The Fuzzy Bench

Description: This week we consider the new time-limited offers being made for free telecommuting tools, the continuing success of the DOD's "please come hack us" program, another take on the dilemma and reality of Android device security, some unwelcome news about AMD processor side-channel vulnerabilities, a new potentially serious and uncorrectable flaw in Intel processors, a 9.8-rated critical vulnerability in Linux system networking, a "stand back and watch the fireworks" forced termination of TLS v1.0 and v1.1, and the evolution of the SETI@home project after 19 years of distributed radio signal number crunching. We then touch on a bit of miscellany, and finish by looking at a new and open initiative launched by Google to uniformly benchmark the performance of security fuzzers.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-757.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-757-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. He makes a pretty strong case for never using an Android phone. That's coming up. We'll also talk about AMD. Yes, parity with Intel. Now there's a speculative execution exploit for AMD processors. Oh, goodie. And we'll talk about the interesting technique called "fuzzing" used to find exploits. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 757, recorded Tuesday, March 10th, 2020: The Fuzzy Bench.

It's time for Security Now!, the show you've been waiting for all week. I know that because I meet people all the time who say, "I wait for it all week." That's the guy you're waiting for, Steve Gibson. He's always here on time. I'm often late. But thank you, Steve, for being patient.

Steve Gibson: It's always my pleasure, and thank you for always bringing your Leo personality to the fore. Whenever I'm going back, and I'm reducing the bandwidth of the video, I hear you just all perky and charged up, and I just think...

Leo: Bubbly. I'm bubbly.

Steve: ...that's what keeps us coming back for more is Leo is just, you know, he is a trained professional. You can fake it like the best of them.

Leo: Actually, I think what they come back for, Steve, is the information you impart. So I will just - I'll thank you, but I will get out of the way and let you go.

Steve: Well, I had a little fun with this week's title, which is "The Fuzzy Bench," because Google is doing another thing to benefit the community. Because, you know, they can afford to, and so why not? We're going to talk about a new public and open initiative to solve the problem of fuzzy security testing benchmarking, after we get to a whole bunch of other interesting stuff. We're going to look at the four time-limited offers now being made for free telecommuting tools to help corporations send their employees home and deal with remote work. We're going to look at the continuing success, sort of surprising, of the DOD's "come hack us please" program.

We're going to look at another take on the dilemma and the reality of Android device security, and look at some unwelcome news - well, unwelcome to AMD - about a new side-channel vulnerability which has been found that affects their chips. But Intel's not getting away from this week easily, either. There is a new, potentially serious, and uncorrectable flaw in Intel processors, unlike what we've been talking about before. There's also a 9.8, right, out of 10, so it's like, I'm not sure why it lost 0.2 - well, actually I am - a 9.8-rated critical vulnerability affecting Linux systems networking.

Leo: Uh-oh.

Steve: Yeah. But there's a spin to it, which is why most of our listeners don't need to worry, but some might. There's also a "stand back and watch the fireworks" forced termination of TLS v1.0 and 1.1 on the horizon. The evolution of the SETI@home project after 19 years of service. And we'll talk about that a little bit. Of course we are familiar with its distributed radio signal number crunching. We touch a bit on miscellany, and finish by looking at a new and, as I said before, open initiative launched by Google to bring some uniformity to benchmarking the performance of security fuzzing. Because what you really want is some good fuzz, Leo. That's what I always say. And we have an apropos Picture of the Week that just, I mean, I got maybe tweeted 50 different variations of this, some text, some images. And our listeners will understand. So, yeah.

Leo: It's a very geeky joke, but I like it.

Steve: It's way out there, yeah. The only thing spoiling this Picture of the Week is that they misspelled the key word of the entire thing.

Leo: Oh, they did, yeah.

Steve: But still the concept is conveyed. The picture says: "Due to the corono" - and that's where the misspelling was, it's corona - "virus COVID-19, all TCP applications are being converted to UDP to avoid handshakes."

Leo: Very geeky. Nice geeky joke.

Steve: Thank you for the gratuitous laughter, Leo.

Leo: No, it's good.

Steve: Appreciated for the soundtrack of the podcast.

Leo: It's like "There's no place like 127.0.0.1." It's one of those geek jokes.

Steve: And I'm in danger of forgetting that this is the 42nd anniversary of also one of my favorite books, which you guys were talking about over on MacBreak Weekly.

Leo: Yes.

Steve: And a listener who likes the book will realize that the 42nd anniversary...

Leo: Is the most important.

Steve: Forty-two, yeah.

Leo: I did not pick that up right away, by the way. I was embarrassed.

Steve: Whenever I see the number, well, for example, our favorite calculator on iOS platform, PCalc, its icon is a 42.

Leo: Mm-hmm.

Steve: For exactly that reason. Anyway, for those who haven't figured out already, this is Douglas Adams's "Hitchhiker's Guide to the Galaxy," which was just, I mean, I agree with all the opinions from MacBreak Weekly. It was so fun. Everybody has their favorite bits. And probably my very favorite one I can't do justice to because it's where he - and he does this several times in the book, where he's describing what's happening, and he says, "And then, after waiting, nothing happened. And then, nothing continued to happen." And he does all of this sort of like who's on first, who's on second.

Leo: It's lovely British wordplay that is unique. So funny.

Steve: Oh, it's just delicious.

Leo: Yeah. Love it, yup.

Steve: But the other thing, the thing I can do justice to, was when the whole group of them, Ford and Zaphod and there were two others, the gal Trillion and somebody else, they land their spaceship right in the middle of a soccer field, and they get out because they're - I mean, a soccer field on Earth. They get out, and they start walking away. And

Arthur, who's not like up to speed on all this yet, says, "We just landed our spaceship in the middle of a soccer field." And I think Ford says, "Yeah, just turn around and look at it." And so Arthur Dent, he turns around, and the way Douglas describes it, he can't quite focus on it. His vision just sort of slides off to the side. And so he's not really sure what's going on. And he turns back and says, "What the hell?" And Ford says, "Yeah, we just turned on the 'somebody else's problem' field." And Arthur says, "What?" And Ford says, "Yeah, that's a field which, when you turn it on, and it just sort of, when you see it, it tells your brain that's somebody else's problem."

Leo: And of course the Infinite Improbability Drive. And my favorite image of the whale conjured up out of nothing, who falls screaming to the planet Earth and only has time to think, "What's going on?" And on and on and on. I mean, what a brilliant, brilliant writer he was.

Steve: There was a lot of amazing stuff in there.

Leo: So good.

Steve: Okay. So apropos of the coronavirus, Microsoft, Google, LogMeIn, and Cisco are all offering limited-time free use of their telecommuting tools. Now, the cynic in me thinks that perhaps this is marketing opportunism, using a global pandemic to get people hooked on the benefits of stay-at-home work through a time-limited offer. But on the other hand, we all know how much resistance there is to change. And in a competitive market, if any of them made their services available for free, the others would be missing out on a true opportunity to expand their user base, hopefully permanently. So, yeah. With employees either being quarantined at home after international travel, or being encouraged to work from home, I think Microsoft has shut down their whole campus; right? They've just said everybody stay home.

So while this COVID-19 coronavirus situation stabilizes, Microsoft, Google, LogMeIn, and Cisco are all offering free use of their meeting collaboration and remote work tools. And, you know, since avoiding a daily commute can be a very positive thing for one's self, for the environment and everyone else, I wouldn't be surprised if a percentage of those who first make the jump because they really have no choice at the moment might end up embracing the option for the longer term. My brother-in-law is on the Peninsula down in San Mateo. And I think he only goes into the city two times a week because both the 101 and the 280 now, which are the main north-south arteries, they're just impassable due to commute traffic. And so, yeah, if you can do your job from a little desk in an office at home, it's tax deductible for the space that you're using for that.

So anyway, so everybody has a different offer. Microsoft is making their Teams available at no charge for six months. Their EVP and President of Microsoft's Global Sales and Marketing Operations tweeted - this is JP Courtois, I guess how you pronounce his name. He said: "At Microsoft, the health and safety of employees, customers, partners, and communities is our top priority. By making Teams available to all for free for six months, we hope that we can support public health and safety by making remote work even easier."

