Microsoft's Reasoned Neglect

Description: This week we briefly look at Firefox's plan to block unsecured downloads. We examine the threat posed by T-Mobile's massive and deep data breach and what current and past customers of T-Mobile should do. We look at three additional so-called "Overlay Networks" in addition to Tailscale, and also at the consequences of another Orange Tsai Microsoft Exchange Server exploit chain discovery. We'll also examine a simple-to-make flaw in the Razer gaming mouse installer, cover another worrisome IoT protocol screw-up, and share a couple of feedback notes and a question from our listeners. Then I want to conclude by following up on last week's discussion of Microsoft's apparent culpable negligence with a proposed explanation of their behavior and motivation which fits the facts so well that it becomes reasoned neglect.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-833.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-833-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here with more on the T-Mobile breach and some very practical advice about what we T-Mobile customers should be doing. We'll also talk about two more overlay network solutions that Steve likes a lot. And then the amazing story of the Razer mouse hack. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 833, recorded Tuesday, August 24th, 2021: Microsoft's Reasoned Neglect.

It's time for Security Now!, the show where we cover your security, your privacy, everything you need to know about how the Internet works with this guy right here, the Explainer in Chief, Mr. Steven Gibson. Hello, Steve.

Steve Gibson: Hey. I like that "set your phasers to stun" line, Leo. You're going to have to hold on to that one.

Leo: We decided everybody should turn their cell phones off and set their phasers to stun for the following program.

Steve: That's right.

Leo: Yes.

Steve: So we're at 833.

Leo: Wow.

Steve: For this second-to-the-last - this is the penultimate episode of August, Leo.

Leo: That's amazing.

Steve: I've got to get that word in...

Leo: Every time. Every time.

Steve: That's right.

Leo: It's always a penultimate something.

Steve: I guess that is true. So this week I'm going to attempt to explain the inexplicable. Last week was Microsoft's culpable negligence, which was - no one could explain it. I asked everybody. No explanation.

Leo: You know, in a way, after you did that, I thought there would be lots of stories repeating that because that's stunning, eight months knowing about a flaw and not fixing it until it was a zero-day for eight months. That should be headline news everywhere, and nobody picked it up except you.

Steve: No, I think we've all just been beaten into submission at this point.

Leo: That's right. It's like we don't expect better. It's like, yeah, well, what do you expect?

Steve: Can I still type on my keyboard? Fine. Okay. So I did, however, get lots of feedback from our listeners. This week's podcast is titled "Microsoft's Reasoned Neglect." And I think I've made a really good case for what is actually going on. It'll be fun to discuss this with you at the end. But first, we're going to briefly look at Firefox's plan to block unsecured downloads. We examine the threat posed by T-Mobile's massive and deep data breach and what current and past customers of T-Mobile should do as a result.

We look at three additional so-called "overlay networks," which add to Tailscale. There are some other alternatives that have come to light since I mentioned Tailscale that I wanted to share with our listeners. And also we look at the consequences of yet another Orange Tsai Microsoft Exchange Server exploit chain discovery. That would be the ProxyShell exploit chain. We'll also examine a simple-to-make flaw in Razer gaming mouse installer, which generated...

Leo: That was the funniest story ever.

Steve: It is. And it generated an outsized response. But, okay, so on the one hand it's like, really? Like everyone was calling it a zero-day and jumping around. But what's really interesting is that once you see this - and Will Dormann from CERT commented exactly along these lines, and I'll share that - you then have to ask, how many other of these have happened and no one noticed, and how many have yet to be found? So we'll talk about that. We also have another worrisome IoT protocol screw-up. I want to share a couple of feedback notes and a question from our listeners. Then I'm going to conclude by following up, as I said, on last week's discussion of Microsoft's apparent culpable negligence with a proposed explanation of their behavior and motivation, which fits the facts so well that it offers an explanation that I would call "reasoned neglect."

Leo: Hmm. Hmm.

Steve: And of course we've got a great Picture of the Week. So, yeah.

Leo: That's quite interesting. I look forward to hearing this thought experiment, I guess we would call this. Yeah? Because you don't have any inside information from Microsoft.

Steve: Well, it's funny because I was sure that I mentioned something years ago. And I put the gang in the Security Now! newsgroup on the hunt, and one of those people found the podcast where I said what it was that I was sure I remembered. And it turns out I was quoting Bruce Schneier, of all people.

Leo: Oh, interesting.

Steve: So that's going to be good.

Leo: Okay. Good. This is a continuing saga. The Picture of the Week saga continues.

Steve: We've been in a, for better or for worse, a wires phase for a while now.

Leo: Yeah.

Steve: This one was sent by a listener who saw the other wiring cabinets, and I guess he remembered this. This is very cool. This is a photo of what was described as the old Telefontornet telephone tower in Stockholm, Sweden, with approximately 5,500 telephone lines circa 1890. If you follow that link below the picture, Leo, there are actually some other very cool photos. But this is, if you scroll down a bit, you'll see like they have one of the tower before it was wired up, which is...

Leo: Oh, my god. It was meant to be wired like that. Wow.

Steve: Oh, yeah, yeah. This thing was designed so it's a 3D large tower, and you can see if you look really closely, little tiny glass bobbins, you know, insulators, where the wires are all going to be terminated in this massive array.

Leo: Holy cow.

Steve: So the description with this thing says: "In the late 19th Century, the miracle device called the telephone had been invented, but the simple concept of undergrounding telephone cables had eluded engineers. Due to technical limitations of the earliest phone lines, every telephone required its own physical line strung between" - they didn't have party lines back then.

Leo: It was point-to-point. It was all point-to-point.

Steve: Point-to-point, exactly.

Leo: Wow.

Steve: "...strung between a house or business to a phone exchange where the call was manually connected by a live operator."

Leo: I am connecting your call now with a patch cable. This is before they figured out switches, I guess. This is incredible.

Steve: Oh, yeah, yeah, 1890, Leo.

