SECURITY NOW!

Transcript of Episode #720

Bug Bounty Business

Description: This week we check in on the state of last week's Linux TCP SACK kernel panic, examine two Mozilla zero-days which were being used against Coinbase and others, and note that performing a full factory reset of an IoT device may not be sufficient. We look at a very clever and elegant solution to OpenSSH key theft via Rowhammer attacks, share an update on the BlueKeep RDP vulnerability, and examine the cause of a three-hour widespread Internet outage yesterday morning. We discuss NASA's APT, which crawled in via a Raspberry Pi, the cost of paying versus not paying a ransomware ransom, and an update on Microsoft's Chromium-based Edge browser. Lastly, we handle a bit of listener feedback, then take a closer look at the state of the commercial bug bounty business.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-720.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-720-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. There is a lot to talk about. Cyberwarfare, candy drops, and how the entire Internet got routed through Northern Pennsylvania for three hours. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 720, recorded June 25th, 2019: Bug Bounty Business.

It's time for Security Now!, the show where we cover your security with this guy right here. We've been doing it for 15 long miserable years, and things aren't getting any better. Steve Gibson. How are you today?

Steve Gibson: I would argue they're getting a lot worse, actually.

Leo: Yeah? Oy oy oy.

Steve: Been a lot of change. In fact, today's podcast is going to address one of those really interesting developments. This is Security Now! #720 for June 25th, and I wanted to talk about the bug bounty business from a slightly different perspective than we have before. We've talked about Zerodium, and we've talked about SandboxEscaper dropping hers, and we've talked about the Pwn2Own competitions. But there's a really, like 100% white hat hacker business underway that I wanted to shine a little bit of light on today.

This was prompted by some coverage of HackerOne in a recent article on ZDNet. And I thought, you know, let's talk about this because Zerodium, as you and I both have said, Leo, just sort of makes us a little queasy, you know, the idea that there's something, some entity is purchasing zero-days and paying some substantial coin, and would only be doing this if they were reselling them for profit to we don't know whom. You know? State actors, law enforcement, intelligence agencies globally, who knows?

So anyway, a lot happened. We're going to check in on the state of last week's Linux TCP SACK kernel panic. Mozilla's Firefox suffered and recovered from two zero-day exploits, which - and I'm a little disappointed because one of them they were informed of in April, and I don't know what they were doing. I guess they thought, well, by itself it's not a problem so we'll fix it when we feel like it. But when paired up with the second one, it became a real problem. So they suddenly went into overdrive and fixed those. Also there was an interesting story about even if you performed a full factory reset of a NEST cam, that that might not be sufficient before reselling it.

We also have a very clever and elegant solution to OpenSSH's key theft, which of course we talked about extensively when we talked about the RAMBleed Rowhammer attack last week. There's been a fix for it that I'll explain, which I think is very clean and clever. Also an update on BlueKeep, the RDP vulnerability. Also, I mean, as I said, lots happened. For three hours yesterday morning, well, yeah, morning our time, about 3:00 a.m. to 6:00 a.m. Pacific time, there was a quite widespread outage that affected a whole bunch of major Internet providers. We'll talk about the cause of that. This is something that we've touched on a few times in the past. But negligence actually on the part of Verizon was largely responsible, although they weren't the proximate cause. Anyway, it's interesting.

Also NASA discovered that they had had an advanced persistent threat in their network at JPL for a year as a consequence of a Raspberry Pi that had been connected to the network. Also some interesting data on the cost of paying or not paying a ransomware ransom. Also a quick update on Chromium, a little bit of listener feedback, and then we're going to talk about the way the commercial white hat bug bounty business has been growing sort of quietly behind the scenes, but it's something that definitely deserves a little bit of coverage. So I think a great podcast for our listeners.

Leo: Yeah. You said it was going to be short. Doesn't sound...

Steve: Yeah, that doesn't sound that short, does it.

Leo: That doesn't sound short at all. But we are always glad. The longer you go, the happier most of our audience is. I've never heard any - I've heard people say "That was too short." I've never heard anybody say "That was too long." Steve?

Steve: So our Picture of the Week is just simple and fun. I was just noticing as I was looking at it that...

Leo: And scary.

Steve: That the credit for who created it got cut off. I didn't cut it off. It was that way from the person who tweeted it to me, so I apologize for not knowing who did it. But anyway, the title is "CyberWar," and there's a big U.S. bomber flying through the sky, and a little balloon callout saying "USB key dropped." And then you can see a dotted line with a little USB key falling out of the bottom of the plane. And of course that's, as we know, in fact we covered it, that there had been some studies done of whether people would plug in random USB thumb drives found on the road, literally like, you know...

Leo: And the answer is yes.

Steve: Yes.

Leo: Oh, look. It's a 64GB drive free.

Steve: Ooh, nice. Yeah, that's right. Just plug it in and see what's there. It's like, eh. Okay. And just so our listeners know, don't do that.

Leo: No.

Steve: That's not good.

Leo: But it was used in cyberwarfare. We don't know by which government, but against Iran; right?

Steve: Stuxnet.

Leo: Stuxnet.

Steve: Stuxnet was believed to have been a so-called "sneakernet." Back before we had networks, sneakernet was floppies that you would carry from machine to machine. And then of course now we have thumb drives. And so yes, you're right, it was designed, Stuxnet was designed - because the centrifuges that were being used to concentrate plutonium were deliberately off the 'Net. They were isolated. There was no connection between them and the Internet, so there was no way electronically to get to them. So they said, okay, great, we'll just see if we can hitch a ride on somebody's thumb drive because you still have to have computers at the other side. Even if they're not dynamically networked, they're sort of virtually networked, thanks to moving data on thumb drives. So that's all it took.

Okay. So everyone may have noted, here we are, it's Tuesday. The Internet is still here.

Leo: What? I didn't expect that.

Steve: I know. It's very resilient, Leo, although it did take a bit of beating yesterday morning. We'll talk about that in a second.

Leo: Oh, really. Oh, okay.

Steve: Yeah. Last week's revelation about the multidecade Linux and FreeBSD - FreeBSD to a lesser degree. The normal configuration wasn't vulnerable. But there were three CVEs affecting the TCP stack, which as we know allowed any vulnerable listening Linux machine to be remotely attacked and halted with a kernel panic. And that has not so far resulted in any reports of widespread use and abuse. And again, the model that's evolving is that the hackers really do seem - this is not surprising, I guess - to be interested in money. And we've sort of moved past the point where, oh, look, this is fun to do, like which is what we used to have in the early quaint days of viruses; right? When viruses were propagating, and we were always sort of saying, well, what's the motivation here? It was just to sort of see if you could.

Now it's about money, which is of course why what we're seeing is cryptocurrency miners are being installed and ransomware is not going away anytime soon. We'll have a story about that. So I guess it's not that surprising that there's not any widespread use of this flaw for crashing Linux servers. I mean, like, okay, that doesn't make anybody any money except maybe in specific targeted instances, and those probably don't make the news. It's some random single server that crashes, and it's like, oh, okay. Reboot. So anyway, nothing much has come of that. I just sort of thought I would follow up.

Oh, but there was one piece of nice work, a script that was produced by some guys at SentinelOne. It's up on GitHub for anybody who wants it. It's a free script, obviously, for Linux systems to both detect and protect the system from this problem. That is, and you would use it before updating. If for whatever reason you had a system that was exposed, you were aware of this problem, but it wasn't convenient for you to update the system, to have the server down at all and so forth. You can run this script. It makes a backup of the things it's changing so that you are able to revert it, if for any reason you want to. It makes changes which do survive a reboot. So unlike the little quick hack we talked about last week, the sysctl command that would transiently but not persistently solve the problem, this does.

