information society." Yeah, like we're all using it. "LTE combines performance goals with modern security mechanisms and serves casual use cases as well as critical infrastructure and public safety communications." That's building the case for why what they've done is important. "Both scenarios," they write, "are demanding towards a resilient and secure specification and implementation of LTE, as outages and open attack vectors potentially lead to severe risks." Previous work on LTE protocol security identified crucial attack vectors for both the physical layer, which is layer one, and the network layer, layer three. The one in between, the data link layer, layer two, they said, has remained a blind spot in existing LTE security research. And thus that was their focus.

Now, just to remind people, one of the neatest innovations in networking, I guess kind of not really to apology, but networking thinking or design architecture, was the separation of communications into well-defined network layers. Rather than just - previous to this you'd just sort of write something that was just a big blob that did what you wanted it to, but it wasn't compatible with anybody else's blobs. It was, as being a big blob, if you needed to do something different, you'd have to rewrite who knows what parts of the blob because it was just a big blob.

So layering solved that by deliberately - and there's something known as the OSI network layer model. What is that? Open Standard something, Initiative or something, OSI. So, for example, at the bottom layer is the physical, or like the electrical layer. So, for example, in the old days RS-232, the old serial communications, the specification for it being plus or minus 12 volts, and input on pin 2 and output on pin 3 and ground on pin 5 and the various, the electrical level spec would have been its layer one.

Also, for example, 10BaseT coax, the way the signal frequencies are pushed through the coaxial fiber would have been, for coax networking, layer one. In our modern RJ45 Ethernet we've got the way differential electrical signals on a twisted pair go back and forth as layer one. So that describes the electrical layer. Then for us in modern networking, on top of that we have Ethernet, which we've talked about often on the podcast, things like the way you deal with packet collisions, that is, so you go to the data, the raw data living on top of layer one, on the electrical layer. And that's where, for example, you say, okay, we're going to have these things called packets, and there's going to be a MAC address, which is the two 24-bit or the 48-bit address that's globally unique for the network interface card. And that's going to be a packet that has a payload. But that's as far as you go.

That is, so the idea is that, by deliberately constricting what you specify at each layer, you create interlayer boundaries that give you sort of a plug-and-play architecture. You can, instead of everything just being, like the whole solution just being a blog that's one chunk of code, and god help you if you want to change it so it runs on something different, instead by deliberately constraining it to a layered architecture, you're able to change things around. So layer three is the network layer, where for example in our networking today we have IP packets, or ARP packets, or ICMP. Those are three different protocols which can all be carried by the data link layer, the Ethernet packet, which gets around the room using the physical layer, the first layer, and so on. And so on top of IP, ARP, and ICMP, then you'd have the transport layer where you have TCP, UDP, and sort of that SSL and TLS, which are kind of grafted on at the same layer.

So anyway, what these guys have done, and the reason I've gone into this detail, is the ultimate solution to this and other problems is where we need to talk in a second. But so there has been previous work at attacking LTE's physical layer, that is, the radio link portion, and the network layer, the protocol that runs on top, but not the data link layer that is the equivalent of essentially the encryption of the radio signals themselves.

So they say they've got a number of different attacks. The first is kind of a yawner. They said: "In this paper we present a comprehensive layer two security analysis and identify three attack vectors. These attacks impair the confidentiality and/or privacy of LTE communication." And I should say I'm sharing the yawner here first. But the one they found is horrifying. We'll get there in a second.

"More specifically, we first present a passive identity mapping attack that matches volatile radio identities to longer lasting network identities, enabling us to identify users within a cell and serving as a steppingstone for follow-up attacks. We demonstrate how a passive attacker can abuse the resource allocation as a side channel to perform website fingerprinting that enables the attacker to learn the websites a user accessed."

So as I was digging into this paper, I was thinking, what? Okay. We know about side-channel attacks. And so how are you going to fingerprint a website if you cannot decrypt the data? Because they don't arrange to do that. If that were possible, we would probably know about it already. So it turns out it's sort of weak and obvious in retrospect. In describing it, they say: "Meta information on the data link layer leak information about the consumption of data per unit time." In other words, like bandwidth flow. And they say, for example, and they have a made-up person Bob, who's their victim: "For example, if Bob watches a video, he uses more traffic compared to when he accesses a simple website. As a preparation step of the attack, Eve records popular websites and their layer two patterns." Thus their term "fingerprinting."

"During the attack, she eavesdrops the meta information and looks for similar patterns. In case she finds a match, she knows which website the victim visited with a certain probability." So it's like, okay. So they're basically saying you've got radio, so eavesdropping is possible. So different websites, if you look at them with sufficient focus and clarity and granularity, they will have characteristic bandwidth signatures. And so, if you built up a catalog of those and called them "fingerprints," then you could see somebody else's traffic doing exactly the same thing that your analytical traffic had earlier done and conclude with some degree of confidence that they were at the same place, even though you have no visibility into the data of their traffic at all, just the fact of their traffic and the amount and timing and so forth.

So it's like, okay. Again, sort of obvious. Not very significant. However, the main attack is potent. They called it the "aLTEr" attack, and they played with the fact that the middle three letters of "aLTEr" are LTE, so aLTEr. They said: "We present the aLTEr attack that exploits the fact that LTE user data is encrypted in counter mode, but not integrity protected, which allows us to modify the message payload." And we talked about this a

long time ago, and I remember at the time sort of shaking my head. It's like, how did this ever happen? And it's just that it's so old. You know, this spec has been around for so long that I guess we forgive them for not authenticating the communications. We'll talk about that in a second. I'll just finish what they said.