Leo: Yeah.

Steve: You know? So, yeah, they had literally patch cords. So this thing said: "The somewhat quixotic result of so many individual lines was the construction of elaborate and unsightly towers that carried hundreds to thousands of phone lines through the air.

"In Stockholm, Sweden, the central telephone exchange was the Telefontornet, a giant tower designed around 1890 that connected some 5,000 lines which sprawled in every direction across the city. Just by looking at historical photos it's easy to recognize the absurdity and danger of the whole endeavor, especially during the winter months. Everything that could possibly go wrong did, from high winds to ice storms and fires. The network was extremely vulnerable to the elements. Luckily, phone networks evolved so rapidly that by 1913 the Telefontornet was completely decommissioned" - oh, so maybe the picture of the tower was after it had been decommissioned. Anyway, it was completely decommissioned "in favor of much simpler technology. The remaining shell" - ah, that's it. "The remaining shell stood as a landmark until it, too, caught fire in 1953 and was torn down."

Leo: Was it wood? Oh, my god.

Steve: Yeah, yeah.

Leo: Holy moly.

Steve: Wow. Anyway, the picture is wonderful. I mean, the cabling is so dense it almost looks like mist. I mean, it's like...

Leo: I thought it was the wind at first. It was the wind whipping through it.

Steve: Exactly. Those are individual phone lines going out to subscribers who, by the way, paid a hefty fee. This thing says it originally had only 121 subscribers. Of course it grew to 5,500. They said: "The telephone company charged subscribers between 160 and 280 Swedish krona, depending on the subscriber's location and distance from the exchange. This was equivalent to paying a subscription fee of what would today be 9,000 to 16,000 krona," which in U.S. dollars is 1,100 to $2,000 in today's value, which is a month, a very high rate, obviously. But still, if you had some business purpose for having a phone, or it was just cool, then yeah, you'd pay the price. Anyway, just another cool photo. And are we done with those? I think maybe we're done. We'll see what happens.

Leo: Wow.

Steve: Okay. Speaking of downloading, which we weren't, but we will, Firefox will soon be blocking mixed-content downloads by default. Back in 2020, as we discussed at the time, Chrome and its brethren Chromium browsers gradually made the switch to blocking mixed-content downloads, sort of in that halting sort of let's not make any mistakes here, we're going to stick our toe in and see what happens way that Chrome has, between releases 81 through 88, when by the time we got to 88 it was like, okay, fine, if you're on a secure page, we're not going to let you download anything over an unsecured link.

So that feature, which is actually already present in Mozilla today, but not yet enabled by default, will soon be flipped on for everyone. Once that's done, all HTTP files downloaded from an HTTPS page will be blocked with a message in the Firefox Download Center. An option will be available to allow you to push past it, after saying yes, yes, yes, I know, I realize there's a danger, blah blah blah, if users choose to. Which I think is the right tradeoff, just to keep you from doing it by mistake, if some page is trying to trick you for some reason. On the other hand, it's an HTTPS page, so it's got to really be at this point, I would argue, negligence on the part of people running HTTPS sites who are sort of lazily still serving HTTP downloads without having switched over. Probably no reason not to switch at this point.

Leo: I wonder, I'm thinking maybe we might fall into that because of course our sites are HTTPS, only because Google would penalize you if you weren't, but maybe the podcasts - I guess, are they HTTPS? I don't even know.

Steve: I bet they must be. I'm never getting any warnings when I'm downloading content.

Leo: Yeah, they must be. Apple warns you every single time you download something from any site at all.

Steve: Wait a minute, you're downloading something.

Leo: It literally says that. Are you sure you want to download something from this site? Yes.

Steve: You know, you're an iOS user. And, you know, really, what do you need to download something for?

Leo: There is a balance between too many warnings, you know, you have too many warnings, people just start ignoring them.

Steve: Yeah, like have you seen one of those cookie warnings lately, Leo?

Leo: Yeah, my god.

Steve: It's like, come on.

Leo: Oh, don't get me started.

Steve: One thing nice about Firefox, because they still are the browser for the rest of us, directly accessed HTTP download links which are copy-and-pasted into Firefox's address bar, will not be blocked. That is, obviously they really can't be because you're not coming from a secure page. The idea is just to prevent somebody from tripping over themselves by mistake, by having some download that nobody's looked at for a while. So if nothing else, if a site has HTTP download links, Firefox is going to say, hey, you know, maybe you should ask the webmaster, if there is such a thing anymore, why these things are HTTP and not HTTPS.

Anyway, the cool thing is you can turn this on right now in Firefox before 92. This will be default in 92, which is like sometime the start of September. Next podcast is the 31st of August. So after next Tuesday, that's the start of September, we're going to get Firefox 92. But if you go to about:config and bring up that infinite list of things you can play with, and you put the word "insecure" into the search box, the first thing still of many that comes up is dom - document object model - "dom.block_download_insecure." You flip that to true. It'll be false right now by default. Turn it on, and then Firefox will start behaving the way it's going to in a week. So make the jump. Not that it really makes that big a difference anyway, one way or the other.

Leo: I think that mixed-content warning confuses the hell out of normal people.

Steve: Yup.

Leo: They don't know what that means, or what to do about it even.

Steve: No. Yes, Leo, this was really not designed for all of this, you know.

Leo: None of this was designed for normal people.

Steve: No. My poor mom. HTPTP what?

Leo: Oh, god.

Steve: Colon slash slash?

Leo: It's as bad as you trying to pronounce Swedish. It's just not going to work.

Steve: She said, "Do I still have to use AWOL?"

Leo: Oh, my gosh.

Steve: I'm like, yes, Mom. Yes, Mom.

Leo: But I feel really bad for people who don't have a technical bent. This stuff is crazy.

Steve: It is.

Leo: That's why we listen. That's why we listen to the show, yeah.