Leo: Oh, that's clever.

Steve: It is. It's like, okay, yeah. Google is offering free access to Hangouts Meet for G Suite Users. Not to be left behind, they announced this week that they are offering G Suite and G Suite for Education customers free access to their Hangouts Meet video conferencing features. So this includes larger meetings, for up to 250 participants per call; live streaming for up to 100,000 viewers within a domain; and the ability to record meetings and save them to Google Drive. The announcement stated that: "These features are typically available in the Enterprise edition of G Suite and G Suite Enterprise for Education, and will be available at no additional charge to all customers until July 1, 2020."

LogMeIn offers free what they call their Emergency Remote Work Kits. Bill Wagner, who's LogMeIn's CEO, posted a blog titled "Coronavirus Disruptions: An Offer for Support to Our Communities and Customers." He stated: "Starting immediately, we will be offering our critical frontline service providers with free, organization-wide use of many LogMeIn products for three months through the availability of Emergency Remote Work Kits." He says: "These kits will include solutions for meetings and video conferencing, webinars and virtual events, IT support, and management of remote employee devices and apps, as well as remote access to devices in multiple locations." He said: "For example, the 'Meet' Emergency Remote Work Kit will provide eligible organizations with a free site-wide license of GoToMeeting for three months." So again, three months.

Cisco is - well, okay. They've modified their free Webex license to now support meetings with no time limit, that is, the meeting itself has no duration limit, and up to 100 participants. And in addition, Cisco is now also offering a free 90-day license to businesses that are not currently Webex customers. Their announcement stated: "Additionally, through our partners and the Cisco sales team, we are providing free 90-day licenses to businesses who are not Webex customers in this time of need. We're also helping existing customers meet their rapidly changing needs as they enable a much larger number of remote workers by expanding their usage at no additional cost." And now that Cisco has finally patched the high-severity vulnerability in Webex, which was allowing strangers to barge into password-protected meetings without any authentication, that offer may be worth considering, as well.

So Cisco is 90 days. LogMeIn is three months. Google is until July 1st. So if we assume a start of April 1st for Google, that's also three months. Only Microsoft is offering double those free-use durations. It's offering six months' free use. And it would be annoying to get all set up on Google, LogMeIn, or Cisco, only to find that corona is taking longer than expected to calm down. Our illustrious President Trump has famously explained that this whole thing will be disappearing like magic once the weather warms up, which appears to be how Google, LogMeIn, and Cisco set their free-use schedules.

Leo: Oh, that's interesting. Until the summer.

Steve: Until the summer.

Leo: Yeah, well, good luck.

Steve: But if it should turn out, yeah, if it should turn out that the virus missed Trump's announcement, it might make more sense to run with Microsoft's six-month plan, if their terms offering meets your company's needs. Or Teams, I meant to say, if their Teams offering. I don't know about Teams, but are you guys using it? I heard you mention something.

Leo: We use Slack.

Steve: Oh, okay, Slack.

Leo: It's interesting. So like the Webex offer, I think it was, with 100 people for unlimited amount of time, you could see the use would be an office that would normally all be working together, everybody goes home, they all log into that conference, and they just run it all day. And so...

Steve: Just have it in the background.

Leo: Yeah. And I do know teams that do this. And I think it's a - it's not as good as working together, but because it's always on, you can say, "Hey, Joe." You can kind of participate as if you're all sitting in the same room.

Steve: It does bind a community together.

Leo: Yeah. It's not as good, but it's a lot safer in this day and age, anyway. I wonder how much of this...

Steve: Especially if Joe is sneezing.

Leo: Yeah. I wonder how much of this will be kind of incorporated in the future into just our general work life. I have a feeling this could be a watershed moment.

Steve: Yes. And that was the point I was making was that I'll bet, I mean, it is nice that the companies are doing this. But eventually those offers are going to expire. We also know how much inertia there is to change. So, you know, companies are sending their employees home for the first time ever, and scrambling around, needing to figure out how to manage that. So we know that these systems exist. As I said, my brother-in-law's been doing this for years in high-tech San Francisco-based companies on the Peninsula. So for them it's no biggie. You know, maybe he's reduced his two days a week to zero days a week because why not?

But it takes something to overcome inertia. This, as you say, Leo, could be the thing that gets companies to consider this, to bite the bullet. And once upon a time not everybody had broadband. Well, now everybody has broadband. I mean, you know, everybody's got a connection to the Internet now. So, yeah.

Leo: Yeah. Interesting.

Steve: Okay. So this report, I loved that it said "Unclassified" at the top. The DOD Cyber Crime Center (because these people love their acronyms, that's the DC3, the DOD Cyber Crime Center, DCCC, so DC3) Vulnerability Disclosure Program (and yes, that's the VDP) Annual Report for 2019, Volume 1. So this is looking back at how 2019 went. I've got a link for anyone who's interested in the full report. Our listeners may recall that four years

ago the U.S. Department of Defense first invited white hat hackers to hack into its systems in what was at the time dubbed "Hack the Pentagon." We had fun with it four years ago.

Today, after four years of success with this initially controversial program, the Pentagon continues to ask hackers to hit them with everything they've got. And really, what could possibly be more fun than attacking the online presence of our own U.S. Defense Department with their official permission, so that we won't be convicted for doing so. This just-released report states that after four years these industrious white hats are submitting more vulnerability reports than ever.

The DOD's Department of Defense Cyber Crime Center, that's that DC3, handles the cybersecurity needs for the DOD. And they're responsible for tasks including cyber technical training and vulnerability sharing. It also runs this Vulnerability Disclosure Program, the VDP. The VDP is what emerged from the initial Hack the Pentagon bug bounty program that the military ran back in 2016. And as I said, that initiative was so successful that it has evolved, and it continues to invite hackers to play with its systems, "play" as in see what mischief you can get up to.

Last year, as we reported - you'll remember this, Leo - the Air Force even brought an F-15 to DEFCON for hackers to tinker with. The F-15 Eagle, for those not up on the U.S. military's air combat inventory, is described as an "all-weather, extremely maneuverable, tactical fighter, designed to gain and maintain air superiority in aerial combat." So yes. They must have brought it on a - I don't think they probably landed it at Logan. They must have wheeled it over. But it was there, and it was powered up, and hackers were seeing what they could do with its avionics wireless systems.

That worked so well that next year the DOD plans to bring a satellite, and hackers will see what they can do with a powered-up satellite.

Leo: That's scary to think that satellites might be hackable. Geez.

Steve: Yeah, well, I mean, and what's interesting is this is working so well, Leo. Okay. We have some numbers about exactly how well. The report reveals that it processed 4,013 vulnerability reports coming in from a total of 1,460 white hat hackers last year. So 4,013 reports from 1,460 white hat hackers. It validated 2,836, so that's just shy of three quarters of all reports, for mitigation. And this past year was the busiest year for bug reports, representing a 21.7% increase over 2017, the previous busiest year, bringing the total number of bug reports for the program to 12,489.

The report explained: "These vulnerabilities were previously unknown to the DOD and not found by automated network scanning software, red teams, manual configuration checks, or cyber inspections. Without DOD VDP (Vulnerability Disclosure Program) there is a good chance," they're writing, "these vulnerabilities would persist to this day or, worse, be active conduits for exploitation by our adversaries." I mean, this is in the official DOD report. So, you know, what more does anyone need to know? This is a crazy good idea. Information exposure bugs were the most common type reported during this previous year, during 2019, followed by violation of secure design principles, cross-site scripting flaws, business logic errors, and open redirects which can be used to launch phishing attacks.

So moving forward, this Cyber Crime Center wants to expand the scope of the program beyond DOD websites to cover any DOD information system. It also wants to partner with the Defense Counterintelligence and Security Agency (DCSA) to create what it calls

a Defense Industrial Base (of course DIB) VDP program to secure the DOD's supply chain.