So it's GitHub.com/sentinel-one. And then once you're there you can see their sack-cve-fixer, as they call it. And it's just a script you can run on Linux. And it's got separate modes, but there is a check separate from the install. And so it would certainly be handy. This does a safe check to see whether your system has been patched, to confirm it, and/or whether it's vulnerable. So even applying this fix, that is, using the install version, if you then run the check command afterwards, it says, oh, you're not vulnerable. So it will basically check for vulnerabilities. So might come in handy for anyone who's got a Linux system exposed to the public Internet.

And my favorite browser stumbled a little bit last week. The good news is that zero-days in Firefox have been extremely rare. Maybe that's because it's not as big a target as Chrome and Microsoft's browsers. But the most recent previous Firefox zero-day was more than three and a half years ago, whereas we're pretty much covering zero-days in other browsers as often as not. That one back in December of 2016 fixed a flaw that was at the time being abused to expose and deanonymize users of the Firefox-derived Tor browser.

But the bad news in this case, as I said at the top of the show, was that there were two zero-days being paired which were discovered being actively exploited against employees at Coinbase, the cryptocurrency exchange. And actually they have evidence now of it being used against others, as well. Philip Martin, who is a member of the Coinbase security team, reported the attacks to Mozilla. And he said Monday a week ago, so Monday the 17th, he said: "On Monday Coinbase detected and blocked an attempt by an attacker to leverage the reported zero-day, along with a separate zero-day sandbox escape, to target Coinbase employees."

And as I mentioned, what's a little disturbing is that Mozilla had been privately informed about this a full two months before, on April 18th, when Samuel Gross, a security researcher with Google's Project Zero, said that he reported this first of the two zero-days, and apparently Mozilla didn't do anything about it. So it's when the Coinbase team reported the same bug being used in the attack against them that Mozilla said whoops and immediately pushed out a fix. We're at Firefox 67.0.4, and that's after two updates. They did one for each zero-day because they also only did - they only updated them individually. So the problem was on 67.0.2 and earlier. They fixed the first zero-day by going to 67.0.3, and then the second zero-day by going to 67.0.4.

So Philip said: "We walked back the entire attack" - that is, after they realized what was going on and caught it. He said that, if successful, the attackers could have gained access to their backend network and used this access to steal funds from the exchange, which is of course an outcome that we've been seeing recently because there have been lots of losses suffered by various cryptocurrency exchanges. Once again, this kind of comes back to our noting that now it's gotten to be all about money these days.

He said: "We walked back the entire attack, recovered and reported the zero-day to Firefox, pulled apart the malware and the infrastructure it was using for the attack." He said: "We are working with various organizations to continue burning down," he wrote, "the attacker's infrastructure and digging into the attacker involved." He wrote: "We've seen no evidence of exploitation targeting customers." And he also added that other cryptocurrency-linked organizations have been targeted by the group, and they are being notified.

So the outcome was all good. What's, again, disturbing is that I don't understand why Mozilla didn't jump on the report of a zero-day mid-April, which would presumably have prevented this from happening because this required the combined use of two zero-day exploits. But for whatever reason, it's fixed now. And so we are all at 67.0.4 and not subject to those targeted attacks.

Leo: It sounds like the attack, maybe you know more detail, primarily was able to get the password stored within the browser.

Steve: That's correct, yes. So they were...

Leo: So that's a great attack, but it's a great reason, I mean, I would, I'm sure you agree, never use a browser as a password manager.

Steve: I really do. In fact, I mentioned, I think it was two podcasts ago, I have a very good friend who's an ex-Microsoft employee who's involved now in the cryptocurrency business, who called me a couple weeks ago asking if I had contacts with the FBI because his Google account was hacked. That allowed them to log into Chrome as him. And once they synced, they had all, I mean, his entire username and password list that was all in Chrome. And so, yeah, it's really handy to have your browser remember this stuff for you. But as we've seen, in fact I told the story months ago that I was trying to log in on something to help Lorrie log in, and she didn't know what her password was. I said, oh, let's go find out.

Leo: Just look in Chrome.

Steve: Exactly.

Leo: It's not, you know, it's not obscured. You just - yeah.

Steve: No. It's right there. And her mouth hung open. She says, "Oh, my god, there's all my passwords." I said, "Uh-huh, yeah."

Leo: Well, you have to be logged in. I mean, that's Google's explanation is, well, once people are logged into your account, then you're kind of in trouble anyway. But most password managers will not do that. Most. And that's why browsers are not a good place, I think.

Steve: Yeah, yeah.

Leo: I use a YubiKey with my LastPass because I don't want anybody to get into that, you know.

Steve: Yeah. And, you know, I had a problem with LastPass a couple months ago where the authenticator stopped working.

Leo: Oh.

Steve: Yeah. It would, like, it would prompt me, and then immediately - it wouldn't wait for me to enter anything, would immediately close. And that has been a problem that, I mean, I immediately, you know, after removing it and starting it up again and so forth, I couldn't get it to fix. And when I did some googling, it turns out that's a problem that people have had before.

Leo: I've had that problem with Google, as well, that for some reason - and I think it's just kind of the weird nature of using a USB authenticator with software, that maybe sometimes the software doesn't - because Google, same thing. I put my key in, pressed the button, said we didn't see it. What? That's frustrating. And maybe we need better hardware authentication methods.

Steve: But anyway, you're right. What they were doing was they were using this to get out of the sandbox, to get into the browser, to get access to all of the stored passwords. And of course they were hoping that, or maybe they knew, that the people they were targeting were Firefox users. But that was the nature of the attack was to try to get a hold of passwords stored in the browser, which as we've just said is not a really good place to keep your passwords.

Interesting story, it was in, oh, it's Wirecutter who ran this. And before I go any further, I'll make it clear that this was a mistake that Google did promptly fix, pushed out an update, and the problem has been quickly resolved. But for some time before Google was made aware of the problem, and we don't know how long this has been the case, it was true that someone purchasing a previously owned and fully factory reset Nest Cam could still be watched by the camera's previous owner.

Last Wednesday, Wirecutter ran the story with the headline: "Buyer Beware: Used Nest Cams Can Let People Spy on You." They said: "We've explained before that when you're selling or giving away your old smart home devices, it's critical to do a factory reset on them first in order to protect your data and privacy." You know, very much like we would wipe our phone or wipe a hard drive when giving a computer away. They said: "We've recently learned, however, that even performing a factory reset may not be enough to protect privacy for owners of the popular Nest Cam Indoor. And in a twist, this time the risk is on the side of the person receiving the device, not the person disposing of it."

They said: "A member of the Facebook Wink Users Group discovered that, after selling his Nest Cam, he was still able to access images from his old camera, except it wasn't a feed of his property. Instead, he was tapping into the feed of the new owner via his Wink account. As the original owner, he had connected the Nest Cam to his Wink smart home hub. And somehow, even after he reset it, the connection continued."

They said: "We decided to test this ourselves and found that, as it happened for the person on Facebook, images from our decommissioned Nest Cam Indoor were still viewable via a previously linked Wink Hub account; although, instead of a video stream, it was a series of still images snapped every several seconds. If you buy and set up a used Nest Indoor camera that has been paired with a Wink hub, the previous owner may have unfettered access to images from that camera. And we currently don't know of any cure for this problem. We are unsure what further implications there may be regarding Nest's video service, including whether it may be vulnerable to other methods or through other smart home device integrations. We're also unsure whether this problem affects the entire Nest lineup, including the Nest Cam Outdoor, Nest Cam IQ Outdoor," blah blah blah.