Steve: Not getting less crazy. Okay. So the news from T-Mobile is all bad. Last week T-Mobile confirmed their latest data breach, making it the fifth data breach in four years. They're going to have to start explaining to someone what their problem is. There were two previous attacks in 2020, last year; one in 2019; and the first in 2018. But this most recent breach is the largest by far. And the numbers of affected customers keep growing, like, as they keep digging into this like, oh, what happened? First it was like, wait, we got breached? Who says? They didn't know. Literally. Until someone started selling the content on the dark web. The most recent update reveals that the cyberattack exposed over 54 million individuals' data. Last weekend, a threat actor began selling the personal information, claiming it was 100 million T-Mobile customers on a hacking forum for - he was asking 6 BTC, which is about 280K right now because, as you commented recently, Leo, bitcoin is coming back - I think it was on Sunday.

Leo: Almost 50.

Steve: It's coming back. It's creeping back up again. And I'm moaning about the 50 that I just said, eh, who needs these?

Leo: Don't think about it, Steve. Don't think about it. It's not good. So this is the text I got from T-Mobile. "Unauthorized access to some of your personal data. We have no evidence." Of course they're morons. But "We have no evidence that your debit/credit card information was compromised. But we're going to give you just in case three years of credit monitoring, free." Thanks.

Steve: Yeah, yeah. So we'll talk about what users should do in a second. Hopefully all of our listeners have. So this hacker claimed that the stolen database contains the data for approximately 100 million T-Mobile customers. Now, here's the bad part. This is not one of those, well, yeah, they got the hashed password.

Leo: No.

Steve: The exposed data can include customers' IMSI, IMEI...

Leo: What's IMSI? I know IMEI.

Steve: That's one of the other things, you know, they're big crypto, you know, strings of numbers. But you don't want anybody to have them because they could get up to mischief, like clone your SIMs and things. Phone numbers, customer names, security PINs, Social Security numbers, driver's license numbers, and date of birth. And of course names; right? So in other words, the keys to the identity theft kingdom.

Leo: Yeah.

Steve: The database was said to have been stolen approximately three weeks ago, apparently when T-Mobile was on vacation, and contains customer data dating back as far as 2004. In an interview with the hacker, which Lawrence Abrams of BleepingComputer, he's BleepingComputer's founder, Lawrence Abrams had this interview, reported that the hacker said their entire IMEI history database going back to 2004 was stolen. So that's all of the, basically, the serial numbers of all the T-Mobile phones that they've had accounts for since '04. Okay. And that was, what, a year before the podcast. So 18 years of data that's been stolen.

After the data first went up for sale, T-Mobile later confirmed, oh, some of our servers have been hacked, what do you know, and began investigating what customer data had been exposed. Last Tuesday, on August 17th, T-Mobile first said that the personal information of 48.6 million individuals was exposed during the attack. They later updated that to include an additional six million customers - oh, six million more - or prospective customers who were also affected by the attack. You don't even have to have, like, signed on the dotted line. No, you just open a conversation with T-Mobile, and you're history. So T-Mobile also confirmed that the attackers stole their customers' IMSI and IMEI numbers. That was confirmed.

Okay. So here's a breakdown. 13.1 million current T-Mobile postpaid customer accounts, as opposed to, distinct from prepaid. So 13.1 million current T-Mobile postpaid customer

accounts that included first and last names, date of birth, Social Security number, and driver's license/ID information. Bad. Bad, T-Mobile, bad. 40 million former or prospective T-Mobile customers, including first and last names, date of birth, Social Security number, and driver's license/ID information. Okay, so a total of 53.1 with all of that. Basically game over. 667,000 accounts of former T-Mobile customers exposing customer names, phone numbers, addresses, and dates of birth were compromised. 850,000 active T-Mobile prepaid customer names, phone numbers and account PINs were exposed. So even if you're prepaid, you're still hosed. And finally, 52,000 names related to current Metro by T-Mobile accounts may have been included. So, yeah, count yourself in.

Okay. So identity theft is one of those things that can really screw up one's life. You need to prove that it wasn't you who applied for and received credit under your name, when the other person provided, you know, the other person you're proving, some other mysterious we-don't-know-who other person provided all of the personal information that only you are presumed to have. So such cretins immediately run up massive charges under your name, using and destroying your credit. There are tons of horror stories about the mess this has caused for people. I mean, it's ruined lives. And what's needed to apply for credit in someone else's name? Exactly the data that has just been exposed for tens of millions of T-Mobile customers because, guess what, that's what you provided to get credit from T-Mobile. They said, oh, yeah, this is what we know about people that convinced us to give them credit. Let's have it all sold on the dark web.

Okay. So this is why the absolutely number one best advice I have, the advice I give to anyone and everyone, is to simply run with permanent locks on your accounts at all three of the credit reporting bureaus: Experian, TransUnion, and Equifax. I've had all three locked for me since I first talked about this years ago during one of these identify theft events. This is easily done for someone who is not routinely applying for credit. You know, Leo, those of us who have thinning hair.

Leo: It's easier for us, yeah.

Steve: It's easy for us.

Leo: Yeah, if you're getting credit cards, you have to unfreeze. If you're getting a car, you have to unfreeze.

Steve: Yeah, because you have to let those creditors check your credit.

Leo: And by the way, you have to give them your Social Security number, your driver's license, all of this stuff that - this is why T-Mobile has it, because they run credit checks before you can get a prepaid phone.

Steve: Exactly.

Leo: Or a postpaid phone, yeah.

Steve: And unfortunately they don't wipe it, obviously. They just hold onto it.

Leo: They just keep it. Why not?

Steve: That's right.

Leo: And we certainly wouldn't want to secure it or anything.

Steve: Cloud storage is so cheap. Maybe that'll be useful. Okay. The good news is it has recently become easier for those who do need to occasionally unfreeze their credit worthiness.

Leo: More importantly, free, thanks to federal intervention.

Steve: Yes.

Leo: Used to be expensive.