You know, all of this is such rare and right thinking on the part of our government. And this experience should go a long way toward further spreading the bug bounty concept throughout other areas of the U.S.'s massive government bureaucracy. I mean, here's the DOD saying, come on, bring it on. We'll pay you some money if you find some problems that we confirm. And the rest of the government would be nuts not to. And a report like this that says, you know, we didn't know about these things. We were looking. We didn't find them. But we put up a bounty, and it happened.

Oh, and by the way, the bug bounty program is not being privately managed. It's being managed by our favorite bug bounty management group, Hacker One. Which is, I still think, the best place to do these sorts of things. During the program's first pilot test, which ran only four weeks, from April 18th to May 12th back in 2016, the results were said to exceed all of their wildest expectations. The first report arrived 13 minutes after the program was initiated. They had 200 reports within the first six hours, and ended up with a total of 1,410 hackers participating, during which time, during this four weeks, they paid out $75,000 in verified bug bounties.

So at this point I think there's ample proof to suggest that any sizable company ought to be participating in the encouragement of white hat hackers to discover and report important vulnerabilities in their own systems, things that their people just cannot, just will not find. Anyway, a few thousand dollars of bounty is a small price to pay for the discovery and resolution of remotely targetable vulnerabilities. So this is a rare instance of the U.S. government planting the flag and leading the way in something that I think is just so useful.

Naked Security site, Sophos Security's Naked Security site, posted some useful detail and insight into one of the worries that I've been recently voicing here a lot. So I wanted to share what they had to say. Their piece, published yesterday, was titled "One billion" - with a "b" - "Android smartphones racking up security flaws." They said: "How long do Android smartphones and tablets continue to receive security updates after they're purchased?" They said: "The slightly shocking answer is barely two years, and that's assuming you bought the handset when it was first released. Even Google's own Pixel devices max out at three years. Many millions of users," they write, "hang onto their Android devices for much longer, which raises questions about their ongoing security as the number of serious vulnerabilities continues to grow.

"Add up all the Android handsets no longer being updated, and you get big numbers. According to Google's developer dashboard last May, almost 40% of Android users still use handsets running versions 5 to 7, which have not been updated for between one and four years. One in 10 run something even older than that, equivalent to one billion devices."

They said: "The point is brought home by new testing from consumer group Which? [W-H-I-C-H, that's a site, dot co dot uk] discovering that it was possible to infect popular older handsets mainly running Android 7 - the Motorola X, Samsung Galaxy A5, Sony Xperia Z2, the Google Nexus 5 [and they said] (LG), and the Samsung Galaxy S6 - with mobile malware. All the above," they said, "were vulnerable to a recently discovered Bluetooth flaw known as BlueFrag, and to the Joker strain of malware from 2017. The older the device, the more easily it could be infected."

They said Sony's Xperia Z2, running Android 4.4.2, was vulnerable to the Stagefright flaw from 2015. Of course, Leo, we spent a lot of time back then talking about Stagefright because it was so bad. It was a flaw in the library that rendered images that

runs with kernel privileges. And because MMS messages were able to send images, just the receipt of a specially crafted MMS message could take over the phone.

And I was curious, so I went to Amazon. The Sony Xperia Z2 is for sale today on Amazon new. It is and will always be vulnerable to the very worrisome Stagefright vulnerability where, as I said, just the receipt of a specially crafted MMS message is sufficient to remotely take over the phone. So here we're selling new Android-based devices which are already out of their update period, will never be updated, and they've got some of the worst, most widespread, well-known vulnerabilities. So they're vulnerable from the start.

Google recently had to remove 1,700 apps containing Joker, that's the Joker strain of malware from 2017, also known as Bread from its Play Store. And that's only the latest in its increasingly desperate rearguard action against malware being hosted under its nose on the Google Play Store. And it's not simply that these devices aren't getting security fixes any longer, but the older models also miss out on a bundle of security and privacy enhancements that Google has added later to versions 9 and 10.

So Kate Bevan, who is a computing editor for Which.co.uk, she was formerly with Naked Security. She wrote: "It's very concerning that expensive Android devices have such a short shelf life before they lose security support, leaving millions of users at risk of serious consequences if they fall victim to hackers." She also raised the interesting point that the idea that a device might only get updates for two years probably comes as a surprise to most Android users. She said: "Google and phone manufacturers need to be upfront about security updates, with clear information about how long they will last and what customers should do when they run out."

And of course Google, you know, they're doing the best they can. They said, in response to this, we're dedicated to improving security for Android devices every day. We provide security updates with bug fixes and other protections every month and continually work with hardware and carrier partners to ensure that Android users have a fast, safe experience with their devices. In other words, nothing. But what can they do? You know, they are offering Android to third parties to include in smartphones, and they don't have control over what those third parties do in detail.

And, you know, when I was thinking about this issue with awareness, it's true about the difference in smartphone handling versus, for example, PC handling. Anyone still using Windows 7 will have received, I did, a big full screen warning that it is no longer safe to use this operating system. You know, you can't not be aware of that fact as a desktop OS user. But no smartphone suddenly warns its user that it's no longer safe to use it. They just stop receiving security updates, if they ever get them in the first place after the original sale; and, you know, that's it. Users just sort of go on not knowing any better.

And Sophos continued, or ended their coverage about this, saying: "In truth, users are being squeezed between two forces. On the one hand, Google is determined to drive the evolution of Android for competitive reasons, releasing a new version every year. On the other side are manufacturers, eager to keep people upgrading to new models on the pretext that the older ones won't run these updated versions of Android," which turns out not to always be true.

They said: "Security sits somewhere between the two; and despite attempted reforms by Google in recent years to make security fixes happen on a monthly cycle, the reality is some way from that ideal." And of course we talk about this all the time. I don't know what we can do about this.

Leo: Should I stop recommending Android phones and just say you just need to buy an iPhone? I hate to see a monoculture, but I think it might be prudent.

Steve: I know. I mean, this really is a problem.

Leo: I mean, I guess it's safe to only buy a Google Android device. But then even then after three years they stop updating it.

Steve: Yeah. And so, I mean, you could argue we know how important security updates are. People are being hacked, and this is being leveraged against them. In the last few months we've talked about specific Android issues which would affect somebody crossing the border, where it's possible for their Bluetooth to be leveraged against them and to put malware on their phone unless they have this most recent, I think it was month's update. Otherwise, that phone is and will always be vulnerable.

So, I mean, to be responsible, the phone should bring up a warning, make it very clear that this device is no longer receiving security updates. I mean, it almost ought to be something dismissible, but nagging, you know, on an edge of the screen, where you have to push it, very much like, say, oh, look, we're using cookies on your browser. It's like yes, and you're forced to say, "Yes, I know." I mean, we've talked about how annoying that is. But, I mean, the user should be informed at this point. And then they could decide whether that matters to them or not. And maybe you could press a button to get a detail of all the things that have been fixed.

I don't know if your typical Android user cares. But still, I would say at a minimum, if the phone, if the Android OS detects that it is no longer receiving updates, it should proactively begin notifying its user. And they can dismiss the notice, but it should come back in a week, and they dismiss it again, and it comes back a week later, just to say, just so you know, you're not getting - this phone is no longer being updated. And so that would have two effects. It would inform users. It would allow them to decide how much they care about this. It might drive sales of a new phone, or it might drive the phone manufacturers to continue offering updates longer.

I mean, they're choosing not to. It's not that they couldn't keep those older phones updated. No one's making them. And so if the Android OS itself performed that notification task, users would be annoyed if a certain brand of phone started doing that after only two years. The point is the service life of the phone is much longer than that. It ought to be kept secure throughout its service life. And at that point it ought to just shut down. If it's not going to be kept updated, sorry.

Leo: It's a good point, yeah.

Steve: I mean, you know, the security really is part of what's being delivered.

Leo: Yeah. This is a tough one. I think I'm going to have to just say "Friends don't let friends use Android."

Steve: The only way to do it is to be on the Google or maybe on the Samsung train, and be willing to keep yourself current with a new phone. I mean, many - I know that, you know...

Leo: It's expensive. That's the real problem.

Steve: It is. That's exactly right.

Leo: We're talking a lot of money, yeah.