Anyway, they finish, saying: "According to Google, the issue has now been fixed." And actually, they also, in an update to their initial reporting, retried all of this and confirmed that it had been immediately fixed and pushed out. They said: " When we asked about how the company corrected the error, a representative said: 'We usually don't share how a fix was pushed out for various reasons. The statement is our update of record for this.' Although this particular bug has been fixed, we advise anyone interested in smart home gear to be especially cautious when considering buying or selling used items, especially ones that have the potential to interfere with your intimate privacy and security, such as cameras, devices with microphones, and smart locks."

And it's sort of chilling. I mean, when you think about it, what's going on with the cloud and with connectivity is a black box. I mean, we get these things. We follow the instructions. Everything is kind of done for us; right? It's automatic. It's scan the QR code on the base of the device or press a button, spin around three times, click your heels, and do two Hail Marys. And, oh, look, it's all connected, and things are linking up, and lights are blinking. I mean, we have no idea what is actually going on. And so we have no control over these systems, and it's very clear that, in this case even, I mean, the device has probably some, obviously has some factory burned-in serial number that is not being randomized by a full factory reset. Something somewhere held onto that.

Leo: Well, but - so I'm thinking, because it's associated with the Wink, right, which is a third-party device, it only did it if you used a Wink Home Hub with your Nest Cam, is my understanding.

Steve: Okay.

Leo: So what I'm thinking is the problem, and of course we don't know because they won't say, is not with the Nest Cam particularly, but maybe with the Wink? That the Wink authenticated the Nest Cam via, as you say, a hardwired serial number or something that identified it, that a reset of the camera alone wouldn't reset. But when you then put that same camera on a Wink somewhere else, it would go, oh, I know you.

Steve: Yeah. Although...

Leo: Do you think it was like that?

Steve: Although Google did fix it. So...

Leo: So that's true, yeah.

Steve: They, like, said, "Oh, crap," and immediately, you know. So it was something that they were doing that, I mean, probably...

Leo: Yeah, okay. But it did require the complicity of a Wink hub. So I don't know. You know?

Steve: Yeah, yeah.

Leo: Maybe they called Wink and say, you know, "We're going to change it on this end."

Steve: That may be. Or maybe all of the cameras do feed through Google's cloud and then go back out. I mean, again, see, and this is my point: It's magic.

Leo: It is. That's the problem.

Steve: Once upon a time - yes. Once upon a time, back when you and I had, well, actually you still have hair.

Leo: Yeah, hey.

Steve: There was, okay, there was more pigment in our hair.

Leo: Yes.

Steve: The connections between these things were explicit.

Leo: You understood them, yes.

Steve: Yes. We knew what was going where. And now it's like, oh, I just pressed a button. Look, it works. It's like, yeah. But what happened? No one knows.

Leo: Well, and Stacey Higgenbotham and others have written long articles about how complicated and sometimes frustrating it is to decommission your smart home before a move. Stacey just moved, and it was a big deal. There was a lot to do. I think Georgia Dow is writing the same kind of pieces. It's just complicated. It's not as easy as it sounds.

Steve: Yeah, I would get one of those - they're not really steamrollers. I don't know why they call them "steamrollers." I guess maybe once upon a time they...

Leo: You and I call them "steamrollers" and "steam shovels." And they haven't been steam in a while. Not even in our lifetime.

Steve: Put all of your IoT stuff down in front of it and just roll it forward.

Leo: That's what I think. Do not resell your IoT stuff.

Steve: Because what are you going to get, 30 bucks for something? Or 20 bucks on eBay? It's like, not worth it. Just enjoy the sound of the crunching plastic. Just...

Leo: I have one of those Nest Cams right here, right behind me.

Steve: Yeah, I wonder, yeah. Oh, well, okay.

Leo: So they're accessing the old recordings, obviously, not, I mean, because you don't have the camera anymore, so it's the old stuff that they're finding.

Steve: No, no, no. It was fresh. It was the buyer's feed taken as a series of stills.

Leo: From a different camera. The buyer. Oh, I get it. So the older owner could see the new camera's output.

Steve: Correct.

Leo: Got it, got it, got it.

Steve: Correct, yup. And it's funny, too, because Mark Thompson and I were chatting a couple weeks ago, and he just happened to talk about the Vector, the cute little kind of steam shovel robot thing. And he had talked about...

Leo: Which also is out of business, by the way.

Steve: Which, yes. These little home robotics things...

Leo: They go fast.

Steve: They're not lasting very long.

Leo: No. This was cute, too. I loved it.

Steve: It was. It was. And to her credit, Lorrie, when I explained that it had a camera, and actually it was doing facial recognition - because you could say, "Hey, Vector, who am I?"

Leo: Right.

Steve: And it would [sound effects] and say "Lorrie." Anyway, she was very uncomfortable by the idea that this thing had a camera in it. And I said, "Well, honey, it's on the kitchen counter. It's not like it's in the bedroom or anything."

Leo: It just shows you normal people have good instincts.

Steve: Yes, exactly. It's like, eh, let's give this to somebody else. So anyway. So I guess the real takeaway, and I'm glad we talked about this, Leo, because it's worth noting that things that we don't understand - and really, I mean, it's been taken from us, like any sense of what's going on. It's like, ooh, magic. And it's like, yeah. But, boy, are you having to trust all kinds of, well, how can you trust? I mean, you can hope. And, you know, we believe that Google would never intentionally do this.

Leo: Oh, I'm sure not.

Steve: But, you know, all those streams are going off to some cloud somewhere, and so, yeah. I mean, you don't have control over it.

Leo: And it's hard to do it perfectly, as we've - this show is essentially the illustration of that.

Steve: And we bring it to you gleefully every week. If this doesn't raise your blood pressure...

Leo: On with the show.

Steve: So we talked about, last week, the exfiltration of very sensitive keys, which was done as a proof of concept by the RAMBleed guys. And so what they did, remember, was that by figuring out how they could pound on their own memory after noticing that the probability of flipping their own bits was influenced by the bits on either side, they figured out how to bring another process's secret into alignment in the DRAM grid so that copies of it were on either side of the row of memory they control. And then by examining the probability of their own bit flips, they were able to infer the bits of secret data. So that's not good.

And what they did, as I said, as a proof of concept was they exfiltrated an OpenSSH key from a process they didn't control that was sharing their same server as theirs. So last Thursday, June 20th, Google security researcher Damien Miller, who's one of the top OpenSSH and OpenBSD developers, added protection against any and all forms of side-channel attacks by leaving the OpenSSH private keys encrypted until they are needed.

And what's weird is that last week, last Tuesday, in the context of talking about RAMBleed, I was just talking about exactly this strategy. I noted that my own SQRL client for Windows never leaves its keys in the clear in RAM. The user's password, you know, the one password you use to tell the SQRL client that you are you, that is used to transiently decrypt the keys briefly for the transaction, after which the plaintext decrypted versions are zeroed in RAM so that they are never sitting around available to be exfiltrated. This is clearly the right way to design code for maximum safety in a hostile environment.

So the good news is the OpenSSH project has received this protection. According to Miller, OpenSSH will encrypt the secure shell private keys while they are at rest inside a computer's RAM. If an attacker manages to extract data from a computer or server's RAM, they will only obtain an encrypted version of an SSH private key, rather than the cleartext version. He indicated that this protection would be able, and I agree, to stop all manner of side channel attacks - Spectre, Meltdown, Rowhammer, RAMBleed, and so forth. And it's an interesting commentary that we now have so many side channel attacks available.