They said: "As a proof-of-concept demonstration" - and they do this - "we show how an attacker can redirect DNS requests and then perform a DNS spoofing attack. As a result, the user is redirected to a malicious website. Our experimental analysis demonstrates the real-world applicability of all three attacks" - the first two I lumped together because, again, they weren't very powerful - "and emphasizes the threat of open attack vectors on LTE layer two protocols."

Okay. So here's what's going on with that. LTE uses a useful but flawed encryption. AES counter mode, you know, everyone gets wound up when you say "AES," like oh, that's a good cipher. Yes, except that, as we know, it takes more than just the cipher to deliver useful encryption. AES counter mode uses a 128-bit counter, which is incremented one step at a time. The value of the counter is encrypted by the AES cipher under a key. And the output is a pseudorandom bitstream. So we've dealt with these before. This was a bad weakness in the early WiFi days, when we had the WEP protocol, and RC4 was our cipher. RC4 was a very cool, developed by RSA, proprietary, although you can't keep these things secret, cipher, which economically generated a pseudorandom bitstream. It suffered from an initialization problem that it hadn't sufficiently randomized itself when it was starting up and so had problems there. Thank you, Wi-Fi Alliance.

AES counter mode has a different problem, and that is that the AES cipher driven by a counter beautifully produces an unpredictable, tremendously good pseudorandom bitstream. The problem is we then XOR it with our data, that is, with the plaintext, in order to get the ciphertext. Now, we know that, sort of contrary to intuition though it is, if you simply XOR plaintext with random noise, which is for all intents and purposes what pseudorandom data is, what you get out is random noise. And remember that what the XOR does, you can think of it as conditional bit-flipping, that is, where there are one bits in the bitstream, those bits of the plaintext are inverted. And though it doesn't seem like that's powerful encryption, it turns out there's no way to decrypt it. I mean, it's really true that there's no way to decrypt it. So doubtless the people who are doing this thought that, wow, this is really great encryption because there's no way to decrypt this.

What happens, of course, at the other end is that, if the other end has the same counter, so that it's synchronized, and the same key so that the same cipher is encrypting the same counter, it will be able to generate the same bitstream. So it will produce the same one bits in the bitstream which will reflip the bits that were flipped during the encryption back, and it becomes decrypted. So, yay, you're back to plaintext.

Here's the rub. If you know what it is that you're encrypting, then you're able to change the message that is sent. Think about this. If you know what the plaintext is, and you have the cipher text, if you XOR the known plaintext and the cipher text, you get back the bitstream. That's a weakness with this XOR in this situation.

So it turns out that DNS queries have a fixed known packet format. We know in a DNS query exactly where the IP address is in the DNS packet. That allows us to take the DNS that the user is using, XOR that IP address with the malicious IP address of a spoofed server, and alter the traffic without, I mean, technically we can decrypt it, but we don't have to decrypt it. Because we know what it is, even though we have a bitstream that we can't predict, because it's just a simple XOR, that portion can be known, which allows us to change the IP address of all DNS queries in LTE traffic that we cannot otherwise decrypt and route a victim to our own DNS server to perform a DNS spoofing attack. And they demonstrate it, and they show it, and it is absolutely a real-world vulnerability.

The good news is it's not passive. That is, it does require a man-in-the-middle position, so like the fake cell tower, the stronger signal cell tower style, the Stingrays. And of course we all remember, Leo, how many more "cell towers" appeared in Las Vegas in the environment of the Convention Center.

Leo: And Washington, D.C. Yeah.

Steve: Exactly, during the conference. So it's no one's imagination that, like, fake cell sites exist. And that gives you a man-in-the-middle position. Now, here's the problem is that there's no authentication. The actual packet has been changed, and this attack is point to point, that is, it's an attack only on the radio link portion, thus LTE. So somebody has to position themselves so that they look like the cell tower to receive this information, alter the data, and then send it on. They're unable to decrypt the whole message because they don't know everything else in there. They're just able to change the IP address or, well, actually they can modify any known data in a known position in any of the packets. But practically, these guys just demonstrated this as a DNS spoof attack. But the problem is, since they can't change anything else, they have to forward the otherwise unchanged packets unchanged to the actual cell tower.

Now, the problem is LTE lacks authentication, meaning that there is no way for the recipient to verify that there hasn't been some alteration of the data in transit, that is, since the sender emitted the packet, there's just no way to detect any changes. We've talked about authenticating and encrypting, how one without the other is, I mean, it provides something, but it doesn't provide what you think. And it always is necessary to encrypt before you authenticate because that way the recipient authenticates, verifies the authentication before they decrypt.

We have encountered attacks where that was wrong. It was backwards in some of the early versions of SSL. That's one of the things they got wrong. They authenticated, then they encrypted. And what that allowed was it allowed bad guys to probe the encryption by having the authentication fail after a test decryption by the recipient. And you don't want to allow that. So the reason you encrypt, then you authenticate before sending is that the moment authentication fails, you fail, that is, the verification at the receiving end fails. You immediately fail the entire transaction, thus providing no additional information to an attacker.

So where we are today is we do have a significant problem in the radio link integrity of LTE. The reason I talked about layers is that they're only able to change this one aspect. And the good news is the higher level layers provide us robust protection that, given that it's present, that is missing at the lower levels. We would like to have good protection all the way down to the bottom. In this case, due to its age and just a fundamental design problem - this was built in a different era, essentially - we now know that it's possible to induce DNS spoofing attacks, and no doubt other clever attacks because we know how clever bad guys can be. When they understand that they're able to change the data, change any known data in a known position in the radio link, there's going to be some mischief that follows.