Steve: Yup. Yup. Otherwise I think I agree with you, Leo. You know, Apple is doing a great job of keeping phones that, you know, don't even run on battery any longer, still updated with security. And, you know, hats off to them. And I agree with you, no one wants a monoculture. But everybody I know, with a few exceptions, is the blue bubble in iMessage, and not the green bubble.

Leo: Now, AMD time.

Steve: So our prolific friends from the Graz University of Technology and Research Institute of Computer Science...

Leo: Kings of fuzzing.

Steve: Boy, they are. They really - they just are on this stuff. They've got that speculative execution stuff nailed, and working with Random Systems they responsibly disclosed the vulnerabilities to AMD back in August of 2019, so some time ago. This past Saturday, March 7th, AMD updated their AMD Product Security page with a response to the unwelcome news. So these guys responsibly disclosed. Then they recently published, they call it "Take A Way."

What AMD said was: "We are aware of a new whitepaper that claims potential security exploits in AMD CPUs, whereby a malicious actor could manipulate a cache-related feature to potentially transmit user data in an unintended way. The researchers then pair this data path with known and mitigated software or speculative execution side-channel vulnerabilities. AMD believes these are not new speculation-based attacks." Uh-huh. So of course AMD has a vested interest in that being true.

These researchers were some of the original guys behind the discovery of Spectre, Meltdown, and ZombieLoad vulnerabilities. They disagree with AMD's assessment. Their research paper is titled "Take A Way: Exploring the Security Implications of AMD's Cache Way Predictors." So, you know, Take a Way. Cache Way Predictors is the mechanism that AMD designed for enhancing the performance of their cache.

In the abstract of their full technical disclosure they said: "To optimize the energy consumption and performance of their CPUs, AMD introduced a way predictor" - that's "way" as in a noun - "a way predictor for the L1-data" - that's L1D cache - "to predict in which way" - this is hard to read - "cache to predict in which cache way a certain address is located." You know how caches are like n-way caches, like eight-way, four-way. The point being that normally a cache is much smaller than main memory.

So obviously you can't cache all of main memory. Well, that means you're caching a small percentage of the total main memory. That means you're having to map all of memory down into a much smaller cache. A one-way cache would mean that various

locations in main memory all mapped to the same cache slot. So if you happened to access another of those areas of main memory that collided with an existing one, you would lose the existing values cache. Because the designers realized that's not a good idea, they then introduced two-way and four-way and eight-way, where essentially you're able to cache up to that many ways of collision in the collapse of all of main memory down into the very much smaller size of the cache.

So it turns out it's better to do an n-way cache where there's depth to the cache than have the cache be one way and wider. Statistically, you end up with a bigger win. But which of the ways, for lack of a better term, is going to be used turns out to require some prediction. You can make the whole n-way cache, for example, an eight-way cache perform better by selecting which one of the ways in the cache slot you use. So now what they're saying makes sense, right, to predict in which cache way a certain address is located. Consequently, only this way is accessed, significantly reducing the power consumption of the processor.

So they said: "In this paper, we are the first to exploit the cache way predictor." I mean, this stuff is - you can imagine, you can sort of see why it's, like, lain undiscovered for so long. We were all just happy that it worked, and it was fast. And then the academics come along, and they go, uh, not so fast. We're the first, they say, "to exploit the cache way predictor. We reverse engineered AMD's L1D cache way predictor in microarchitectures from 2011 through 2019, resulting in two new attack techniques." They have the first one they call Collide+Probe.

They said: "With Collide+Probe, an attacker can monitor a victim's memory accesses without knowledge of physical addresses or shared memory when timesharing a logical core." The second one is called Load+Reload. They said: "With Load+Reload, we exploit the 'way predictor' to obtain highly accurate memory-access traces of victims on the same physical core. While Load+Reload relies on shared memory, it does not invalidate the cache line, allowing stealthier attacks that do not induce any last-level cache evictions."

They said: "We evaluate our new side channel in different attack scenarios. We demonstrate a covert channel" - and this was surprising - "with up to 588 kbps." Which is a huge amount of data. They're able to extract nearly, actually more than half a mbps through this attack. They said: "Which we also use then in a Spectre attack to exfiltrate secret data from the kernel. Furthermore, we present a key-recovery attack from a vulnerable cryptographic implementation. We also show an entropy-reducing attack on ASLR of the kernel of a fully patched Linux system, the hypervisor, and our own address space from JavaScript. Finally, we propose countermeasures in software and hardware mitigating the presented attacks."

So it's often been the case that AMD was skirting, over the last two years, some of these attacks that seemed targeted at Intel. However, Intel does not use a way predictor. AMD does. And so this one is all AMD's. So in other words, they are by no means immune to attacks similar to those that were affecting Intel. It was just that the academics hadn't yet turned their attention on AMD. Whoa.

Leo: Amazing.

Steve: Yeah, I mean, this is all down in the weeds. I think, you know, that the one thing we keep seeing is that it's the core sharing. I think what we're going to end up having to do is to isolate the cores in order to get our performance back. Which is to say that we already know that turning off hyperthreading is the first thing you do for the Intel because hyperthreading explicitly allows threads to share a core. The problem is it does

become more expensive when you're not able to allow cores to share Level 1, Level 2, Level 3 caching. And so there is that other memory technology that's on the horizon. Remember, Leo, we talked about it? I think HP was claiming to have something soon. It was a XPoint memory. I've sort of kept my eye on it over time, and it has not gone away. It's being a little more challenging than people thought.

Leo: That's not Optane? That's something different.

Steve: Yes.

Leo: Oh, it is Optane. Okay.

Steve: That's the Optane memory.

Leo: Yeah. They renamed it. XPoint originally, and Optane was the brand. But they're selling Optane stuff like crazy.

Steve: Yeah. But they haven't yet moved that Optane technology all the way into the processor, or at least nothing [crosstalk].

Leo: Right. No, they're using it as smart memory or storage, yeah.

Steve: Right. But it is very, very fast. And the promise was that it could be way faster than DRAM, yet with a density of mass storage. So, you know, very compelling-looking. Or the density at least of - the density of DRAM, yet the speed of static RAM. That's, you know, we talked about this years ago. It's still sort of around there. But it always, you know, we've also talked about battery technology that was going to revolutionize everything years ago.

Leo: Oh, yeah.

Steve: Whatever happened to the supercapacitors? We're still waiting for those and other miracle next-generation battery stuff. It's one thing to make one in the lab. It's another thing to make it work in production and also not mess up the environment, which is something we would like to protect moving forward.

Okay. And Intel also has a different sort of serious new trouble on its hands. And, boy, this is going to result in some lawsuits. On March 5th, last week, that's Thursday, Positive Technologies posted a report titled "Intel x86 Root of Trust: Loss of Trust." Which is not to say that it doesn't affect 64-bit. It does. Bizarrely, this came as no surprise to Intel. They were apparently hoping that no one would notice, which is why I really do think they've opened themselves to some lawsuits.

Leo: That's a unique security stature.

Steve: Yeah.

Leo: Maybe nobody will notice.

Steve: Maybe we can - yeah. So what has effectively happened is that a flaw which is very similar to Apple's unpatchable CheckM8 boot ROM flaw has been found to be present in all Intel processors produced in the last five years. The exception is the very latest 10th-generation processor which doesn't have the problem because they fixed it. But all the silicon that all of us are using from Intel in the last five years has a flaw in its ROM.

Okay. So here's what Positive Technologies had to say. I've edited it down a bit. But this is their disclosure because it does a perfect job of explaining it: "The scenario that Intel system architects, engineers, and security specialists perhaps feared most is now a reality. A vulnerability has been found in the ROM of the Intel Converged Security and Management Engine." That's an important acronym. You'll hear me using that a lot here in the next 10 minutes: CSME, Converged Security and Management Engine. In other words, the single point of failure. It's where everything converges.

They wrote: "This vulnerability jeopardizes everything Intel has done to build the root of trust and lay a solid security foundation on the company's platforms. The problem is not only that it is impossible to fix firmware errors that are hard-coded in the Mask ROM of microprocessors and chipsets. The larger worry is that, because this vulnerability allows a compromise at the hardware level, it destroys the chain of trust for the platform as a whole.