But what I thought was so cool, Miller had an additional challenge that I did not have when doing something similar with my own SQRL client design. The advantage I had was that the knowledge of the key to decrypt the master did not need to exist in the SQRL client since the user would be the one providing that missing data when it was necessary to perform the decryption. But that wouldn't work for OpenSSH because the system must be able itself to autonomously decrypt the SSH keys on the fly without user input whenever they are needed.

Miller's solution was clever and elegant. The comments in his code commit read: "This change encrypts private keys when they are not in use with a symmetric key that is derived from a relatively large prekey consisting of random data, currently 16KB." So 16KB of data. I guess it's bytes. That would make more sense than bits because you want to have a lot.

He said: "Attackers must recover the entire prekey" - and I put in brackets "perfectly," and I'll explain why in a second - "before they can attempt to decrypt" what he calls, he uses the term "shielded," as he kind of invented a new term, "the attempt to decrypt the shielded private key. But," he says, "the current generation of attacks have bit error rates that, when applied cumulatively to the entire prekey, make this unlikely." And I would say way more than unlikely.

I didn't dwell on the details last week. But even in their research, the research paper, that successful recovery of the OpenSSH key, which was used to demonstrate a successful RAMBleed attack, it strongly depended upon the use of some very powerful algorithmic post-processing to perform bit guessing among the always somewhat uncertain recovered bits. In other words, RAMBleed returns bit probabilities, not bit certainties.

So it turns out that there are some clever algorithms that can use the known properties of the relationship between the public and private key components. Well, for example, we know that a private key is the product of two primes. So it turns out just knowing that allows you to immediately exclude many bit combinations that break that assumption. So they were able to use these algorithms to essentially turn the probabilities that they were getting from RAMBleed into enough certainty that they were ultimately able to recover the OpenSSH key.

But none of that would work against a purely random 16KB prekey, where every single bit has to be exact because it's high random. I mean, it's pure entropy. There's no assumption of interbit relationships, which you do have with an RSA key used by OpenSSH. And presumably he takes this - I didn't look at the code, but he probably takes this 16KB prekey and then hashes it in order to get the key, which is then used to statically decrypt the OpenSSH key on the fly.

So it's just - it's brilliant. Basically, he very cleverly leverages the fact that any of these RAMBleed-style side channel attacks, whatever they are, they're basically getting statistical guesses of bits, and at a relatively low bit rate, and by locking the decryption of the in-RAM OpenSSH key to a large prekey where not a single bit can be off, or when you hash it you're going to get, as we know, something completely different. And that will not symmetrically decrypt OpenSSH. And this is all still very fast. So you can afford to do it on the fly.

Anyway, very cool solution. And that has been committed into the OpenSSH project as of last Thursday. So it is available to anyone who updates their builds of OpenSSH. And I'm sure it'll be moved into various ports and available in Linuxes and Unixes and so forth. Anyway, he somewhat modestly calls this elegant process "shielding." And he said: "Implementation-wise, keys are encrypted shielded when loaded, and then automatically and transparently unshielded when used for signatures or when being saved or exported." Oh, and he also noted, he says he hopes they'll be able to eventually remove this special protection against side channel attacks in a few years, when computer architecture has become less unsafe.

So he recognizes, and we've talked about this, that we're sort of going through this awkward phase where the cleverness that we were using to accelerate our systems turned out to have lots of little edge cases, as we've been talking about now for about a year and a half, ever since Spectre and Meltdown hit. Well, actually even before that because the DRAM attacks, the Rowhammer attacks, that goes back to at least 2017 or earlier. So anyway, just a very cool piece of work. And it was nice to see a response from the developer community that gave us a really practical solution to these problems.

So a BlueKeep patching status update. Of course we know that BlueKeep is the very bad authentication bypass attack against RDP. Despite the fact that everybody has been calling it "wormable," because it is, we've seen no worms. And I've argued that we probably won't because you don't need a worm. There is a worm against the Exim email server vulnerability because it takes a week of camping out on an email server to cause the right combination of timeouts and retransmissions in order to execute commands on the vulnerable server.

So it totally makes sense that you'd have a squad of worms out there looking for vulnerable Exim servers, setting up long-term persistent connections, and then waiting a week while they dribble out a byte every four minutes to keep the connection from getting dropped over the course of that time, and then getting their exploit to happen. And that is in, I mean, we know from last week that that is happening.

So where are we with BlueKeep? The Patch Tuesday of May, when this happened, was - what was Patch Tuesday? I don't have the date in front of me. Anyway, it was Second Tuesday of May. By the next to last day of May, so 16 days after that Patch Tuesday, Raviv Tamir, who is the group program for Microsoft's threat protection, 16 days after Patch Tuesday he tweeted: "My dashboard remains bleak, as only 57% of exposed machines I see worldwide have patched for CVE-2019-0708," which is this RDP exploit. And he said, in all caps: "GO PATCH."

Okay. So then the following Wednesday, June 5th of this month, Raviv updated his numbers, tweeting "Numbers are going up, now at 72.4% worldwide. That's better, but still not good enough. KEEP PATCHING," in all caps. And then this past Thursday, June 20th, so another 15 days after the previous one, he updated again with a tweet: "Another update. Worldwide update rate for CVE-2019-0708 numbers are up to 83.4%." And then "KEEEEP PATCHING" was his final.

Leo: Like the people who aren't patching are reading him.

Steve: Exactly. Exactly. It's like, okay. I think, I mean, what we're seeing is probably Microsoft's updates are hitting machines that are not rebooting very often. And so they're rebooting, and they're doing their updates, and then they're disappearing from his radar.

Leo: I'm glad there's progress being made. That's a good sign.

Steve: Yes. That is a good sign. BleepingComputer reached out and asked Raviv about the number of computers that continue to be vulnerable to BlueKeep. And what he's looking at is his Microsoft Defender ATP, the Advanced Threat Protection network that they've got. And he said, while it is a lot better, there are still several million machines out there. And as we know, DHS confirmed that even Windows 2000 is vulnerable to this. Well, it's not getting any updates. And Microsoft did make that Windows XP update available, but that's not auto updating; right? You've got to go get that. So there are, like, old, creaky systems that are not going to get themselves fixed. And they're going to get themselves owned here, probably, before very long.

Okay. Yesterday morning, 3:00 a.m. Pacific time to about 6:00 a.m.

Leo: This is my favorite story.

Steve: Oh, Leo.

Leo: Oh, man.

Steve: Yeah. About 2% of the global Internet was mistakenly routed through a Pittsburgh, Pennsylvania steel mill.

Leo: Say that sentence again. How much of the Internet?

Steve: Yeah, about 2%. Think about that. One out of 50. Two percent of the Internet was mistakenly routed through a steel mill in Pittsburgh, Pennsylvania. And of course that didn't turn out so well. Okay. So first of all, to get a little - okay, wait. I'm going to share what The Register said because they're always irreverent.

Leo: Snarky, yeah, yeah.

Steve: Yeah, snarky. I mean, they redefine the term "snark." They said: "It all started when new Internet routes" - no, actually I should explain first so that we have a context. So remember that the way that the global Internet works - and this is not our routers, our little NAT routers that we have at home. They share the term "router" with the so-called "big iron routers" that are out on the Internet, actually moving all of this traffic around between ISPs. They've got multiple interfaces on them with packets coming in and going out. And every one of these big routers is a routing table that basically knows, for any packet that comes in, which interface to forward the packet to, that is, like, towards its destination.