Steve: A couple of years ago I realized that I was losing money by not using an Amazon branded credit card for my many purchases through Amazon, since they were providing a couple of points of discount for purchases made through their own card. Why would I not take advantage of that? So I had the occasion to need to lift the locks on my credit. What I discovered was that all three of the agencies now offer a convenient free 10-day unlock with automatic relock. So I told them all to start the 10-day counter. I applied for the Amazon card, qualified through whichever one of the three agencies Amazon queried, and then all of them were automatically relocked.

So it makes it so practical, you know, with accounts locked this way, no would-be creditor is able to query for my credit, and thus no creditor would allow a thief to open a credit account under my name. So I'll just say it again. If you have not done it, if you are not continually needing for some reason to have creditors accessing your credit with those big three firms, then why not lock those agencies down? It's trivial to do, and it buys a lot of peace of mind.

Leo: Doing it right now. And it's free now, as we said, to freeze and unfreeze, yeah.

Steve: Yeah.

Leo: So there's a fraud alert, and there's a credit freeze. The credit freeze is what you want. Fraud alert you have to have fraud first.

Steve: Actually, there's a freeze, and there's a lock. Unfortunately, the jargon can be confusing. The freeze is temporary. The lock is permanent.

Leo: Ah.

Steve: So the lock is what you want.

Leo: Okay.

Steve: So I'm glad you brought it up because you do need to read through that and look at what it is that they're doing. But mine are just permanently locked.

Leo: Lock, yeah.

Steve: And the other cool thing is when I was just looking at it yesterday, I'd forgotten that there are now iOS apps that allow you, after verifying you are who you are, to use the app to unlock your credit on a transient, temporary basis. So it really becomes quite a practical thing to do. Again, everybody in my opinion, I mean, unless you're newlyweds buying all kinds of stuff and cars and homes and things, just lock that stuff down. You want to protect your credit and not have it destroyed by somebody, I mean, look at all this mess that T-Mobile has just created for people. Oh, we're going to give you a free three-year...

Leo: From McAfee of all people. Geez, Louise.

Steve: Oh, great.

Leo: Yeah. Two years, by the way. Not three years.

Steve: Endorsed by John himself.

Leo: Yeah.

Steve: Okay. So ProxyLogon's kissing cousin ProxyShell. Now, we're talked about this guy before, the Taiwanese security researcher known as Orange Tsai, T-S-A-I, whose work we've covered before, unveiled his ProxyShell exploit chain during the Pwn2Own 2021 hacking contest a few months ago, in April of this year. We talked about it at the time. The chain invokes three unique and original exploits, and they are doozies. That's the technical term. There's CVE-2021, then we have 34473, 34523, and 31207.

So get this. The first one provides a mechanism for pre-authentication remote code execution, enabling malicious actors to remotely execute code on an affected system. And we're talking about Microsoft Exchange Server. So pre-authentication remote code execution. Meaning you don't have to have an account. You don't have to prove who you are. This is before authenticating an identity, you can remotely execute code.

Then we have the second one, enables malicious actors to execute arbitrary code post-authentication on Microsoft Exchange servers due to a flaw in the PowerShell service not properly validating access tokens. And the third one enables post-authentication malicious actors to execute arbitrary code in the context of system, you know, root for Windows, and write arbitrary files.

Okay. So we have a pre-auth remote code execution exploit and a pair of post-authentication exploits, one of which enables an attacker to execute arbitrary code as root and to write arbitrary files. Do you imagine that a talented hacker might be able to make something of those? Orange Tsai used this trio to completely take over a Microsoft Exchange Server remotely, and in so doing earned himself $20,000 for a successful server compromise last April. Of course, part of the deal of Pwn2Own is that the details of any vulnerabilities used are immediately turned over to the publishers of the impacted software. So in this case the details of the exploit chain were immediately shared with Microsoft, following which Microsoft promptly patched the three vulnerabilities in May and July this year.

Okay, now, I'll just note that the vulnerabilities were demonstrated publicly. The details were handed over to Microsoft, whereupon Microsoft promptly patched the three vulnerabilities the following month and month after. That's not what happened at the beginning of the year, when vulnerabilities were privately disclosed to Microsoft. But I'm getting ahead of myself. We'll be discussing that at the end of the podcast. Just worth noting that when they are publicly disclosed, Microsoft fixes them.

Okay. Then during his follow-up talk at Black Hat, during the first week of this month, of August, Orange Tsai disclosed additional information about the attack. As one would at a Black Hat conference; right? Microsoft patched this in May and July. So next month, time has passed, disclose a little bit more. That information was enough to allow it to be leveraged. And shortly following his attack, there was publication of additional technical information by other researchers who had managed to recreate it.

After that, it didn't take the bad guys long to start looking for vulnerable Microsoft Exchange servers. And sure enough, as with the earlier ProxyLogon and ProxyOracle disclosures in March and April of this year, not all server administrators jumped on those patches. Consequently, a scan performed two Sundays ago, on August 8th, by SANS Security, which was two days after the publication of ProxyShell's proof-of-concept code, yet weeks after the April and July Patch Tuesdays that were supposed to fix this, that scan discovered, okay, two Sundays ago, more than 30,400 Exchange servers from a total of 100,000 systems that had yet to be patched and remained vulnerable.

So before the releases, 100,000 systems, there are 100,000 Microsoft Exchange servers out on the Internet, they were all originally vulnerable. Two weeks ago, what, a little more than two-thirds of them had been patched, but were still at 30% unpatched. I have a chart in the show notes that shows sort of the typical exponential patch rate that you see, where there's initially a lot of machines being patched. The number of vulnerable machines is falling rapidly. But inevitably it just sort of slows down. And notice that this chart, Leo, is zero-based. And it stopped dropping. I mean, after a couple weeks, all the machines that were going to be patched have been patched.