"Positive Technologies specialists," they wrote, "have discovered an error in Intel hardware, as well as an error in Intel CSME firmware, at the very early stages of the subsystem's operation, in its boot ROM. Intel CSME is responsible for initial authentication of Intel-based systems by loading and verifying all other firmware for modern platforms. For instance, Intel CSME interacts with CPU microcode to authenticate the UEFI BIOS firmware using BootGuard. Intel CSME also loads and verifies the firmware of the Power Management Controller responsible for supplying power to Intel's chipset components.

"Even more importantly, Intel CSME is the cryptographic basis for hardware security technologies developed by Intel and used everywhere, such as DRM, TPM, and Intel Identity Protection. In its firmware, Intel CSME implements EPID (that's Enhanced Privacy ID). EPID is a procedure for remote attestation of trusted systems that allows identifying individual computers unambiguously and anonymously, which has a number of uses. These include protecting digital content, securing financial transactions, and performing IoT attestation. Intel CSME firmware also implements the TPM software module which allows storing encryption keys without needing an additional TPM chip." And many computers no longer have such chips because Intel moved it into their own domain, and now it turns out that's not secure.

"Intel tried to make this root of trust as secure as possible. Intel security is designed so that even arbitrary code execution in any Intel CSME firmware module would not jeopardize the root cryptographic key, what they refer to as the 'chipset key,' but only the specific functions of that particular module.

"Unfortunately," they write, "no security system is perfect. Like all security architectures, Intel's had a weakness, the boot ROM in this case. An early stage vulnerability in ROM enables control over reading of the chipset key and the generation of all other encryption keys. One of these keys is for the Integrity Control Value Blob" - literally, Integrity Control Value Blob (ICVB). "With this key [or blob], attackers can forge the code of any

Intel CSME firmware module in any way that authenticity checks cannot detect. This is functionally equivalent to a breach of the private key for the Intel CSME firmware digital signature, but is limited to a specific platform."

They go on. There's one thing I wanted not to miss. Ah, yeah. "The vulnerability discovered" - oh, no, here. "And since the ROM vulnerability allows seizing control of code execution before the hardware key generation mechanism in the Secure Key Storage is locked, and the ROM vulnerability cannot be fixed, we believe that extracting this key is only a matter of time. When this happens," they wrote, "utter chaos will reign." Or maybe my word, "mayhem." "Hardware IDs will be forged, digital content will be extracted, and data from encrypted hard disks will be decrypted.

"The vulnerability discovered by Positive Technologies affects the Intel CSME boot ROM on all Intel chipsets and SoCs available today other than Ice Point." That's, as I said earlier, the 10th-generation build. "The vulnerability allows extracting the Chipset Key and manipulating part of the hardware key and the process of its generation. However, currently it is not possible to obtain that key's hardware component, which is hard-coded in the SKS directly. The vulnerability also sets the stage for arbitrary code execution with zero-level privileges in the Intel CSME."

They said: "We will provide more technical details in a full-length whitepaper to be published soon. We should point out that, when our specialists contacted Intel to report the vulnerability, Intel said the company was already aware of it." CVE-2019-0090, okay, that's a low number. That's a low CVE-2019 number, 90.

"Intel understands they cannot fix the vulnerability in the ROM of existing hardware. So they are trying to block all possible exploitation vectors. The patch for CVE-2019-0090 addresses," they write, "only one potential attack vector involving the Integrated Sensors Hub. We think there might be many ways to exploit this vulnerability in ROM. Some of them might require local access; others need physical access."

Okay. So what does this mean? It means that all Intel processors released in the past five years contain an unpatchable vulnerability that could allow hackers to compromise almost every hardware-enabled security technology that's designed to shield sensitive data of users, even when a system is compromised, that is, to keep it protected even under compromise. And as with Apple's CheckM8, because the vulnerability resides in ROM, it cannot be repaired without replacing the chip.

This Intel CSME is a separate security microcontroller incorporated into the processors that provides an isolated execution environment protected from the host operating system running on the main CPU. Maybe it would be possible to exchange with an identical processor that is, you know, same processor, but with the ROM, this CSME ROM fixed, and we'll see what happens. The CSME is responsible for the initial authentication of Intel-based systems by loading and verifying firmware components, the root of trust-based secure boot, and cryptographically authenticates the BIOS, Microsoft System Guard, BitLocker, and other security features. Intel obfuscated and dramatically downplayed this problem when it was previously patched last year. At that time, Intel described it as a "privilege escalation and arbitrary code execution" in Intel CSME firmware modules.

And remember the researchers wrote: "Since the ROM vulnerability allows seizing control of code execution before the hardware key generation mechanism is locked, and the ROM vulnerability cannot be fixed, we believe," they wrote, "that extracting this key is only a matter of time. When this happens, utter chaos will reign. Hardware IDs will be forged, digital content will be extracted, and data from encrypted hard disks will be decrypted." And if the guys from Graz University are listening, they're probably rubbing their hands

together right now. It's like, oh, goodie. Something new that we can go and demonstrate, turn into proof of concept.

There is no proof of concept code yet, but this is something Intel knew about. It seriously, I mean, if the guys that did this research at Positive Technologies are correct, then this is potentially not good. However, we haven't - there's been no discussion yet of what it takes to leverage this in fact. And this is not going to be remotely exploitable. It's possible to do very clever things during a reboot. So I don't know, if a bad guy got in and could reboot the system, if that would allow them to get control. It may need actual physical access.

On the other hand, there are many instances where we're presuming that our Intel-based systems are protected against physical access using TPM. The whole point of storing this stuff in storage that cannot be read is physical access doesn't let you read it. So, you know, most of the people who are listening to this podcast are never going to be affected by this. But this is a serious issue for Intel to deal with. And so far it looks like they have - you can imagine when this came to their attention somehow a year ago, after already a year of Spectre and Meltdown, they must have just been thinking, oh, please. We liked it when we were just printing money. Actually having to have secure processors, that's much harder. And Leo.

Leo: Yes?

Steve: SETI@home.

Leo: Oh, I know.

Steve: Is shutting down its distributed computing project after 21 years.

Leo: How many aliens did we find?

Steve: Yeah, that's a problem. But there's a theory about that.

Leo: Oh.

Steve: SETI, of course, is the Search for Extraterrestrial Intelligence. And Leo, in light of recent events, the team has decided to turn their search inward and start searching for any signs of terrestrial intelligence.

Leo: Good luck on that.

Steve: Yeah. I don't know.

Leo: You're going to need more than a telescope to find that.

Steve: Okay, well, no. But at the end of this month, exactly three weeks from today, on March 31st, they will be evolving their very cool 21-year-long project. They're not giving up. But it's time to figure out what, if anything, they have found during this period. So as many of us know, SETI@home is a long-lived, multidecade, 21-year-long distributed computing project where throughout these last 21 years, and thanks to the global Internet, volunteers spread across the world have been contributing their spare CPU cycles to analyze raw data received from the Arecibo Radio Telescope in Puerto Rico and West Virginia's Green Bank Telescope for signs of coherent and organized radio emanations, which we presume and hope would invariably arise from any non-Earthly intelligent beings.

So during this 21-year run, UC Berkeley's SETI Research Center, which went online on May 17 of 1999, has been sending small chunks of raw data out to volunteers. Each of these little chunks forms a job to volunteers. The raw data are being siphoned in a piggyback form passively, while those two telescopes are being used for other scientific programs. So they're not having to be dedicated to SETI. SETI just said, hey, would you mind if we tap in and process all the noise that you're listening to? And the scope said, "Yeah, cool, you know, because we like aliens." I mean, they're out there peering into the past.

So the data are digitized, stored, and then sent out through the SETI@home facility. They're divided up, both in frequency and in time, and then analyzed to search for signals, where signals are defined as variations that cannot be ascribed to noise and hence contain information. Using distributed computing, SETI@home sends millions of chunks of data to the home computers of users who are participating.