So a router that belongs to Level 3 will receive a packet, and it's connected to a bunch of other high-level ISPs, you know, backbone, Internet backbone Tier 1 providers. And so it looks at the destination IP. And we know how Internet routing happens where the most significant bits are the network, and the least significant bits are the machine on the network. And there's a mask that is used in order to figure out where this should go. So the routing table does that work. It looks at the destination IP, consults its table, and then says, okay, the ultimate destination is somewhere down that wire.

And so it puts the packet on the wire and off it goes, and it's done its job. And so the recipient router does the same thing. The idea is the packet keeps getting closer and closer - well, actually, hopefully, not in this case, but normally - until it finally gets to its destination. So the way these tables are managed, because you can't do this by hand, I mean, it's just not possible, is this protocol that we've talked about from time to time. And it always comes up when something breaks like this: BGP, Border Gateway Protocol. That's the communication among the routers which allows them to share their routing tables and to pass updates about the way the Internet is connected essentially among themselves.

And the term that we'll hear as we talk about this is "advertising a route." It's kind of a weird term. But the idea is that a router that has a customer connected to it will advertise that block of IPs to its peers, to its peer routers. And that informs them that it should receive traffic for its customer, for the littler guy that's hooked to it. So when any of the peer routers receive a packet with that network, bits at the high end of the IP, they should send it to it; right? So again, it advertises the things it wants to receive, on behalf of its customers, to its peer routers. And it uses the BGP protocol to do that.

Okay. So you could imagine the problem if a router for some reason advertised a whole bunch of networks that it didn't actually have a right to advertise. That is, it didn't actually have customers connected to it that it could route that to. That is, if it were advertising networks that belong to other people. That's really bad.

So The Register writes: "It all started when new Internet routes for more than 20,000 IP address prefixes" - now, that is to say 20,000 IP networks, right, roughly 2% of the Internet - "were wrongly announced [advertised] by regional U.S. ISP DQE

Communications. This announcement informed the sprawling Internet's backbone equipment to" - this is The Register I'm reading - "to thread netizens' traffic through DQE and one of its clients, steel giant Allegheny Technologies, a redirection that was then, mind-bogglingly, accepted and passed on to the world by Verizon, a trusted major authority on the Internet's," as they write, "highways and byways.

"This happened because Allegheny is also a customer of Verizon. It, too, announced the route changes to Verizon, which disseminated them further. And so systems around the planet were automatically updated; and connections destined for Facebook, Cloudflare, and others" - and actually a lot of others, including Amazon - "ended up going through DQE and Allegheny, which buckled under the strain." No doubt. No kidding.

Leo: Because Cloudflare, is it their traffic, their Internet protection traffic, all of that's going through there? I mean, Cloudflare's not just, like, your average Joe on the street.

Steve: Right. They have 20 million websites.

Leo: Yeah.

Steve: And so many of their customers were advertised as being located at Allegheny Steel Plant in Pittsburgh. I mean, so that's exactly what happened. This said to Verizon that Cloudflare's and Amazon's and Facebook's customers, or their networks, were actually located here in Pittsburgh. So send all the traffic there.

Leo: It's got to be frustrating because Cloudflare - and I know their CTO, John Graham-Cumming, and I know how committed they are to what they do.

Steve: Oh, yes, quality service.

Leo: And I had chills when I saw Cloudflare is down. This was before we knew why. I thought, that, of all the companies in all the world, that's the last company I'd ever expect to have an outage. Even less so than Google.

Steve: And think about it. There was nothing they could do.

Leo: Exactly the point, is that's got to be terrifying to them.

Steve: Yes. And in fact he was rather miffed. John Graham-Cumming was tweeting, but also Matthew Prince, the CEO?

Leo: CEO, yeah.

Steve: Yes. He tweeted: "It's networking malpractice that the NOC" - that's the Network Operations Center - "the NOC at Verizon has still not replied to messages from other networking teams they impacted, including ours, hours after they mistakenly leaked a large chunk of the Internet's routing table."

Leo: Just horrible. Just horrible.

Steve: So here was, what, like, people were frankly astonished that Verizon would not have filters because what came up their wire from their customer, because this Allegheny is dual-homed - meaning that Allegheny was served both by Verizon and DQE. So DQE was actually the source of the problem. DQE was running a BGP optimizer, which is some software designed to improve their routing tables. Well, it went wacky. It, like, broke. And so it provided a bunch of these bad routes to the customer they share with Verizon, Allegheny, which then forwarded them to Verizon.

Well, Verizon should and could absolutely know that these were bogus, that there was absolutely no way that Allegheny could own these routes. And so this is called "route filtering." And it should have been in place. And people could not believe that a major Tier 1 entity, backbone Tier 1 provider like Verizon would blindly accept these routes and propagate them out onto the Internet. And they're inherently trusted, so everyone believed them and sent all the traffic there.

Leo: This is the ThousandEyes diagram of that happening. You know, our sponsor, ThousandEyes, this is what they do is they monitor this stuff. Do you think Cloudflare - user locations on the left side are experiencing high packet loss when trying to reach the Cloudflare CDN. This is all the packet loss here. Do you think Cloudflare looks at that and goes, oh, a BGP leak? Or is it harder to figure it out?

Steve: I looked at the timeline from John, and it looked like it took him about maybe 15 to 20 minutes, which is really pretty quick. Because, I mean, it could be - the problem could be anywhere. But we would argue that, and we do, always say anybody can make a mistake. You know? Stuff happens. So, yeah. That could happen. But what is unconscionable is that there was nobody at Verizon responding. That is, I mean, if you're Verizon, and you're a trusted Tier 1, you have to have somebody who picks up the phone when Cloudflare calls.

Leo: Yes. Yes.

Steve: And says, "You know, you're advertising a bunch of routes that are ours. Knock it off." And it took quite a while for that to get cleared up.

Leo: A great piece on ThousandEyes about kind of the whodunit. And I'm just shocked that this is still possible. That's the thing that blows my mind. How can this - we've known about BGP leaks for, I mean, we've been talking about this for a decade.

Steve: Yes. Forever. It has always been possible. The problem is we have a bunch of infrastructure in place, and it works kind of like well enough; you know?

Leo: Yeah, yeah. Well enough.

Steve: It's like when a sinkhole opens in the middle of an intersection in a major city, and cars fall in. People go, oh, my god, our infrastructure is breaking. But, you know, they plug up the hole, and everyone keeps driving. No one actually fixes the underlying problem. We just sort of go, oh, well.

And so on the Internet we have a problem. I mean, this is a, you know, and here we are. Once upon a time when it was email, well, email didn't care. It would just wait until the connections came back up, and then your mail would get delivered a little bit later. And you'd click refresh and be thinking, oh, you know, they said they sent me that email. Then it would come through, oh, look, your email came. Well, that's not good enough anymore because imagine the serious business use of the Internet and what an outage for three hours does if you're doing real-time stock market trading or managing a nuclear reactor. Oops. And, you know, it could actually matter.

Leo: This is Chernobyl-level incompetence on the part of Verizon, honestly.

Steve: It really is.

Leo: This is the Cloudflare blog. "A BGP session can be configured with a hard limit of prefixes to be received so the router can decide to shut down if the number goes above that threshold. Had Verizon such a prefix limit in place" - which is, I gather, a standard practice - "this wouldn't have occurred. It doesn't cost a provider anything to have such limits in place." And here's the payoff quote: "There's no good reason other than sloppiness or laziness that they wouldn't have such limits in place."