Leo: All right. You said you were going to - you teased us.

Steve: Yes. So we have a fully feasible attack on LTE radio link. It's not passive. But as we know, no one has any idea what cell tower they're connected to.

Leo: That's right, yeah.

Steve: I mean, that's just sort of - you just, oh, look how strong my signal is. Yes, because there's an evil tower in the next room. You don't always want to have a strong signal. However - so, okay. So the problem is that the design of LTE is where the attack is. So that's not getting changed any time soon. I mean, nothing can change that. That's in all of the deployed cell towers everywhere, and in the baseband processor of our smart phones. So nothing's happening to change that.

The team of three researchers - they were from the Ruhr University in, looks like Bochum, B-O-C-H-U-M, Germany - and a researcher from New York University were the team that did this research. They dutifully notified the relevant institutions: the GSMA, which is the GSM Association; also 3GPP, the third-generation partnership project; and the various telephone companies about the issues they discovered. But not like any of them could do anything about it. This is a bug that cannot be patched. I mean, it's a fundamental flaw in the radio link protocol that we're all using. And as I said, it's built into every LTE device that we have right now. So it simply will not be fixed.

The problem is the researchers are also concerned about the future because it turns out, unless some sort of pressure is applied, it's very likely that we're going to see this problem persisting in the 5G standard. 5G is the successor. It's already being deployed by some large providers, at least in the U.S. I don't know where it stands internationally, but we're generally sort of behind things. So I wouldn't be surprised if other - I know that Verizon and AT&T have already begun implementing the 5G protocol.

Of course, 5G is backwards compatible, and therein lies the rub because that means that the newer reliability and security features, for example, 5G does provide authentication, but it's optional. And as we know from studying the evolution of SSL, you offer stronger protocol options, and it's nice to have them, but unless everybody knows about them and uses them, you don't get the benefit of them. And you've got the classic downgrade attack problem where - and this is something that a man in the middle could certainly do. That cell tower in the middle could pretend to be a 4G tower, even though there's a 5G tower further away, but its signal is weaker, so your device locks onto the stronger signal and says woohoo. Oh, but only 4G? Okay, fine. And so now you're open to that level of attack. And then of course that tower appears to be a 4G device to the 5G tower, which also downgrades itself to 4G. And so we've made no progress.

So there has been some research already to look into the 5G specification implementation, and it has been found wanting. Now, in a related technology, we've talked about the SS7. And I know the SS7, Signaling System 7, is the very old technology, decades old, which is what glues together different providers. And I know, Leo, on other podcasts I've heard you talking about SS7 and the problems that we know it has.

Leo: It just can't be fixed. That's the sad thing. There's just no way to fix it.

Steve: Exactly. It's deployed now. It's global. And it's in place. So there's a new system known as the Diameter system, which is the formal successor to SS7. Which, as with SS7, it has been designed to manage the internetwork handshaking. It's the way a Verizon customer can talk to an AT&T customer is that the Verizon and AT&T systems use at the moment SS7, in the future Diameter, in order to themselves interoperate in order to pass traffic back and forth. So that's the internetwork handshaking.

But the problem is Diameter is being deployed along with the 5G protocol and also appears to be fraught with problems. It does provide messaging encryption, which was completely absent from the design of SS7. But some security researchers have looked at the actual implementation and have found problems there, too. They said that Diameter misconfigurations that have been spotted are often unique to each network, but they repeat enough that they were able to group them into classes. There's subscriber information disclosure, network information disclosure, subscriber traffic interception, fraud, and denial of service. And apparently all of these problems that were known under SS7 continue to dog us under Diameter.

So this shouldn't be a surprise. Much as we wish this were not the case, if there's a lesson that keeps being learned on this podcast, it's how difficult it is to retire widely deployed technology that is functioning, even if it's not secure, or as secure as we wish it were. So what do we do? As I mentioned before, we fall back upon the upper layer, the encryption and endpoint authentication provided by TLS. Remember that DNS spoofing can take you to a malicious DNS server that can give you a malicious IP for a valid domain. But unless that domain has a certificate that your browser trusts, and if you are using HTTPS or some other network protocol over TLS, then the inherent authentication at the certificate level, at the higher level, will protect you from that spoofing.

So it's really looking like the move several years ago to we just have to have encryption everywhere, and it has to be authenticated encryption, and the only way we know to do that today is with certificates, so we've got to make certificates readily available. It's looking like the Let's Encrypt movement, and just the broader availability and awareness of the need for encryption which has become pervasive, has been a really good change to the industry. And in fact, because we're relying on upper layers to encrypt and authenticate, even if shenanigans are played down at the lower layer, it makes it much more difficult to pull off an attack.

The requirement to have a fraudulent TLS certificate, we know absolutely that anyone who controls a trusted certificate authority can selectively produce such certificates, which is to say certainly any state-level actor. There's no way that the U.S. federal government, the CIA or the NSA, probably even the FBI under certain circumstances, I mean, technically there's no way they don't have the ability to produce a certificate if they want to which is currently trusted by all our browsers.