And in fact there's a little bit of a dip, it looks like a few new ones came online and haven't been patched or something. Or maybe the scan just had not found them that particular week. But this is the lesson that we see is that, you know, there are still machines out there with Code Red and Nimda scanning the Internet that will be doing it until someone finally trips over the plug and unplugs those things. So it looks like it's leveling off around, what, 15,000 machines that are just still vulnerable and are going to stay that way.

So the initial pre-exploitation started with scans for vulnerable systems. Security researchers with honeypots and fake Exchange servers set up, they were monitoring this. Pre-exploitation started with scans for vulnerable systems, which then quickly turned into actual attacks over this past weekend, according to honeypot logs collected by security researchers Rich Warren and Kevin Beaumont. We've talked about that pair of researchers before. This is a thing they do.

After that, attacks intensified last week, and a new ransomware operation known as LockFile began using this ProxyShell exploit as a way to enter corporate networks. Kevin Beaumont, tweeting from his @GossiTheDog account, wrote last Friday: "ProxyShell is now being used to drop corporate ransomware, as is PetitPotam, same IP and actor as in this thread. Myself and @buffaloverflow..."

Leo: Telefontornet. Oh, I'm sorry, no, that's last week, never mind.

Steve: Yeah, exactly, "..have been watching them." Anyway. And then on Friday, security firm Huntress Labs said it scanned Microsoft Exchange servers that have been hacked using ProxyShell and - get this - found more than 140 different web shells on more than 1,900 Exchange servers. So game over for those. And that's about where that line leveled off, right up around somewhere like 1,900, 1,500 servers. And remember that a web shell is the most trivial to plant and difficult to spot bit of code.

On any server using server-side script interpretation - and Microsoft Active Server Pages (.asp) is ubiquitous on Windows servers, and it's effectively impossible to get rid of. Every time I update, this thing rears its ugly head again, and I have to shut all these things down manually. Anyway, so the point is just a short and simple page of text can be written to implement a listening web shell. You get that file. Remember that there was a file write vulnerability? Uh-huh. That's all you have to do. Write one of these little pages, tuck it into the server somewhere among all the other .aspx things, and you've got yourself a web shell.

And this fact, I mean, just that it's that easy to do this, really demonstrates the security short-sightedness that's inherent in server-side scripting. It's another classic example where ease of use has vastly outstripped the teachings of security principles.

Kyle Hanslovan, the CEO and co-founder of Huntress Labs, said: "Impacted organizations thus far include building manufacturers, seafood processors, industrial machinery, auto repair shops, a small residential airport, and more." So anyway, the long tail on that "servers remaining vulnerable" chart makes clear that it's going to be a long time, if ever, until these previously installed instances of Exchange Server stop being infested with malicious malware.

So once again, a gifted researcher, the same guy who way back in October of 2020 told Microsoft about those previous ProxyLogon vulnerabilities that Microsoft then sat on for months, that guy, he used three different ones to win himself $200,000 from Pwn2Own, waited for Microsoft to patch them, gave them a few more weeks for the patches to take hold, then did his Black Hat briefing to explain a little bit more about them. Still holding back some information, but that was enough for talented researchers to essentially reverse-engineer from the data he had given, create ProxyShell, proof of concept went public. Immediately it was picked up, and Exchange Server is being attacked. In this case, you can't blame Microsoft. They published the patches, pushed out the patches immediately. It was just that there was the typical, far less than 100% uptake rate. And that's enough these days for there to be problems.

Probably the most noise was made during the past week, this weekend, over a clever hack that someone discovered involving the installation of drivers for the Razer mouse, R-A-Z-E-R. Though describing it as a zero-day vulnerability, as much of the tech press has, seems a bit much to me.

Okay. So for those who don't know, Razer is a popular manufacturer of gaming-oriented peripherals, mice and keyboards. So as Windows will, when a device is encountered for which it lacks currently installed drivers, it fetches them on the fly over the Internet from

its vast repository of driver installation packages which have been pre-written and submitted by the vendors of those third-party products. We all know this as "plug-and-play," or somewhat less charitably as "plug and pray."

In any event, when a Razer device is first introduced to Windows, the OS automatically obtains and begins to install the Razer Synapse software onto the local machine. Razer's software allows their customers to configure and customize their hardware, setting up macros, mapping buttons, and so on. And Razer's products are quite popular, being used by more than 100 million users worldwide.

So as it happened, a security researcher using the handle "jonhat" (J-O-N-H-A-T) discovered an exploitable escalation of privilege mistake that had been made in the driver installation flow which allows users whose local privilege may have been tightly locked down, to escape that lock and obtain full local privileges of the root system account, and to do so with somewhat surprising ease. Since device installation needs to make deep changes to a system, Windows runs these installers with system privileges so that they'll be able to make the changes they need.

And by the way, this is another horrible kludge in Windows. I learned during the development of SQRL's self-installation system that, believe it or not, Windows employs the messy heuristic of sniffing programs being run, I kid you not, looking for strings such as "setup" or "installer," and treats such programs differently. I couldn't believe it when I ran into that. But yes, it's true. Anyway, the program RazerInstaller.exe will be downloaded and executed by a Windows operating system process running with system privileges. This allows the RazerInstaller.exe child process to inherit the same privileges as its parent. That's necessary to enable that installer to do what it might need to do.

Okay. Everything's fine so far. But the mistake the Razer developers made was in then causing the setup wizard to allow the user to specify the folder into which the software should be installed without restricting its rights. Okay. In other words, if the user elects to alter the default installation destination, Windows will present the Choose a Folder dialog. But in this instance this dialog, because it's been launched from the Razer installer, which has system rights, will also have inherited the installer's full system privileges.

So if a clever user holds down SHIFT and right-clicks on that dialog, a context menu will pop up and, surprise, among its various options is "Open PowerShell Window Here." Whoopsie. If that option is selected, sure enough, a PowerShell window is opened, having the same rights as the dialog that spawned it, namely full system privileges.