Steve: Yup. You absolutely want to put sanity testing on BGP. And it's easy to do. So the router says, whoa, I mean, here was 20,000 new network prefixes, bang, that arrived. And it said, oh, okay. What? You know, that never happens. Yeah, yeah. So anyway, I mean, the good news is each of these events raises the pain threshold. And so what we have, I mean, even the idea that we're rate-limiting BGP updates, well, that's clearly a patch. I mean, that's not the right solution, to rate-limit the updates. It would have stopped this, but it wouldn't keep, for example, a single malicious update, a deliberate malicious update, from happening.

So we need a system that is able to properly verify the changes that are always occurring as one entity moves from one ISP to another, for example. And we don't have that yet. So we'll get there. I mean, we have a system which works, and it's amazingly resilient when you consider how long ago it was designed and how well it has withstood everything that we've been throwing at it. But it's not perfect.

Leo: The Register said normally if a little company in Pennsylvania announced it owned the Internet, you would have some protections in place. You'd filter that out. But for, well, how many hours? Was it eight hours? It was a while. This was...

Steve: No, no, it was only three hours.

Leo: Okay, only three hours.

Steve: From about 3:00 a.m. to 6:00 a.m.

Leo: The longest three hours of John Graham-Cummings' life.

Steve: Boy, yeah.

Leo: Oh, god.

Steve: NASA, as I mentioned, was infected by an APT for more than a year. They've tracked it down to an unauthorized Raspberry Pi, which someone at JPL, actually, was where the problem was, connected to their network.

Leo: It was probably monitoring the coffee pot or something.

Steve: Yeah, exactly. It was probably doing who knows. Because, you know, JPL is a freewheeling place. I mean, they do beautiful work, but they're, you know...

Leo: No, I don't want to have to go down the hall to see if the coffee is fresh. I'm just going to put a Raspberry Pi in here.

Steve: And we'll stick a little webcam on the Raspberry Pi so I can just get a picture of the coffee pot and see what the water level is.

Leo: And, oh, by the way, we've got a network. I'll just put in the network.

Steve: Why not, Leo?

Leo: Why not? Why not?

Steve: Yeah. So in a report published last week by NASA's OIG, the Office of Inspector General, in that report it revealed that in April of 2019 hackers breached the agency's network and stole - they did, I mean, this was an active threat - stole approximately 500MB of data related to Mars missions. Because of course that's what JPL does. The point of entry was a Raspberry Pi connected to the network at, as I mentioned, JPL, without authorization or going through the proper security review. I mean, they have systems in place where you're supposed to log and have approved anything connected to the network. But hey, look, there's an RJ45 port. Let's plug in the Raspberry Pi.

According to this 49-page OIG report, the hackers used this point of entry to move deeper inside the JPL network by hacking a shared network gateway. The hackers used this network gateway to pivot inside JPL's infrastructure and gained access to the network that was storing information about NASA JPL-managed Mars missions, from where they exfiltrated information. Quoting from the report: "The attacker exfiltrated approximately 500MB of data from 23 files, two of which contained International Traffic in Arms Regulations information related to the Mars Science Laboratory mission." The

Mars Science Laboratory is the JPL program that manages the Curiosity rover on Mars, among other projects.

JPL's primary role, as we know, is to build and operate planetary robotic spacecraft such as the Curiosity rover and various satellites that orbit planets in the solar system. JPL also manages NASA's Deep Space Network, which is the worldwide network of satellite dishes which are used to communicate with NASA spacecraft during their missions. Investigators said that, besides accessing the JPL's mission network, the April 2018 intruder also accessed the Deep Space Network's IT network. Upon discovery of the intrusion - I got a kick out of this - several NASA facilities immediately disconnected from JPL and DSN. It was like, agh!, you know, just pull the plug. Do not connect to JPL, those crazies out there in California, in Pasadena. We don't know what they've got crawling around their network. And they disconnected fearing, of course, that the attacker might pivot into their systems, as well.

NASA's OIG said: "Classified as an advanced persistent threat, the attack went undetected for nearly a year, and the investigation into this incident is ongoing." The report blamed, and here it is, JPL's failure to segment its internal network into smaller segments. And of course, as our longtime listeners know, we've talked about the need for strong network segmentation for several years, even at home and in small offices. The reason we liked that amazing little Ubiquiti five-port router so much was that for $49 they contained five entirely separate NIC interface adapters, which supported strong network segmentation. The downside was that they were a beast to configure. They had a bizarre configuration language. But many of our listeners were able to get them working and use them in that way.

But of course the reason JPL probably didn't bother with network segmentation is the same reason that most home and small offices don't bother. It's a pain in the butt to maintain separate networks. It's so much easier to have everything able to talk and see each other on the network. But of course that's exactly the problem because, once anything malicious gets onto your network, then it also can talk to and see everything else that it then has access to. So, not good.

I said I would talk about, and there's some fun data here, about the status of ransomware because what we are seeing is that hackers are now pursuing money. Riviera Beach, Florida, I think it was - where was it? I have the data here. Ah, last Monday evening. Okay. Riviera Beach, Florida, is coughing up $600,000, Leo, to hackers after a ransomware attack brought down its computer system.

Leo: It's a town of 35,000 people.

Steve: Yes. Small town.

Leo: Like 20 bucks per person.

Steve: Small town hit by a ransomware attack at the end of May, on May 29th, after a city employee clicked on a malicious link in an email, according to local reports. Attackers behind the malware, which then spread throughout the city's network and shut down its computer systems, asked for a ransom of 65 bitcoin. And it's funny because, well, not funny. But I meant to go look and see what that would be worth now because I'm not sure when that price was set. But bitcoin has been enjoying a recent resurgence. That's like more than $11,000 per coin. So I'm not sure what, I mean, 65 bitcoins is going to be more than 600,000. Anyway, in exchange for unlocking computers.

Basically, the systems controlling the water utility were offline. Government email and phone systems were nonfunctional. 911 emergency calls could no longer be entered into computer databases. According to local reports, the computer systems controlling city finances and water utility pump stations are now at least partially back online. So last Monday evening the City Council voted unanimously to authorize its insurer to pay the $600,000 ransom. So they were insured.

The security community, for its part, has argued that the city is taking a big gamble. Remember that we're dealing with criminals here; right? So the fact that they make the payment doesn't necessarily mean that they're going to get their data back. On the other hand, it's useful to remember also that future victims' willingness to pay ransom will be a function of this outcome. So the bad guys who receive $600,000 as a consequence of some city employee clicking on a link, it's in their interest to provide the data to allow the city to recover their computers because these bad guys want to get paid rather than not.

The technical program manager at HackerOne, and we'll be talking about HackerOne at the end of the show, said the Riviera Beach City Council has taken a big gamble by paying the ransom as there are no guarantees the attackers will return any of the data, which could leave the city in an even worse situation. "By paying the ransom," this person writes, "the council also encourages more of these types of attacks as it makes it more profitable for attackers." And of course the FBI is formally on record saying do not pay.

Well, that's easy for the FBI to say. Last year, several Atlanta, Georgia city systems were crippled by a ransomware attack. A ransom of $51,000 was demanded for recovery. Atlanta said no thanks, and ended up spending $2.6 million in recovery costs, including incident response and digital forensics, additional staffing, and Microsoft Cloud infrastructure expertise.

Now, certainly, had they said yes to the ransom, it's not like paying $51,000 would have instantly brought all their systems back. So there would have still been substantial cost over and above paying the ransom. Hard to say how much. But the point was they said no to $51,000 ransom, and it cost them $2.6 million to fully recover.