I was curious, so I did a little bit of digging. The software searches for five types of signals that distinguish them from noise, that is, that would distinguish the signals from noise. They look for spikes in power spectra; Gaussian rises and falls in transmission power, which would possibly represent the telescope's beam passing over a radio source. They look for triplets, which are three power spikes in a row; also pulsing signals that possibly represent a narrow-band digital-style transmission. And they perform autocorrelation to detect signal within the waveforms.

So last week what the SETI guys said was, they said: "On March 31st, the volunteer computing portion of SETI@home will stop distributing work and will go into hibernation." They said: "We're doing this for two reasons: First, scientifically, we're at the point of diminishing returns; basically, we've processed all the data we need for now. Second, it's a lot of work for us to manage the distributed processing of data. We need to focus on completing the back-end analysis of the results we already have, and writing this up in a scientific journal paper."

They said: "We're extremely grateful to all of our volunteers for supporting us in many ways during the past 20 years. Without you there would be no SETI@home." They finished: "We're excited to finish up our original science project, and we look forward to what comes next." So very, very cool. I remember running it for a while at some point, just to see what it looked like. And the concept of a big distributed number-crunching project for what I would regard as a very good cause, it always appealed to me, and I imagine to our listeners, as well.

Leo: So what's your theory, why they never found any aliens?

Steve: Well, the theory is that are some bad aliens.

Leo: Oh, no. They're hiding the good aliens?

Steve: Yeah. The good...

Leo: They're sneaking around?

Steve: Exactly. It turns out it's better to be dark and not let your emanations, you know, all of your episodes of "I Love Lucy" go streaming out into the void in perpetuity.

Leo: Well, that was the premise of the three-body problem, right, is don't contact them. That's just asking for trouble.

Steve: Yeah. It may not be a good idea. In fact, the book that I'm now in the second of the trilogy, John I'm sure has probably read it twice already. I reread the first. Actually, he did, too. This is Peter Hamilton's...

Leo: "Salvation."

Steve: "Salvation" trilogy. And it actually - it doesn't really give anything away because the way he's writing this, it annoys me. He's jumping back and forward in time.

Leo: Oh, I hate that.

Steve: I don't - it is so annoying.

Leo: Movies have been doing that a lot lately. It drives me nuts. I think, is this now? Where are we?

Steve: Yeah. And it's like, do you think it's not interesting enough just to give it to me linearly? Because that's the way I like to ingest stories.

Leo: Life happens.

Steve: Yeah, and I don't need to know the future in order to find the past because, I mean, he's a great writer. But anyway, so he's jumping us around. And so anyway, the point is there are bad aliens. And you really do want to hide your radio emanations from - anyway. So I don't know how the story's going to turn out, but it's another one of his fabulous pieces of work. I mean, it just - he just writes so well.

Okay. Of importance to, eh, maybe to some of our listeners, but big importance to the Linux distro guys, we have a critical flaw. And Leo, here's why it's not a big concern. It's in the Point-to-Point Protocol.

Leo: Oh, I don't use that, yeah.

Steve: Yeah. Nobody does. Well, that's not true. But anyway, so the PPP daemon, if it's there, your system is vulnerable. So last Thursday US-CERT issued an advisory warning users of a newly discovered and dangerous 17-year-old remote code execution vulnerability affecting the PPP daemon, that's pppd, which is always present, if not usually running, on most Linux-based OSes. And they note that PPP may be powering the communications of Linux-based embedded networking devices. There are a few of note at the end of this coverage.

So what is it, PPP? I would imagine certainly our more mature listeners have seen PPP around. It is seriously old school. It was originally used with dial-up modems - remember those? - and later with DSL and some VPN links. So while it's not in super active use today, it's old and, as with most things on the Internet, never really dies until it is deliberately removed. Consequently, this won't affect most Linux users. But it is yet another legacy protocol that has been found to be critically buggy and which, if it were to be in use and were not updated, could open any system at either the client or the server end of a PPP connection to hostile takeover.

Thus the reason it got the 9.8 rating. I mean, it is remotely exploitable and trivial. It's a critical stack buffer overflow which was discovered by the IOActive guys due to a logical error in an aspect of it, the Extensible Authentication Protocol (EAP) packet parser which runs on the pppd software. And as its name suggests, Extensible Authentication Protocol, it's an extension that provides support for additional authentication methods in PPP connections. The vulnerability is being tracked as CVE-2020-8597, and it carries a CVSS score of a whopping 9.8 because it can be exploited by unauthenticated attackers to remotely execute arbitrary code on any system that has a PPP server or client and take control of them.

However, as I noted, you first need to be actually using PPP for there to be any danger. So as long as PPP is never used, you're not in danger. But if it is in use, all an attacker needs to do is send an unsolicited malformed EAP packet to a vulnerable PPP client or server - and they're all vulnerable for the last 17 years until being patched - over a direct serial link, ISDN, Ethernet, SSH, Socket CAT, PPTP, GPRS, or ATM network, that is, you know, any of these transports that support PPP, and all of them do.

And there are many things that do maintain a persistent connection, like old-school ATM machines. An ATM might have been around for years, be running an older version of Linux to drive its hardware, and has and maintains a persistent connection out to home base over PPP. And since pppd often runs with high privileges and works in conjunction with kernel drivers, the flaw could allow attackers to potentially execute malicious code with the system or root-level privileges.

So anyway, the advisory says: "This vulnerability is due to an error in validating the size of the input before copying the supplied data into memory." In other words, eh. Classic buffer overflow, "an error in validating the size of the input before copying the supplied data into memory." Which means the attacker supplies more memory than they are saying or more data than they claim they're supplying. This thing doesn't catch it, and the attacker is able to write their own data into the system memory and then arrange to give it execution. The vulnerability is in the logic of the EAP parsing code, and you don't need to be using EAP. It's always there listening for an EAP request packet.

And they said: "It is incorrect to assume that pppd is not vulnerable if EAP is not enabled or EAP has not been negotiated by a remote peer using a secret or passphrase." In other words, it's there, it's listening, and if you've got a PPP client or server, and there's an open connection, you could be in trouble.

The researcher said the Point-to-Point Protocol Daemon versions 2.4.2 through 2.4.8. Okay. So 2.4.2, 2.4.8. In other words, there has not been much action, not surprisingly, over the last 17 years in the Point-to-Point Protocol support. That's all versions released in the last 17 years. They are all vulnerable to this longstanding remote code execution vulnerability. The Linux distros are updating their PPP daemons. So for those whose systems are being proactively maintained, this problem will disappear soon. All the Linuxes are going to get this fixed.

But again, we absolutely know that a great many machines which are serving turnkey embedded functions will not be updated, and will now become subject to targeted attacks. And if such a machine were on the network of any security-sensitive entity, for example, a major enterprise, maybe something controlling the power grid or a nuclear reactor, this could open any of those otherwise secure systems to attack. So I hope this comes to the attention of people who are responsible for anything that might be using this point-to-point protocol.

And as I mentioned above, the vulnerable protocol is also present in some turnkey product offerings that people may not be aware of. Cisco's CallManager is one of them. TP-LINK products use it. The OpenWRT Embedded OS uses it, and Synology's products use Point-to-Point Protocol. Again, it would have to be exposed. If you're behind a firewall, if you're behind a NAT router, you're probably okay. So in the unlikely event that something is actually using Point-to-Point Protocol, keep an eye out for an update. You'll want to apply that.

Leo: I just checked my default install of Pop!_OS, and PPP and pppd are installed. I'm sure the daemons aren't started. That would be unlikely. But I just uninstalled them, and that's fine. You know, I just removed them.

Steve: Good, yes. I was going to - that's very...

Leo: Sudo apt remove PPP, and everything's gone.

Steve: Nice.

Leo: Another way to do it.

Steve: Yes. And I'm sure nobody will miss it because nobody's using it.

Leo: Yeah.

Steve: Okay. So speaking of deliberately terminating legacy protocols, that's obviously a recurring theme of the podcast also. It's that this old stuff doesn't die unless you go out and kill it on purpose. Back in the spring of 2018, so coming up on two years ago, the developers behind all of our major web browsers - Safari, Chrome, Firefox, and Edge - gathered following the release of TLS v1.3 and jointly announced in October of that year their intention to remove support for TLS v1.0 and 1.1 early this year.