The same is certainly true of Russia and China, any of the major players who have control over a certificate authority trusted by our browsers because, as we know, the weakness of the current model is that we need to trust, in order for connectivity to work, we need to trust any of the certificates signed by any of the CAs that we trust in order to establish secure connections. So it's not a perfect system. But at least it moves it probably above the level where a casual attacker is able to spoof a website by intercepting traffic and redirecting someone to a fake Google or to a Facebook or something where TLS encryption with a known certificate is in place.

So it's looking like we're going to end up relying on the higher layers. And that takes me into what will probably be next week's topic. It's too big to add to everything else I want to talk about this week, but the EFF has launched another encryption initiative which is known as STARTTLS Everywhere. STARTTLS is an optional protocol which email servers, that is, SMTP servers, are able to advertise their support of. So traditional email has, as we've often said, has never had encryption bound to it. Port 25 is the SMTP port where email servers talk to each other in order to exchange and forward email on behalf of their clients; port 110 is sort of the same for client to server, POP; and 143 client to server over the IMAP protocol. There are now the equivalent, in the same way that 80 is unencrypted web and 443 is encrypted web, there are encrypted ports defined for email, which I'm going to get into in detail next week.

But STARTTLS is an earlier solution which has been moving forward, but hasn't had as much push as it could receive. I think something like 89% of the current email servers on the 'Net support STARTTLS. And what this is, it's a port 25 resident protocol. So it runs over a nominally nonencrypted connection, where over port 25 one SMTP server connects to another, and in the Hello message from the server connected to, it advertises whether it supports STARTTLS. And what that is, is essentially it is a mechanism for upgrading the unencrypted dialogue the servers are about to have to one which is encrypted. And as its name suggests, STARTTLS Everywhere, this is the EFF's initiative, which we'll go into in detail next week, to move this; to solve some of the remaining problems behind the deployment of STARTTLS.

But again, way up at a much higher level, this allows then not only clients to obtain encryption, for example, if you're using Gmail, you're already using HTTPS in order to do your webmail interchange between client and server. But where any of the endpoints in the hops between the source and the destination are not secure, that transit of email can be unencrypted and viewed visibly, and there's no question that there has been a large sucking sound made by the NSA at all of their nodes where they're grabbing traffic. Once upon a time it was HTTP that was almost never encrypted, except during logins. We remember those days. We were talking about them here. And it's decreasingly the case that email is not encrypted. On the other hand, email downgrade attacks are possible because STARTTLS is still optional.

So anyway, we'll go into this in much greater detail next week, when we get into what exactly STARTTLS Everywhere is all about. And I salute the EFF again for another probably, well, certainly useful initiative. Yeah, very cool.

Leo: Yeah.

Steve: Okay. So as I was looking at some other research that Aleph Security has produced, I was thinking it's good that the term "mitigation" was used about Spectre and Meltdown. No one said "prevention." They didn't say "solution." They said "mitigation." Because that sort of says we made it harder. Which it turns out is all that was done by the browsers. In Edge, Chrome, and Safari, the Spectre and Meltdown mitigations have been partially defeated. The research by these guys has demonstrated that, except for Firefox, which incorporated a different mitigation, which they haven't yet defeated, all the other browsers have, that is, the mitigations that were put in place have had the effect of slowing down the rate at which protected data can be acquired through inference using these attacks in browsers, but have not eliminated them.

We talked last week in the context of WebAssembly about the disabling of the SharedArrayBuffer. There's also something known as "index masking of array objects" is another mitigation; site isolation feature in Chromium-based browsers; reducing the precision of the performance.now timers; and adding jitter to the response of the performance.now API, the idea being make it less certain what time it is. And since all of this, as I've described several times in the past, is about detecting a difference in performance based on the history of the processor's execution, thus what Spectre is trying to protect us from, it turns out that all that was done was a slowdown was introduced.

Their paper, or their blog posting, is titled "Overcoming (Some) Spectre Browser Mitigations." And basically they said: "Because of this vulnerability discovery" - and I snipped a bunch of the top out where it just talked about stuff we all well know about this. So in the third paragraph they said: "Because of this vulnerability discovery, browser vendors implemented different mitigations for this vulnerability. Some of them are meant to disable known methods of querying CPU cache state of memory slots, i.e.,

JavaScript variables. These mitigations include the resolution reduction of the JavaScript timer performance.now and adding jitter to its results.

"In our research we were able to overcome the cache access timing specific mitigations. Although these mitigations cause a serious slowdown in our proof of concept, they are not effective in preventing this attack. These mitigations also have some negative performance implications and are hurting the functionality of some legitimate Javascript web application use cases."

In other words, these guys took the position that, okay, let's see how much our access to private information is reduced by these mitigations. And what they found, almost across the board, was they are able to determine one bit per second of secret information. So it's not a lot; but if a malicious page or malicious JavaScript is allowed to run over time for some period, basically what's been done is the bandwidth channel has been constrained with the addition of this jitter, but that's all it's been. It hasn't robustly solved the problem, which is why I was saying at the beginning of this that, as I was thinking about this, it's a good thing that we're using the term "mitigation" because we mitigated, but we did not solve or eliminate the problems. I mean, they are still fundamentally there.

And in some instances it may well be that one bit per second, although it would take two minutes to get 128 bits, that's a key these days. So if you know which 128 bits you want in protected memory, it only takes two minutes to get it. Their position is, the position of these developers is, given that these changes are ultimately ineffective in browsers, except Firefox, and I didn't have a chance to dig into what it is that Mozilla did differently, but it looks like maybe Mozilla did something right, or maybe the performance impact is more onerous in what Mozilla did for Firefox. I don't know. But their position is, given that all that's been done is the rate at which bits can be obtained has been constrained, and the cost is relatively high in terms of its hurting the performance of specific Java application uses, maybe it makes more sense to back these out and look for some other solution. I don't know.