Baltimore, Maryland, another victim, was hit in May. The attack halted city services like water bills and permits and more. The ransom demanded was $76,000. Baltimore also said no, and ended up spending a rather staggering $18.2 million in restoration costs and lost revenue.

Leo: Whoa.

Steve: So, I mean, I guess, you know, putting it into context, it's clear that, as I said, even if you pay the ransom and you get the ability to decrypt your computers, it's still going to - there's a lot of remediation that goes into performing the decryption, getting rid of the gunk, and then how do you know your systems are clear? I mean, you know, you're still going to be hurt. But ouch for, you know, 18.2 million in restoration costs. So many people have said that obviously, I mean, it's easy to say don't click that link in email. And it really does look like these cities are going to be needing to be in a position where, if something like this hits them, they are able to recover themselves in a more timely fashion.

Also I did want to note real quickly that Microsoft's project to bring their Chromium-based Edge browser has now reached Windows 7. It is only available in the Canary channel, which is the most bleeding-edge channel, not yet dev and beta. But if you go to MicrosoftEdgeInsider.com, you can now download the installer that runs under Windows

7 and bring the Edge version of the Chromium-based browser to a Windows 7 system. I haven't done it yet, but I am sitting in front of a Windows 7 machine, and I will probably do that. Seems like a good thing.

One quick little bit of closing the loop. A listener, Hafthor, wrote: "I think you might be wrong about worming the RDP bug." He said: "The advantage would be to gain access to machines on the LAN to have them mine, as well." And that's, Leo, that's a point you have made a couple times is that using these things to get inside your LAN is an advantage. I would argue that, if you're trying to use RDP and this bug in the LAN, it's no more necessary than it is on the Internet. That is, the reason the worm makes sense for Exim, and thus we see one that is out on the Internet now attacking Exim mail servers, is that it takes so long to make this happen. After you find the server, it takes a week of a persistent connection. So you want somebody else's computer to be doing that, not your own.

So a worm is perfect. Here, it's an instantaneous, get it onto the system, with an authentication bypass for RDP. So in the same way that you can immediately get those Internet-exposed systems, so, too, could you get the internal network-exposed systems just by scanning the LAN that your inside interface is connected to. So for what it's worth, I mean, I guess I take his point. But I still don't think that a worm makes sense.

Leo: Okay, Steve.

Steve: So as we know, we've talked about it in several different contexts, bug bounties have become a permanent feature of today's software and system security ecosystem. We have fun every year talking about the Pwn2Own competitions, a white hacker competition sponsored by large sponsoring companies, some of whom have their products hacked and their defects then responsibly disclosed before they are publicly known, and that way they're able to be repaired. Then we have the somewhat sketchy outfits such as Zerodium, whose tagline, a little bit chillingly, reads: "The leading exploit acquisition platform for premium zero-days and advanced cybersecurity capabilities." And as we know, they pay big bucks for big exploits, which they presumably resell to major player state-level actors, foreign and domestic law enforcement and intelligence services, and we don't really know who. So there are those categories.

But the other one is white hat bug bounty clearinghouse middlemen, and the premier one is HackerOne. They sort of serve as a matchmaker between those who put up bounties for the responsible discovery and reporting of bugs in their own products, and the hackers who enjoy finding them and reporting them and are motivated by those bounties. So in contrast to Zerodium's pitch, HackerOne states: "More Fortune 500 and Forbes Global 1,000 companies trust HackerOne to test and secure the applications they depend on to run their business."

HackerOne reports that, for organizations that found vulnerabilities before they were exploited using HackerOne, Forrester found benefits of up to $1.6 million, and an ROI, a return on investment, of up to 646%, meaning that the economic benefit to these large companies of having bugs found before they're exploited is, like, way pays back the bounties that they are offering. And I've got a list of the Top 20 bounty payers that we'll be going through in a second. But I wanted to mention that HackerOne touts, they say: "From implementing the basics of a vulnerability disclosure process to supercharging your existing security programs via a bug bounty program, HackerOne has you covered." And the other thing they say is more security teams use HackerOne to manage vulnerability disclosure and bug bounty programs than any other platform.

So the beauty of this is it's certainly possible for an organization to set up their own bug bounty program. But I can really see the advantage of certainly HackerOne is a commercial operation. They're going to get a piece of the action. But on the other hand, they sort of form a central clearinghouse that makes it much easier for a company to say, okay, we want to establish a bug bounty program. We're simply going to register ourselves with HackerOne, let them manage it for us. Essentially outsource that whole process. Okay. So in their promotional material they quote GM, General Motors' Vice President of Global Cybersecurity saying: "Hackers have become an essential part of our security ecosystem."

Okay. So I mentioned the Top 20. Interestingly enough, we were just talking about Verizon. At the head of the list of the Top 20 biggest, fastest moving, and most lucrative - oh, oh, oh, I forgot. I have a link in the show notes here to a page that is titled "Hactivity" at HackerOne. And when I brought the page up - and in fact I'll just refresh it now. Leo, if over in the upper left you click on "New" rather than "Popular," and then "Bug Bounty," for example, about an hour ago there is one of HackerOne's clients, Deliveroo, paid out a $500 bounty. GitHub, three hours ago, a $2,000 bounty. Four hours ago, GitHub, $617. Also four hours ago, $617.

Leo: This is actually scary.

Steve: I know.

Leo: These are all serious, I presume, bugs.

Steve: Yes, yes. In fact, if you scroll down a little ways, there's PayPal, eight hours ago, paid $3,200 for a bounty that was found.

Leo: Wow.

Steve: Spotify.

Leo: There's a lot of bugs out there.

Steve: Uh-huh. But many, many, many fewer because we have good guys finding them and reporting them.

Leo: Right, right.

Steve: Here's GitLab, $1,000, nine hours ago. Upserve, $2,500, nine hours ago. Mail.ru, believe it or not, $750, 10 hours ago. Upserve a bunch, several thousand dollar bounties.

Leo: Holy camoly. New Relic. Automattic. Uber. Casper. I mean, this is unbelievable.

Steve: Ooh, look at this one. $11,600 paid out by PayPal yesterday.

Leo: So that means it was a pretty severe flaw.

Steve: Yes. GitLab, three grand paid out.

Leo: Holy cow.

Steve: Yup.

Leo: Wow. This is amazing. And this is all through HackerOne; right? So this is just one of the people who provide these.

Steve: Correct. And, in fact, here's HackerOne paid $2,500 for a problem found in their own system.

Leo: On HackerOne.

Steve: Four days ago. Yup.

Leo: Oh, man. People are making some, I mean, this is a living.

Steve: This is - oh, there's Uber, $6,500, four days ago. $2,600 paid by New Relic. It is a living.

Leo: Should I be encouraged? Here's one for Uber for $14,500 last week.

Steve: Oh, baby.

Leo: Should I be encouraged to see these? Like, oh, these are bugs that are fixed? Or discouraged at the vast number of them?

Steve: I know. It is a mixed blessing.

Leo: Here's a $10,000 bounty by Valve two weeks ago. Wow. This is amazing.

Steve: Yeah.

Leo: That's kind of an eye-opener.

Steve: Isn't it sobering?

Leo: But you've got to praise HackerOne for publishing this. I guarantee you Zerodium is not publishing this information; right?

Steve: No. They are a black hole.

Leo: Yeah. Wow.

Steve: So get this. This gives us some - now I'm glad we took a look at that, and I didn't forget to cover this. Verizon is number one of the Top 20. This was a list that was pulled together by Trend. No, no. ZDNet, sorry. ZDNet pulled this together. Verizon has been a member of HackerOne for nearly five and a half years. They joined in February of 2014. Are you sitting down, Leo?