So the first stage of this deliberate protocol deprecation began last year, when browsers began labeling sites that were using TLS 1.0 and 1.1 with a "not secure" indicator in the

URL address bar and on the lock icon. This hinted to users that the HTTPS connection was not as secure as they might imagine or hope, since the maximum TLS version a site is offering was not something any end user has control over. The hope was that users would notice this and complain to websites so that those sites would proactively move to support newer TLS versions, namely 1.2 and 1.3.

So how'd that work out? Well, today, more than 850,000 websites are still offering the use of protocols no higher than TLS 1 or 1.1. This includes sites of major banks, governments, news organizations, telecoms, ecommerce stores, and Internet communities. This according to a recent report published by Netcraft, which is the well-known U.K. technology networking firm. All of the 850,000 websites do offer HTTPS, but only over versions of TLS that are now regarded to be weak and insecure. And of course we've been talking about this for the last 12.5 years. Time flies. TLS 1.0 and 1.1 have gotten old. 1.0 was released in 1996, and 1.1 was 14 years ago in 2006.

The protocols support the use of, as we know, many weak cryptographic algorithms and are vulnerable to a series of cryptographic attacks that we've carefully described during this podcast's 12-year history, including BEAST, LUCKY 13, SWEET 32, CRIME, and POODLE. To varying degrees, these attacks allow attackers to decrypt portions of HTTPS connections and access varying numbers of bits of a user's plaintext web traffic. None are critical. But given that we now have far superior alternatives, it really is time to move on.

So to that end, later this month, okay, later this March, I mean, this current month, our web browsers, all of the web browsers intend to switch from showing a hidden warning, which needs to be deliberately accessed to be seen, to showing full-page errors when users access sites over TLS 1.0 or 1.1. So that should be interesting. I have a feeling we will be talking about that a few weeks from now because, boy. That will finally get the attention of the website operators who are just not bothering to update their servers in order to support newer protocols. That's crazy. So anyway, stay tuned. Three weeks from today is March 31st, and this will have happened by then. I have a feeling it'll be back in the news.

Leo, a couple bits of miscellany. We talked last week, I just went bonkers raving about that cool Kickstarter project, SandSara. I got email.

Leo: Uh-oh.

Steve: "Hi, Steve. Huge fan of the podcast."

Leo: Oh, good. Oh, that's wild.

Steve: "Kind of surreal to hear you talk about my project after listening to you talk so much" - oh, and, yeah, I do talk a lot - "over the years." He says: "I really appreciate all the nice things you said. If you ever get to building your own big table and have any questions, don't hesitate to reach out." Then he says: "I just upgraded your order to include both the RGB lighting and tempered glass upgrades." He said: "I also wanted to do a special offer for your listeners. Any order that comes through this link will get the RGB upgrade for free." And he said: "You can redirect https://grc.sc/sand to that one if you want." So he already knew about my shortcut that I created for last week's podcast.

And so I immediately switched the link over so that any of our listeners using that would get the benefit of the fact that the guy behind the project is also a Security Now! listener. And the last time I checked, 17 of our listeners - 3,445 had clicked on the shortcut, 17 of

whom decided that, like you and me, Leo, they had to have one. When I looked this morning to put these notes together, there were still six remaining of the circular birch wood, six of the circular dark walnut, nine of the star in birch, and 22 of the star in dark walnut. So they are there. I don't expect they'll be there for a lot longer. And for anyone using grc.sc/sand, if you use that link, you also get the RGB lighting. It's a ring of LEDs around the inner perimeter that lights up the sand so you can see it at night.

Leo: Nice, nice.

Steve: So anyway, very big thanks to Ed, and I'm delighted that he's a listener to Security Now!. What a hoot.

And the last little bit of miscellany, before we talk about Fuzzy Benches, I got a Twitter from - his handle is @freddfarkl, who said, can you make mention of a specific Vitamin C liquid that seems to be so well liked. You and I were talking about it. Of course last week I talked about Vitamin D relative to the podcast I did a long time ago. And we talked about other things that one could do. Vitamin C is another underappreciated vitamin. You and Lorrie really like the liquid. Anyway, so I created a shortcut to take people who are interested to an Amazon page, grc.sc/liquid. That's the shortcut, grc.sc/liquid. And that will take you there. That's...

Leo: That one's really good. I didn't use it because it has sugar in it. So it's not ketogenic. I use one called, from Aurora, it's called Mega Liposomal Vitamin C. I get it at the Vitamin Shoppe. It's sugar-free. It's also three grams per serving instead of one. So it's a little more Vitamin C. And I know you...

Steve: Wait, no, three grams. Oh, three grams of C.

Leo: 3,000 milligrams, yeah.

Steve: And is that for a tablespoon?

Leo: Yeah, a good amount.

Steve: Or is that a teaspoon? Okay. So the one you've got onscreen, the one that my shortcut links to...

Leo: This tastes better because it's got sugar in it.

Steve: Yeah, Lorrie likes it a lot.

Leo: Yeah, I like it a lot.

Steve: And that's three bottles. That's a three-bottle case for that $30-some. So if you price it out, it ends up being a good deal. And it certainly is easy to take.

Leo: Yeah. I put it in my water, drink it all day.

Steve: That's exactly the way to do it because Vitamin C is water soluble. It doesn't stay in us, unlike Vitamin D that does and, in fact, builds up. Vitamin C is constantly being processed and eliminated.

Leo: It's a good idea to drink it, I think you told me this, with a straw because it is acidic, and you don't want to run over your teeth.

Steve: I may have mentioned that, or somebody else did. But yes, that is the case. There are a couple people whose lips have become a little - they described it as "chapped."

Leo: Well, I think that's from the Vitamin C. But I'm saying the ascorbic acid is also bad for your tooth enamel.

Steve: Oh, I see, the actual - the lemon flavor.

Leo: Yeah. If you're taking pills, it doesn't hit your teeth. But if you're drinking it, use a straw so it goes past your teeth, down your throat.

Steve: Right, right. Okay. The Fuzzy Bench. This was a post on the open source Google blog titled "FuzzBench: Fuzzer Benchmarking as a Service," which they posted early last week, actually last Monday. So, okay. Let's review fuzzing so that we're all on the same page. I pulled from Wikipedia since they have a nice description of it. And we've talked about it on this podcast before.

Wikipedia said: "Fuzzing or fuzz testing is an automated software testing technique" - automated being key - "that involves providing invalid, unexpected, or random data as inputs to a computer program. The program is then monitored for exceptions such as crashes, failing built-in code assertions, or potential memory leaks. Typically, fuzzers are used to test programs that take structured inputs. This structure is specified, for example, in a file format or protocol, and distinguishes valid from invalid input. An effective fuzzer generates semi-valid inputs that are 'valid enough' in that they are not directly rejected by the parser, but do create unexpected behavior deeper in the program and are 'invalid enough' to expose corner cases that have not yet been properly dealt with."

So we've talked about this. I love the idea. I remember, I was trying to remember, his name is Rick. He was in Southern California. There was a security firm whose name I've forgotten, and I haven't heard of them for a long time so maybe they were purchased by someone. But they had a huge room of PCs running fuzzers on various software. And as I described it at the time on the podcast, they're all running along, and someone kind of just visually scans the room and makes sure that everything's still running. Every so often, one of them crashes.

And the fuzzer has been keeping a log of what it's doing and making sure that that gets written to nonvolatile storage before any crash that that data that it is about to send into the app might cause. So then, when a machine crashes, you look and see what it was

that the app under fuzz testing had just been given prior to its crash. Sometimes you may have to back up a little bit. If an earlier bit of data destabilized the app, then the subsequent data caused the crash.

But the point is, apps should not crash. And how many times have you and I noted, Leo, that exploitation begins with something that crashes, and then a really talented hacker rolls their sleeves up, figures out exactly why the crash occurred, and then goes, ah, I know how to execute my own code, rather than have it just go off into the wild and crash.