But anyway, I just wanted to bring up the fact that there was this interesting piece of work. There's proof of concept. They've got their code up on GitHub, and it's able to extract a bit per second from private memory on a browser which is trying to protect against this. And so, as we've said from the beginning of this year, this is a big problem.

So I've talked a couple times about the increasing use of Universal Plug and Play exposed to the WAN side. Crazy as it is, there is a large population of browsers exposing their UPnP interface. And we first talked about this years ago, and I immediately added a test to ShieldsUP! at GRC, our listeners who've been around for a long time will remember, because I wanted to, as quickly as I could, let anybody know if their port 1900, that's the SSDP port, if port 1900 on their router is publicly exposed. And what is it, 53,000 some, I don't remember now what the count was. I do show the count, if you go to GRC.com, ShieldsUP!, and then do the UPnP test. It'll first of all tell you probably that your port is closed. If you're a listener of this podcast, you've done this already. Certainly you want to make sure.

So in any of the DDoS attacks which reflect off of servers - and we know that many of them do. For example, DNS amplification attacks are popular. You spoof your source IP and make a DNS query to a publicly accessible DNS server, asking it a question that generates a large reply. And it will send that large DNS reply, it thinks back to you. But because you have put your victim's IP address as the source of the query, it sends that big response to your victim. Thus it's participating in a DDoS flood.

So what happens at the victim side is a torrential flood of traffic from DNS servers. Innocent themselves DNS servers all over the Internet come pouring in. What's

significant is they all come in from port 53. That is, they come in - because it's a query, DNS is on port 53 - they come in from all of those DNS servers' port 53s. So the first thing somebody who's mitigating DNS attacks will do is block all traffic coming in on port 53. It's simple.

Now, if you've blocked all traffic on port 53, then legitimate users of that server or that site, that is, that site itself or somebody in the network would not be able to get DNS from their own external DNS servers. So first of all, they might not have external DNS. They might have internal DNS. But you could whitelist port 53 only for a couple IPs, that is the legitimate DNS servers that are probably not going to be attacking their own network, just probabilistically, but block all the others.

The point is that, from a network plumbing standpoint, blocking a port is simple to do. It just doesn't take any time at all. It's a single rule in a router: Drop all traffic incoming from any IP with a source port of 53. And bang, that attack is blocked. The bandwidth is still hitting that router very hard. But if you're a large ISP, and you've got a distributed perimeter with lots of connections to the Internet, then the individual places where it's entering your network, those bandwidths are smaller, and at least you're preventing them from getting inside. And the same thing is true, for example, of NTP, the Network Time Protocol. That is another popular source of DNS reflection attacks. Again, you generate a query. You ask what time is it, and you pretend to be your victim's IP, and the network time server blasts its response back to the person who requests it.

So the point is, port-based filtering, as it's called, is a simple and potent solution for DDoS. And it's what most of these DDoS sites offer is they know that there's no reason to have a torrential input of DNS queries. So either they will quickly block DNS if it's under attack, or maybe they just leave a block up all the time so that foreign DNS incoming queries are blocked because there's no reason that they should ever be valid coming into that network. So you can see where any technology which allows the source port of an attack to be other than DNS or NTP or any of the valid services is powerful. And even better would be if every different stream looked like it was coming from a different port.

And so what's happened over time, and again, entirely foreseeable, but it went from oops, this could be bad, to oh my god, DDoS attacks are now far harder to block than they were. Because it doesn't take long for the bad guys to figure out how to leverage open UPnP ports for their own purposes. And as we were saying before, back when this was a theory, we were talking about it as a means of masking traffic in like the sci-fi bounce around seven different nodes around the world before sending your traffic somewhere. That is, it was possible to expose Universal Plug and Play to not only map the traffic back into the network behind the Universal Plug and Play-equipped router, but to map it to another public IP on the WAN side, outside the network.

Well, DDoS attack tools have now matured. They are now corralling and bringing to use all of the Universal Plug and Play exposed routers to randomize the source ports of their attack traffic so that what DDoS mitigation providers and services are now seeing are significant floods which are significantly more difficult to block. Their primary previous tool of blocking on source port because these were server reflection attacks is now, to the degree that Universal Plug and Play routers exist - and it looks like there are many, many tens of thousands of them, and unfortunately probably new ones coming online all the time. Once they're found, they get grabbed up by a DDoS attacker and then used as a reflection point for traffic which has the ability to randomize the source port, making DDoS attacks way more difficult, significantly more difficult to block.

And my last bit of coverage, I wanted to mention WordPress. They are a sponsor of the TWiT Network, so full disclosure there. But this doesn't really impinge upon that. We were just talking last week about path traversal bugs. And it turns out that all versions of

WordPress have a known path traversal bug vulnerability, but one which apparently doesn't have the WordPress devs very worried since they've known about it since last November. So I saw a bunch of hair-on-fire stuff last week, and I just wanted to explain to our listeners, if anyone was wondering or worried, exactly what is going on with this.

WordPress, as I mentioned, was notified of this last November by security researchers who made the discovery. And I imagine it'll be fixed at some future point when WordPress does another rev of their work, if they choose to. I would be surprised if WordPress hasn't been revised in nine months. But this problem is still there. So maybe it fell through the cracks. In any event, there is no chance that it could be used for a widespread attack, which is probably why it hasn't gotten more attention.