Leo: Yeah.

Steve: During this time, Verizon has paid out more than $4 million in bounties.

Leo: What? Whoa.

Steve: The top award was $16,000. I'm sorry, $6,000. And a total of 5,269 bug reports resolved across Verizon's products. The number two position on the list launched two years later, in March of 2016, is Uber. Since their launch, Uber has shelled out $1.8 million in bounties, paying a top bounty of $15,000 and resolving 1,172 bugs.

Behind Uber in third place is PayPal. Since just September of 2018, they've paid out more than $1.17 million, $1,170,000, for a total of 430 reports. Top bounty, they paid out $30,000 for some bug or bugs, which is twice Uber's maximum and five times Verizon's maximum. So PayPal, they're parting with some serious coin. But they clearly take security seriously. They know they have to have it. And they're willing to have good guys find and report the problems, pay them some money, and not suffer the reputation damage of having a major breach. What's more expensive to them? And so this is why it makes sense.

Shopify since April 2015 has paid out more than $1.1 million across 996 reports with a top payout of $25,000. Twitter is in fifth place, since May of 2014 paying out the same as Shopify, 1.1 million, across almost the same number of reports, 995 reports, paying a top bounty of $15,120. Intel, in number six place, $800,000 since February of 2018. Airbnb in seventh place, paying out more than $600,000 since joining in February of 2015. Oh, and Ubiquiti, who I was just talking about. I'm glad to see that they are there. They're participating ever since March of 2015, paying out $600,000 to have bugs privately found and responsibly disclosed, allowing them to fix them rather than bad guys getting them.

And here's Valve. You mentioned that you saw one they had just paid, Leo. They're ninth in line, having joined in May of 2018, so almost about a year ago. And their total payout is lower, $20,000 across 470. That doesn't sound right. $20,000? No, it must be $200,000. I slipped a zero when I was transcribing this yesterday. Or maybe more because GitLab is in the middle - oh, that must be a single payout of $20,000, yes, from Valve. GitLab is right at the middle point, having paid out $570,000 since February of

2018. And immediately adjacent, one spot below, is GitHub in the 11th location, having joined in April 2016, having paid out a total of half a million, of $520,000 in 348 reports. And I won't keep going. But there's Slack is there.

Starbucks is in 15th place, having paid out more than $300,000 since November of 2016. Mail.ru, Grab, Coinbase, Snapchat, HackerOne themselves. Dropbox paid a hefty maximum bounty of $23,000. And finally, in the last place, now number 20, is VK, which bills itself as Europe's largest social network. So anyway, I wanted to share these numbers, as we saw, to sort of drive home the reality of the fact that there is, you know, we've talked about how possible it is for a hacker to make a career, I mean, this is the world we're in today. Everybody is online. Apps have bugs. Companies are willing to pay some serious money relative to typical annual salaries to have bugs privately reported. So maybe spend some time in the evenings, people who are interested in bug hunting. Supplement your income, and maybe ultimately make a career out of it. It's certainly possible.

Leo: What does this say about the quality of software?

Steve: I know, Leo. It says this stuff is being shipped on a schedule, not on "it's ready." And if you talk to developers, they're being, you know, they estimate how long it's going to take them. It's why I never, ever say when I'm going to have something done. I learned a long time ago, I don't know. And fortunately, I don't have a boss who makes me ship something that's not ready. So yes, it takes a long time for my stuff to go out the door. But once it does, it's done. I know that SpinRite is old. SpinRite 6, I finished it in 2014. I haven't touched a byte of code since - wait, no. Not 2014. 2004.

Leo: Oh, man.

Steve: So it's 15 years old.

Leo: And have you had any bugs discovered?

Steve: There are no bugs. It's old, yes, and I need to update it. But there are no bugs in SpinRite because when I'm done, I'm done. But that's old school. That's not the way it is these days. So, yeah, we've got - it is really, really sobering. So again, to our listeners, HackerOne, H-A-C-K-E-R-O-N-E dot com, then click on the Hactivity link at the top and browse around and see if you wouldn't like to earn some of that because you probably could.

Leo: Well, that's a, yeah, where do you learn those skills?

Steve: In fact, well, in fact they have it. If you hover over For Hackers, then there is Start Hacking, Hacker 101, Leaderboard, and over on the right...

Leo: They'll teach you.

Steve: ...Hacker 101: Learn How to Hack and a Get Started button.

Leo: I'll be diggety-danged.

Steve: Yup.

Leo: Well, that's kind of cool. You know, you looking for work? Learn how to hack. You could be working at home.

Steve: Yup.

Leo: Oh, this is really interesting. This is really interesting. Wow. I wonder if it's really that easy to learn how to do this. Wow. That's really - that's amazing.

Steve: Yeah.

Leo: Steve Gibson. There are no bugs in this man's tree. If you want to listen to the show, we do it every Tuesday, a little late today, but normally 1:30 Pacific, 4:30 Eastern, 20:30 UTC. Best place to do it is at TWiT.tv/live. Live video and audio stream there. And join us in the chatroom at irc.twit.tv if you're listening live. Now, if you can't make it live, and I know most of you can't, don't worry because this is a podcast. That means there's on-demand versions of the show. Steve has audio available at GRC.com. And transcripts, Elaine Farris's carefully crafted transcripts of every episode. And it's free at GRC.com.

In fact, there's only one thing that's not free right now at GRC.com. That's SpinRite. That's Steve's bread and butter, so if you need - if you have a hard drive you need a copy. And if you need a copy, go to GRC.com, the world's best hard drive recovery and maintenance utility. Everything else there is free, and it's fun to browse around. It's just a little nerdy treasure trove of fun and interesting stuff. SQRL is there. You can learn more about that. You can get passwords. You can, I mean, it's just great. It's just the greatest. GRC.com.

Steve's on Twitter at @SGgrc. You can leave him a direct message there if you have a question or a comment. You can include that in our feedback section, or GRC.com/feedback. And we have audio and video of the show ourselves at TWiT.tv/sn. That's the website, TWiT.tv/sn. And of course you can subscribe. This is one of the oldest podcasts in the lexicon. And so every podcast application should have a copy available for you. If you subscribe, you'll get it the minute it's available every Tuesday afternoon.

Okay, Steve. I saw "I Am Mother." Enjoyed that.

Steve: Oh, yay.

Leo: Enjoyed that quite a bit. Yeah, good show.

Steve: Yeah.

Leo: And you were right. The trailer. Don't watch it. And the worst thing, I'm sitting there, and I'm hovering over it on Netflix because I'm just trying to decide, and it starts playing. And I forgot that you said don't watch the trailer. And they played - Lisa will tell you. I'm going, "Oh, no, no, I don't want to see it." And she said, "What?" And I said, "Oh, Steve said not to watch the trailer. Now I know what's going to happen." But you don't fully know what's going to happen, I think.

Steve: No, no. But still.

Leo: In fact, I have to talk to you off the air because I don't understand the ending. I want to - I'm trying to figure out, well, you know what I'm trying to figure out. Did you ever...

Steve: I do. And in fact, I will send you a link to the reviewer who joined IMDB just to post...

Leo: Just to explain it.

Steve: Just to explain it because he nailed it.

Leo: I think I know. But I'd like confirmation. That's the modern thing, by the way. I can't watch a show anymore without going to the web and seeing what people say about it. Like, okay, well, why did - who did - why did they have - what did that - and then she - because I'm not as clued in as some of these people. They watch pretty carefully.

All right. Have a great evening, Steve. We'll see you next week.

Steve: Okay, my friend. Bye.