So, in short, the mistake of not reducing the rights of a dialog spawned by a privileged installer, which any locked-down user having no rights can cause to be launched just by connecting any Razer mouse to a system, grants that user full and unrestricted root level access on any Windows machine to which they have physical access.

Leo: It seems like this is going to be much more than just that Razer mouse; right?

Steve: Exactly. Having discovered this important oversight, "jonhat" attempted to do the right thing with his discovery. And had he succeeded, this would have made much less high waves. He reached out and tried to responsibly contact and notify Razer. They blew him off. So after not receiving any response from Razer, he posted his discovery to Twitter last Saturday, tweeting: "Need local admin and have physical access? Plug in a Razer mouse or the dongle. Windows Update will download and execute RazerInstaller as system. Abuse elevated Explorer to open PowerShell with SHIFT+RIGHT CLICK. Tried contacting @Razer, but no answers. So here's a freebie."

Now, if you're thinking, Leo, as you are, gee, that seems like an easy mistake for someone to make, I wonder whether any other installers do that, too, you would be in good company. The following day, last Sunday, CERT/CC's Vulnerability Analyst Will Dormann observed and tweeted: "Many vulnerabilities fall into the class of 'How has nobody realized this before?'"

Leo: Yeah, yeah.

Steve: He said: "If you combine the facts of 'connecting USB automatically loads software' and 'software installation happens with privileges,'" he said, "I'll wager that there are other exploitable packages out there." So, yeah. You didn't have to do any, as you said, deep hacking. Just it's a clever hack, and it's probably a lot more widespread than we suspect. And again, as I said, this is just - this is what happens when a system like Windows becomes so complicated that nobody, there's no single individual who understands the whole thing. You just can't. There's too much of it. And it's just grown with all these barnacles growing on it over time that end up giving you ways around things that were never intended.

I actually had to, in the case of my SQRL self-installer, I had to, I think I had to, I think in the properties in SQRL's context menu I had something like, you know, SQRL self-install or something. Or maybe I was - I think I was renaming the EXE as SQRL Installer and then relaunching it under that name, and suddenly Windows got involved and popped up the UAC dialog because something was trying to run that had "installer" in its name. And I said, what? You're kidding me. The name of the EXE matters? Yup. Sure enough. Windows just goes, oh, look.

Leo: We've got to check something.

Steve: It's got "installer" in the name. Let's...

Leo: We've got to look at something.

Steve: Oh, my god, you're kidding me.

Leo: Yeah. Seems pretty basic. But, you know, hey.

Steve: Unbelievable. Okay. So in IoT news, a critical ThroughTek - ThroughTek is a company no one's ever heard of because they sell an SDK and technology to OEMs who then repackage it in their own products. Anyway, this SDK flaw enables IoT spying pervasively. It turns out that out there somewhere on the order of 83 million actively in-use IoT devices which are streaming live audio and video, are doing so over a little known and not very secure peer-to-peer network that goes by the name Kalay, K-A-L-A-Y.

Last Tuesday, so a week ago, the guys at FireEye from their Mandiant security team posted the news of their discovery under the title "Mandiant Discloses Critical Vulnerability Affecting Millions" - yes, 83 millions - "of IoT Devices." So in their posting they explain: "Today, Mandiant disclosed a critical risk vulnerability in coordination with the Cybersecurity and Infrastructure Security Agency" - that mouthful known as CISA -

"that affects millions of IoT devices that use the ThroughTek Kalay network. This vulnerability, discovered by researchers on Mandiant's Red Team in late 2020, would enable adversaries" - and in fact does - "to remotely compromise victim IoT devices, resulting in the ability to listen to live audio, watch live real-time video data, and compromise device credentials for further attacks based on exposed device functionality. These further attacks could include actions that would allow an adversary to remotely control affected devices."

And again, we don't know it, but things that have video on them, you know, webcams, baby monitors and such, they all OEMed this technology to solve their audio and video streaming problem from these guys, and as a consequence there are 83 million devices out there, not secure, all using this technology.

"At the time of this writing," they said, "of writing this blog post, ThroughTek [T-H-R-O-U-G-H-T-E-K] advertises having more than 83 million active devices and over 1.1 billion monthly connections on their platform. ThroughTek's clients include IoT camera manufacturers, smart baby monitors, and Digital Video Recorder products. Unlike the vulnerability published by researchers from Nozomi Networks in May of 2021, also in coordination with CISA, this latest vulnerability allows attackers to communicate with devices remotely. As a result, further attacks could include actions that would allow an adversary to remotely control affected devices and could potentially lead to remote code execution."

Okay. I know I sound like a broken record about this. But let me just say this again: You do not want any of your IoT devices sharing a network with your PCs.

Resuming their posting, they said: "The Kalay protocol is implemented as a Software Development Kit which is built into client software" - thus we don't see it - "for example, mobile or desktop applications and networked IoT devices such as smart cameras. Due to how the Kalay protocol is integrated by original equipment manufacturers (OEMs) and resellers before devices reach consumers, Mandiant is unable to determine a complete list of products and companies affected by the discovered vulnerability." Again, it's a library that's just built into things. The vulnerability that they have uncovered has been assigned a CVSS base score of 9.6. In other words, it's a baddie.

Leo: Wow.

Steve: Yeah. And it is tracked as CVE-2021-28372. And then FireEye has their own designation, 2021-0020. They said: "This blog post discusses the Kalay network and 28372 at a high level. It also includes recommendations from ThroughTek and Mandiant, along with mitigation options." And again, here we are, great, it's neat that they came up with this and that they worked with Kalay this year to fix the problems in the SDK and their network, but there's already 83 million devices that will never be updated. Your baby monitor, your webcam, never be updated. They can't have their protocol changed. It's in their firmware.