The flaw was discovered in the PHP functions that are used to delete thumbnails for images uploaded to a WordPress site. What the researchers discovered was that users who already have access to a site's posting editor and can thereby upload or delete images and their corresponding thumbnails have the ability to insert code into the site which could delete crucial files which are part of the WordPress content management system core. Which is something that should not be possible without direct access to the server's FTP login or the server itself. So all that can be done is that some files that should not be deletable can be, and only by somebody who already has credentialed login capabilities to the site.

So maybe, and this is what's been postulated, it could be a limited form of privilege elevation attack, although it's not immediately clear how useful deleting other files would be. What's been suggested, again, sort of far-fetched, is that a site could be hijacked by deleting the main wp-config.php file, which is the site's main configuration file. If that were done, then it might be possible to reinitiate the installation process and install the site using a different set of database settings to essentially hijack the domain and deliver custom or malicious content.

But again, this is like, okay, the WordPress devs have understood that this problem exists since November. It requires that you already have the ability to post and delete and that with that you then leverage this vulnerability in order to make it happen. It wasn't clear to me yet that even now there were like a proof of concept or it had been made public. However, the guys who discovered it, if this is a concern to anyone, there is a hot fix which they have produced which can be downloaded and added to the functions.php file on the site's active theme in order to prevent this.

But anyway, I wanted to - the headlines were, oh my god, any WordPress site can be taken over. It must be that, since the devs know about this, there's no way that it could be mass exploited, and it seems hard to imagine that it's going to be useful to anybody who has access to the WordPress site itself anyway. And I'm pretty sure it'll be fixed as soon as WordPress gets around to it. And I looked for some comment from WordPress, but I was unable to find anything either way, although that shouldn't be considered authoritative.

I did get an interesting note from Ben in Greensboro, North Carolina that discusses an aspect of SpinRite that we've touched on, but not deeply, which caused some confusion for him. He said: "SpinRite recovers bad sectors, but afterwards SMART status shows 'good'?" He says: "Hi, Steve. Long-time Security Now! listener and SpinRite user/abuser." I'm not sure how you abuse SpinRite, but okay. He says: "I recently came across a situation I found odd. A client's machine was having issues staying booted and BSODing." So we know that's Blue Screen of Deathing. "So, naturally, the first thing I did was run SpinRite at Level 2 on the Windows partition, and it recovered SEVERAL" - he has in all caps - "bad sectors," he says, "which typically tells me that the drive is going bad." Ah.

Anyway, he says: "Of course this fixed the trouble, and the system then worked flawlessly. But since the machine was still under warranty, I figured that I might as well get the drive replaced. So I contacted the manufacturer's tech support, and while on the phone it passed their built-in drive diagnostic, Windows WMI SMART check, and all tests showed the drive as good." He says: "Usually when SpinRite recovers sectors for a machine that won't boot, it will fail any subsequent SMART checks. But not this time."

He says: "Do I have a fundamental misunderstanding of what recovery of a sector means?" He says: "I'm currently running a second Level 2 scan on the whole C drive to verify it's operational before giving it back to the client, and the ticket with the manufacturer temporarily is still open," he says, "just in case I come across something."

So, yeah. We sort of touched on this a couple weeks ago, where we had - it was that laptop that was being bounced on the coffee table by somebody whose wife was using it for, I think, Photoshop or something. And he ran SpinRite to fix the problems, and then as he saw the laptop bouncing on the coffee table as she dragged it over to show it to him, he thought, well, I'm going to probably have to run SpinRite again.

First of all, drives are very resilient. They obtain the resilience through having a large pool of spares. And sparing out defective sectors is just now something drives do in the regular course of business. One of the ways SMART can cause an error or SMART can show that the drive is in trouble is if the error rate starts going up to a point where the drive thinks, okay, something is not happy here somewhere. So that can happen. Another way that the SMART system can show trouble is if the pool of available spares starts being consumed at a high rate, or if the number of available spares starts approaching zero.

So I guess the point is that drives are surprisingly dynamic. And that's probably, I mean, that's certainly one of the things that SpinRite works with when it's working with drives, is it's watching the SMART status while it's running. It's learning about the drive. It's watching the rate at which error corrections occur and the rate at which sectors are being spared out.

In the case of recovering bad sectors, the sector is bad only inasmuch as the drive was unable to read it normally. SpinRite read it, fixed it, the drive replaced it with a new one, and the drive's kind of fine now. So as long as the errors are not occurring too often to upset the drive, which it would then report in the SMART status, and as long as there are still sufficient spares for SpinRite to sort of do this kind of work, or the drive maybe to do them on its own if it's able to, then we're sort of in this elastic phase of the drive where, yeah, it's not perfect, but with the density that we're storing data on drives these days, no drives are. And so everything's okay.

So I would say, Ben, if you run that second pass, and I imagine that Level 2 showed everything was fine, since SpinRite fixed it the first time you ran it, and SMART still says everything's good, the manufacturer's going to say, "We don't see any problems here." I think you're probably good to go. It's just probably the fact that somehow a couple sectors suffered some damage. But the truth of today's drives is that's kind of happening all the time. SpinRite works hand in hand to maintain them.

And I often see, I don't talk about it because it's sort of a non-story, but I often have people writing, saying, you know, I don't have any SpinRite miracles to report because I run it periodically for maintenance. And that's what happens is SpinRite is able to help the drive see these problems before they become critical. And so it sort of enhances the drive's built-in resilience and keeps any problems from actually biting anyone. So that's a good thing. And that's SpinRite.