And so the point is that this isn't a genius coder causing something to crash. It is an interesting way to uncover bugs in programs. Rather than having a highly skilled hacker with domain-specific knowledge like of all past vulnerabilities, carefully and methodically poking at something, trying to find things that worked before, fuzzing is sort of like - I guess I would use the analogy of the thousand monkeys, all pounding on typewriters, to see whether any of them might by pure chance hit upon something novel and useful. Maybe.

So this is another of Google's "working to make the world a better place because they have plenty of money, so why not?" Last week they posted the explanation, this explanation of their latest initiative. They said: "We are excited to launch FuzzBench, a fully automated, open source, free service for evaluating fuzzers. The goal of FuzzBench is to make it painless to rigorously evaluate fuzzing research and make fuzzing research easier for the community to adopt."

And I don't remember now when it was or what it was, but I do remember that we discussed some research coming from academia where we spent some time talking about the fact that fuzzing is not as easy as just throwing noise at something. In order to get the largest scope of testing to increase the probability of actually finding something, there were strategies involved. And so there's actually much more to this than you might just immediately think.

They said: "Fuzzing is an important bug-finding technique. At Google, we've found tens of thousands of bugs with fuzzers like libFuzzer and AFL. There are numerous research papers that either improve upon these tools" - and they cite a bunch of them - "or introduce new techniques" - and they cite those - "for bug finding.

"However," they say, "it's hard to know how well these new tools and techniques generalize on a large set of real-world programs. Though research normally includes evaluations, these often have shortcomings. They don't use a large and diverse set of real world benchmarks, or they use few trials, or they use short trials, or they lack statistical tests to illustrate whether findings are in fact significant. This is understandable since full-scale experiments can be prohibitively expensive for researchers. For example, a 24-hour, 10-trial, 10-fuzzer, 20-benchmark experiment would require 2,000 CPUs to complete in one day.

"To help solve these issues, the OSS-Fuzz team is launching FuzzBench, a fully automated, open source, free service. FuzzBench" - I just love saying it. "FuzzBench provides a framework for painlessly evaluating fuzzers in a reproducible way. To use FuzzBench, researchers can simply integrate a new fuzzer, and FuzzBench will run an experiment for 24 hours with many trials and real-world benchmarks. Based on data from this experiment, FuzzBench will produce a report comparing the performance of the fuzzer to others and give insights into the strengths and weaknesses of each fuzzer. This should allow researchers to focus more of their time on perfecting techniques and less time setting up evaluations and dealing with existing fuzzers."

Leo, I have - which one is it? Oh, the sample report, two links down. It's on the middle of page 15 of the show notes. Very, very interesting sample report to show what this thing produces.

They said: "Integrating a fuzzer with FuzzBench is simple, as most integrations are less than 50 lines of code." It's Python based. "Once a fuzzer is integrated, it can fuzz almost all 250-plus OSS-Fuzz projects out of the box." And you're scrolling through the projects right now, Leo. They said: "We've already integrated 10 fuzzers, including AFL, libFuzzer, Honggfuzz, and several academic projects such as QSYM and Eclipser.

"Reports include statistical tests to give an idea how likely it is that performance differences between fuzzers are simply due to chance, as well as the raw data so researchers can do their own analysis. Performance is determined by the amount of covered program edges, though we plan on adding crashes as a performance metric. You can view a sample report here." And that's that page that's onscreen.

And so, for example, fuzzers, they said, were run against all sorts of well-known code bases. And I was curious, and I wanted to share them with our listeners. So cURL, FreeType, JsonCpp, libjpeg, libpcap, libpng, libxml2, OpenSSL, OpenThread, PHP, SQLite3, Vorbis, and the WOFF2 font interpreter. So real-world libraries which are there. And so basically somebody with a new fuzzer just sort of interfaces it. They're estimating about 50 lines of code. They give you a sample integration so you can see. And this allows your fuzzer to be driven by their FuzzBench and have it added to a report that already includes all of those existing fuzzers so you can see how you're doing with the design of your own fuzzer.

Under "How to Participate" they said: "Our goal is to develop FuzzBench with community contributions and input so that it becomes the gold standard for fuzzer evaluation. We invite members of the fuzzing research community to contribute their fuzzers and techniques, even while they're in development. Better evaluations will lead to more adoption and greater impact for fuzzing research. We also encourage contributions of better ideas and techniques for evaluating fuzzers. Though we have made some progress on this problem, we have not solved it" - meaning generically the fuzzer problem - "and we need the community's help in developing these best practices."

So, yeah, another big tip of the hat to Google for so willingly and usefully giving back to the community, and really to the world. Although everyone knows that fuzzing is not the end-all solution for hardening our software and making it bulletproof, it inarguably provides another avenue into providing real-world code security. And as we know, efficient fuzzing is not easy. It's the province of university academics, and there's still a lot of progress to be made. So having an open platform for testing and comparing fuzzers side by side only helps academics to test and hone their new ideas. So bravo to Google.

Leo: It's cool.

Steve: Yeah.

Leo: It's really cool. There are, in the test-driven design world, where you write tests for your code, there are techniques you can use, I know the language I like, Racket, and I know LISP has this - where you can generate sample test results. And it's sort of like reverse fuzzing. As you're writing your code, you bang on it with tests that have generated a huge number of different samples and see how it comes out. And as long as every test passes, you're not proving that it will never fail, but you're expanding the universe of possible bugs.

And in fact there's even, as you know, an academic discipline of proving your code. And there are tools for doing that, to actually write tests that say, well, this is provably accurate code. It only works in functional programming because only functional programming, you know, takes the same input and produces the same output every single time.

Steve: Yup. Yup.

Leo: But it's really a great exercise. And I think ultimately it's a path forward. It's kind of the opposite end of the fuzzing. It's a path forward for making reliable code.

Steve: Yeah.

Leo: Yeah, yeah, it's interesting. Yeah, the fuzzing's the opposite end, where you just - you flail at it until it breaks.

Steve: Yeah. I think the thousand monkeys typing is the perfect visual analogy. And it's like, well, one of them may crash your program. And, if so, that's not supposed to happen. So then you'll see, well, how did you do that?

Leo: It's cool. It's really cool. And if you're a coder, you really want to look at some of these concepts of testing before you write anything, writing the tests. And then writing tests, there are test tools that let you kind of reverse fuzz before you distribute your code, which is really cool.

Steve: Yup. Typically referred to as "unit tests."

Leo: Yeah. Unit tests, another way to put it. And, boy, I wish I could remember some of these test commands. But there's somewhere you can just say, here's a range of tests. Bang on it as hard as you can. I think, I feel like that's a LISP feature, not a Racket feature. Racket's a scheme. Anyway, enough of that. Enough of that. Let's wrap it up because you've got some TV to watch, I think.

Steve: It's another...

Leo: Another Tuesday.

Steve: Another Tuesday. You betcha, baby.

Leo: Steve Gibson hides out at GRC.com. That's his Fortress of Solitude on the web. You can find so many great things there, like ShieldsUP! and his Perfect Paper Passwords, all about Vitamin D. But also his bread and butter, the thing that keeps him alive, which is SpinRite, the world's finest hard drive recovery and maintenance utility. Pick up a copy there. Check out SQRL. That's there, too. GRC.com.

You can leave questions for Steve at the website, grc.com/feedback. But he's also on Twitter at @SGgrc, and he takes direct messages, another place to comment or question Steve at @SGgrc. Steve has 16Kb versions of the show, low-bandwidth versions; 64Kb. He also has those great transcripts, so that's the only place where you can get a transcript of the show.

We have audio and video at our website, TWiT.tv/sn. You can watch us do the show live every Tuesday, 1:30 Pacific, 4:30 Eastern, 20:30 UTC. If you want to watch live, it's at TWiT.tv/live. There's also a couple of live audio streams there. And you can get on-demand versions of the show, as I mentioned, TWiT.tv/sn. Or the best thing to do is subscribe in your favorite podcast application, and that way you'll just get it automatically. Collect the entire set, all 757.

Thanks, Steve. We'll see you next week on Security Now!.

Steve: Thanks, buddy.

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/