So, okay. "Mandiant," they said, "would like to thank both CISA and ThroughTek for their coordination and support in releasing this advisory." I've got a link for the whole advisory for anyone who's interested. I won't go into that. But what they did is kind of cool. Mandiant researchers analyzed ThroughTek's Kalay protocol using two different approaches. First, the researchers selectively downloaded and disassembled applications from both the Google Play Store and Apple App Store that included the ThroughTek libraries. These libraries typically do not contain debugging symbols. You hope they don't because that makes reverse engineering them by bad guys really easy. It also makes the apps much bigger. So hopefully the debug symbols have been stripped out before these

things were shipped. No debugging symbols required the team to also perform dynamic analysis with tools such as Frida, gdb, and Wireshark.

In addition, Mandiant purchased various Kalay-enabled devices. That is to say, you know, the baby monitors and the webcams and so forth. The team performed local and hardware-based attacks to obtain shell access, recover firmware images, and perform additional dynamic testing. These techniques included identifying UART/JTAG interfaces; performing chip-off attacks, meaning lifting the chips off and then analyzing them separately; and exploiting other debugging functionality present on the devices. So I just want to stop here and note, this is a lot of work that these guys went through. Why? Because it's a closed, secret, undocumented, proprietary network. No one ever looked at it before. That didn't stop it from being licensed. It should have, but it didn't. And as a consequence, 83 million devices already out there, unfixable, can be taken over remotely and be used to spy on people. Ooh, look what that camera is showing.

Anyway, over the course of several months, the researchers developed a fully functional implementation of ThroughTek's Kalay protocol. In other words, they reverse engineered and had to develop from scratch the entire thing by taking the things that were implementing it apart in order to figure out how it worked. This enabled them to perform key actions on the network, including device discovery, device registration, remote client connections, authentication, and most importantly, process audio and video data. They demonstrated all these attacks themselves.

Equally as important as processing AV data, the Kalay protocol also implements remote procedure call functionality. What could possibly go wrong? This varies from device to device, but typically is used for device telemetry, firmware updates - oh, maybe there is a chance this thing could be updated remotely - and device control. Again, depending upon the device.

Having written a flexible interface for creating and manipulating Kalay requests and responses, Mandiant researchers focused on identifying logic and flow vulnerabilities in the Kalay protocol. So not only did they first reverse engineer the whole thing and create their own implementation of it, then they said, okay, what's wrong with it?

"The vulnerability discussed in this post," they wrote, "affects how Kalay-enabled devices access and join the Kalay network. The researchers determined that the device registration process requires only the device's 20-byte uniquely assigned identifier, called a UID, to access the network. In Mandiant's testing, this UID was typically provided to a Kalay-enabled client, such as a mobile application, from a web UI hosted by the company that markets and sells a device model. Mandiant investigated the viability of brute forcing ThroughTek UIDs and found it to be infeasible due to the necessary time and resources." So we can thank our stars for that.

If an attacker obtains a UID of a victim Kalay device, which they noted can be done through passive eavesdropping, they can maliciously register a device with the same UID on the network and cause the Kalay servers to overwrite the existing Kalay device. Once an attacker has maliciously registered a UID, any client connection attempts to access the victim UID will be redirected to the attacker. The attacker can then continue the connection process and obtain the authentication materials, a username and password, needed to access the device. In other words, it's like made for man-in-the-middle attacks.

With the compromised credentials, an attacker could use the Kalay network to remotely connect to the original device, access audio-visual data, and execute remote procedure calls against the device. Vulnerabilities which exist within the device-implemented RPC interface can lead to fully remote and complete device compromise. Mandiant observed that the binaries on the IoT devices processing Kalay data typically ran as the privileged

user root and lacked common binary protections such as Address Space Layout Randomization (ASLR), Platform Independent Execution (PIE), stack canaries, and no-execute bits. Anyway, they provide additional details. They then post a video which demonstrates a functional proof-of-concept for this exploit, CVE-2021-28372. And they note that they will not be releasing any public exploit code.

They conclude by explaining: "CVE-2021-28372 poses a huge risk to an end user's security and privacy and should be mitigated appropriately. Unfortunately, nobody knows if they have the risk because nobody knows if their things are using this Kalay network. Unprotected devices, such as IoT cameras, can be compromised remotely with access to a UID, and further attacks are possible depending on the functionality exposed by a device." And as I've said, these things are typically connecting. We don't know where. Maybe back to China. So now we have the ability to remotely reprogram the firmware on 83 million compromisable devices.

You know, I get it that doing all that reverse engineering is their job, and that it presents an engaging challenge. But it is a big problem, as I noted before, that on the one hand a company like ThroughTek can produce and sell, I mean, is allowed to produce and sell under license an arbitrary protocol and SDK that is closed and proprietary and has undergone no external third-party security verification; and also that, on the other hand, the IoT vendors will purchase licenses to this sort of closed and proprietary technology. I mean, the damage that is pending out there is astonishing. It's become a theme of the podcast because it has to.

The Mandiant researchers listed a bunch of things that need to be done to tighten up the security of the system. ThroughTek has responded. But those 83 million webcams and baby monitors already sold likely won't be getting their firmware updated because it doesn't have to happen. It didn't have to happen that anyone made sure this stuff was secure in the first place. What makes us think that aftermarket sales are going to update baby monitors? Like I said, no one wants that crap sharing the same network as their PCs. When the IoT chickens come home to roost, you want to be a spectator and not a victim.

Leo: That's quite an image you just threw in my head.

Steve: Okay. Something cool for our listeners: Overlay Networks. It's clear from the rave reactions from all of those who took Tailscale out for a spin and adopted it easily within minutes, that the era of overlay networks is upon us. The first popular mainstream overlay network that we talked about was one that this podcast helped put on the map. That of course was Hamachi. And I loved it because back in the pre-IPv4 exhaustion days, it very cleverly used the unallocated and never before used Class A five-dot network for its overlay endpoint numbering. It was an overlay network. It ran over existing routable IP and created a virtual network whose IPs started with five dot something something something. Another emerging term we're going to be seeing is the notion of so-called "Software Defined Networks," so SDNs, or sometimes SD-WANs, Software Defined Wide Area Networks.