Leo: Indeed it is.

Steve: And with any luck, WPA3 will help us do that better.

Leo: Maybe.

Steve: Yeah.

Leo: But is there something wrong with WPA2?

Steve: Oh, yeah. There is a known password attack known as the KRACK attack. If somebody sniffs your authenticating to an access point, they can then perform an offline brute-force attack. So consequently the strength of your password to the access point is important. And so in various settings where maybe the password is sort of predictable or guessable, like it's the car dealership or a company that may have their name or address or something in the password, there is some vulnerability. So there is a known problem with WPA2. But the good news is WPA3 fixes not only that, but offers some new features which we've been wanting, everybody's been wanting for a long time.

So it's been 15 years since WPA2 arrived back in 2004. So that's been around a while. And you might say that's 14 years, since it's 2018, except that we're not actually going to get WPA3 till late next year. So, okay, it will be 15 years by the time we actually get WPA3. It's unclear whether, just because of the Wi-Fi Alliance and their stickers and their certification programs, whether existing hardware will be upgradeable. We have to imagine that our smartphones, like Apple and Android and widely deployed devices, will be upgradeable. This probably requires the baseband processor to be upgraded, which I assume can be done over the air by the phone provider.

But it's not clear whether we're all going to have to get new WiFi routers if we want to use this, or whether a firmware update would be able to bring these features to us. Again, this is all licensing based and certification based, and the Wi-Fi Alliance is the Wi-Fi Alliance. And as I said, I got excited this morning by the idea that they had published the specs, and it turns out it was just an absolute tease. So it was like, okay, fine.

So it is of course backward compatible so that any device that isn't WPA3 will be able to fall back to WPA2 functionality. There are two flavors. There is WPA3 Personal and WPA3 Enterprise. The Personal brings better protections to individuals by, as I mentioned, and as you prompted me for, Leo, providing more robust password-based authentication. It's known that there is a weak password authentication problem. WPA3 uses something known as SAE, Simultaneous Authentication of Equals, which replaces the longstanding PSK that we've talked about and pounded on for years, the Preshared Key approach, which was built into WPA2. So this SAE, this Simultaneous Authentication of Equals technology, is resistant to offline dictionary attacks which the KRACK attack makes possible under WPA2. So that strengthens WiFi.

And again, I don't have to tell everybody that the world has just become WiFi. I mean, it's incredible to me. Once upon a time it was do you have WiFi? Now it's like, well, do you have a cell phone? It's like they just - they don't ask you. They say, "What's your mobile number?" So it's like, "Yes, what's your WiFi?" So it's just ubiquitous, obviously. So it was the KRACK attack, K-R-A-C-K, which stood for Key Reinstallation Attack. And at the time we covered it on the podcast extensively. So if anyone is interested in a refresher, you can find the podcast [SN-633], KRACK, K-R-A-C-K, which is an attack on

WPA2 authentication, which allowed someone sniffing the transaction to grab it. And remember also that it was possible to deauthenticate anyone by sending a deauthenticate packet to the access point, which would force a reauthentication, allowing an attacker to capture the authentication traffic if they showed up too late otherwise, and then launch the offline attack.

So a determined attacker who is either passive and patient or active would be able to, given the complexity of the password - and WiFi passwords probably suffer from a lack of sufficient complexity just because people don't think that's important enough. That could be a problem. So the Wi-Fi Alliance brags that this WPA3 allows users to choose passwords that are easier to remember because a traffic analysis attack which WPA2 does suffer from would not function, would not succeed under WPA3.

Oh, and WPA3 also gives us perfect forward secrecy. So we know what that means. That means that traffic captured now cannot be later decrypted if in the future the key is determined. So perfect forward secrecy is another good feature of encryption which arguably all modern crypto should have. And when we have WPA3, we'll have it in our WiFi for the first time. We don't have that now. We have that in the higher level links in TLS, so it's good to have it there, but not at the WiFi link.

There's also an Enterprise version of WPA3, as there is sort of nominally for WPA2, which increases the encryption key strength. And it's not clear to me why they just don't have that in WPA3 Personal. But I guess they want to have a higher class grade. In their scant coverage of this, they said that WPA3 Enterprise offers an optional mode using 192-bit minimum-strength security protocols to better protect sensitive data. And it uses just larger communications protocols. Presumably it requires a little more oomph at each of the endpoints, and so it may not be something that a weak IoT device wants to or needs to deploy. And again, it's not clear to me that there's a compelling need. But of course the Enterprise WPA offers Kerberos key negotiation and lots more fancy features that the typical end user just doesn't need.

However, in terms of good features, we've got a couple things. Top of the list, I think, well, there's two. We now encrypt opportunistically open WiFi networks, which is like, yay. Why has this taken this long? We've often talked about it. I've lamented its lack because it is so easy to do. I was talking about it just a couple weeks ago, how in a world where we have Diffie-Hellman encryption, even though you don't have authentication, which means you're still susceptible to a man in the middle, with Diffie-Hellman key exchange - that's what I meant to say, sorry, Diffie-Hellman key exchange - in plain sight, endpoints can establish a secret that no passive observer can determine.

So for a long time there's been no need for unencrypted WiFi. You could have WiFi where you walk into any coffee shop, or car dealership waiting for your car, or airport, and it's like free WiFi, which is also per connection encrypted. But we haven't had that because the Wi-Fi Alliance didn't give it to us until now, and until next year, late next year. So there is something known as OWE, Opportunistic Wireless Encryption, which is defined by the IETF. It's an IETF-defined RFC 8110, which the Wi-Fi Alliance has adopted and will be part of WPA3.

So unfortunately, it will require the retirement or the upgrading of each endpoint for us to have this, but it does mean that, in the absence of an active man-in-the-middle attack, because remember that Diffie-Hellman key exchange doesn't authenticate, it allows - as long as you know who you're talking to, thus the authentication component, it allows a visible exchange of data to establish a secret between two parties. But if somebody gets in the middle and pretends to be the other person to each end, then that person ends up establishing secrets with each end and is able to decrypt the traffic moving through. But still, much better than nothing.

And the last new feature of WPA3 is what they call Wi-Fi Easy Connect, which they're not describing, and I hoped to get it, but it was one of those I got the seven-page teaser with acronyms defined, okay, thank you. They did say that it uses visible QR codes printed on access points and IoT devices, and that it uses public key crypto. So we can pretty much reverse-engineer what it is from that. What that will mean is that a router will have a factory-set asymmetric private key burned into its firmware with the public key shown as a QR code on its exterior label. And the same will be true of an IoT device. It'll come with a per-device private key, and printed on it will be a little QR code, which is the matching public key.

And as we know, just that allows us to solve this problem. It allows a device with a camera to obtain the public key for which the matching private key is only known to the device on which that public key is printed. The public key would allow the device seeing the QR code to generate a random number, we'll call that the ephemeral key, encrypt that under the public key, which can then only be decrypted by the device's secret private key. So it allows the secure establishment of a temporary link because it doesn't matter if somebody even in this case is a man in the middle. A man-in-the-middle attacker can't change anything or has no way of decrypting it.

Essentially, you can think of the QR codes as optical channel out-of-band information exchange. It's out of band in that it's just sitting there as a label, and if you don't have optical access to it, then you have no way of knowing what the device's public key is. So they don't explain how this works in detail. They talk about a device like a smartphone being used for a device that has a limited user interface. So a router has a very limited UI. And a light bulb or an IoT device, a burglar alarm switch on the door or something, very limited UI.

So the idea would be you just scan the QR code with your phone. It then participates in this interchange in order to get the devices set up in a way that is - now, and people have been setting up IoT devices using a temporary hotspot in a smartphone for a while. The problem is, it is vulnerable to an attacker who is present watching that process occur. So there's a window of vulnerability. That's what this eliminates. And it also means that just moving forward, anytime you want to establish a secure association, if there's an optical channel that allows that to happen, it can be done with true security.

So they're calling it Easy Connect. We know from having discussed WPS, that was the previous easy way of connecting that ended up being horribly flawed because it turns out it was an eight-digit code that could be chopped into a four-digit and a three-digit because the last digit was calculable, was just a check digit. And it allowed the whole thing to be brute-forced very easily. So we're going to hope that the Wi-Fi Alliance has not made any similar mistakes. In any event, we're getting a bunch of welcome features. We'll have to wait another year and a half, and then it will require that hardware be changed and updated.

But as that happens over time, I mean, given that we were with this last WPA2 spec for 15 years, there'll certainly be time for us to be obsoleting and replacing devices with WPA3. And so we just get more security. And again, as with the case with LTE, we would like these radio links to be secure. But to the degree that they are simply the carriers of the lower layer traffic of sessions which are themselves much more secure and authenticated, we could argue that, so long as what they are carrying is providing for its own security, then the worst that can be done is a denial of service attack, or things break for some reason. But there's no active vulnerability or lack of privacy or information disclosure.

So anyway, that's WPA3. I'm sure we'll know much more about it. I hope we know more about it as we get much closer. It'll be about a year and a half from now. So I'm sure we could calculate which episode of Security Now! that will be.

Leo: And it comes in conjunction with 802.11ax; right? It doesn't require it, but I know that that's part of the improved spec for ax.

Steve: Yeah.

Leo: Which also is a year and a half out.

Steve: Yeah.

Leo: Okay. Thank you, Steverino. We now know everything we ever would want to know about WPA3. Well, no, that's not true at all. We know everything they'll let us know.

Steve: Yeah.

Leo: Then get back to us. Steve does this show every Tuesday, 1:30 Pacific, 4:30 Eastern, 20:30 UTC. If you want to stop by and watch, please do. We'd love you to do that. Just go to TWiT.tv/live to watch the live stream. You can see us making all the shows we do roughly at those times. If you do that, do go into the chatroom because that's a big part of the behind-the-scenes crowd. Irc.twit.tv is the place to go to talk to the persons involved.

You can also get the show on-demand. Steve's got copies and transcripts. It's the only place you can get the transcripts of the show at his website, GRC.com. While you're there, pick up a copy of SpinRite, the world's best hard drive recovery and maintenance utility, and all the other freebies. SpinRite's his bread and butter, but everything else is free at GRC.com. We have audio and video at TWiT.tv/sn. And of course, as with all our shows, you can subscribe with your favorite podcast program or application. Just look for Security Now!, and that way you'll get it automatically every week, the minute it's available.

Steve, have a great week. Enjoy your Fourth. You doing anything fun in Irvine?

Steve: Nope, just going to hang out with Lorrie and enjoy the day.

Leo: You should. It'll be beautiful, I think.

Steve: Yup.

Leo: We'll see you next week on Security Now!.

Steve: Okay, buddy. Thanks.

Leo: Bye.