Okay. So Tailscale.com was the first one we talked about, and they appear to have nailed a solution to this need. As we've seen, their free tier for personal use has been hugely popular among our listeners. But for the sake of completeness, and because some people will prefer totally open solutions without any commercial component, I wanted to bring three other very similar offerings to everyone's attention.

Okay. Nebula. Nebula is a full, open and open source scalable overlay networking tool developed by Slack. Slack runs on Nebula. It focuses upon performance, simplicity, and

security to enable its users to seamlessly connect computers anywhere in the world. Nebula is portable. It runs on Linux, OS X, Windows, iOS, and Android. It can be used to connect a small number of computers, but it's equally able to connect tens of thousands of computers. And as I said, Nebula is what Slack runs on. It incorporates a number of existing concepts like encryption, security groups, certificates, and tunneling. Nebula didn't invent any of those individual pieces, obviously. They all existed before Nebula in various forms. What makes Nebula different from existing offerings is that it brings all of these ideas together, resulting in a sum, as they put it, that is greater than its individual parts.

Slack rhetorically asked: "What's the easiest way to securely connect tens of thousands of computers, hosted at multiple cloud service providers, in dozens of locations around the globe?" That was the problem they had. Slack's answer is Nebula. They wrote: "At Slack, we asked ourselves this very question a few years ago. We tried a number of approaches to this problem, but each came up with tradeoffs in performance, security, features, or ease of use. We will gladly share those experiences in future presentations and writing. But for now, just know that we did not set out to write software to solve this problem. Slack is in the business of connecting people, not computers." Anyway, unquote. What Slack found was that at the time they had no choice other than to build their own solution.

Okay. So what is Nebula? Nebula is mutually authenticated, peer-to-peer, software defined network (SDN) based on the Noise Protocol Framework. I'll explain that in a moment. Nebula uses certificates to assert a node's IP address, name, and membership within user-defined groups. Nebula's user-defined groups allow for provider-agnostic traffic filtering among nodes. Discovery nodes allow individual peers to find each other and optionally use UDP hole punching to establish connections from behind most firewalls and NATs. Users can move between nodes in any number of cloud service providers, datacenters, and endpoints, without needing - and of course ISPs - without needing to maintain a particular addressing scheme.

Nebula uses elliptic curve Diffie-Hellman key exchange and AES-256-GCM in its default configuration. And of course those are the same technologies and protocols I chose for SQRL. Nebula was created to provide a mechanism for groups of hosts to communicate securely, even across the Internet, while enabling expressive firewall definitions similar in style to cloud security groups.

To set up a Nebula network you need, one, the Nebula binaries for your specific platform. Specifically, you'll need Nebula-cert and the specific Nebula binary for each platform you use. And, two, optional, but you really should have at least one discovery node with a routable IP address which Nebula calls a "lighthouse." Nebula lighthouses allow nodes to find each other anywhere in the world. A lighthouse is the only node in a Nebula network whose IP should not change.

Running a lighthouse requires very few compute resources, and you can easily use the least expensive option from a cloud hosting provider. If you're not sure which provider to use, Slack wrote that a number of them, that is, a number of the people within Slack have used the $5 a month DigitalOcean droplets as lighthouses. Once a Nebula instance has been launched, ensure that Nebula UDP traffic - the default port is UDP 4242 - can reach it over the Internet, and you're good to go. That's it.

Okay. So of some concern is, wait a minute, the Noise Protocol Framework? Well, they did that right, too. We never talked about that before. The Noise Protocol Framework is at NoiseProtocol.org. The Noise Protocol Framework describes itself: "Noise is a framework for building crypto protocols. Noise protocols support mutual and optional authentication, identity hiding, forward secrecy, zero round-trip encryption, and other

advanced features." Right? All things we want. Open source implementations are available in C, C#, Go, Haskell, Java, Javascript, Python, and Rust.

And now here it is. Noise is currently used by WhatsApp and WireGuard and others. In other words, rather than building a solution on top of WireGuard, as some of the other software-defined overlay networks have, essentially creating a dynamic configuration manager for WireGuard which is an entirely acceptable solution, by the way Nebula drops down to a lower level to use the same Noise Protocol Framework upon which WireGuard was built. So there is no separate instance of WireGuard, and Nebula incorporates the features and benefits provided by WireGuard by virtue of being based upon the same shared framework.

Okay. That's Nebula. Another open source open solution is known as Innernet, is built upon WireGuard but, unlike Tailscale, open and open source. The Innernet developers explain: "Innernet is similar in its goals to Slack's Nebula or Tailscale, but takes a bit of a different approach. It aims to take advantage of existing networking concepts like CIDRs - classless inter-domain routing - and the security properties of WireGuard to turn your computer's basic IP networking into more powerful ACL (access control) primitives. This allows you to nurture and shape your own private networks with simple, free, open-source infrastructure." And they're hosted on GitHub, as is Nebula, by the way.

They said: "We had some simple goals: conveniences a typical WireGuard user wants - peer names, auto-updating peer lists, groups based on IP blocks, and automatic NAT hole punching; free, open source, and made to be self-hosted." They said: "We think it's especially important for such a vital, low-level piece of our infrastructure to not be dependent on the livelihood of a company no one has control over. And a straightforward architecture, no Raft consensus here. It's a simple SQLite server/client model." So that's Innernet.

And lastly, just for the record, there is another offering known as ZeroTier, T-I-E-R, dot com. Their slogan is "It Just Works." ZeroTier combines the capabilities of a VPN and an SD-WAN to simplify network management. They wrote: "Enjoy the flexibility while avoiding costly hardware vendor lock-in." They brag that it has speed. "Set up ZeroTier in minutes with remote, automated deployment. Flexibility: Emulates Layer 2 Ethernet, IP level, with multipath, multicast, and bridging," which is similar to all these.

"ZeroTier's zero-trust networking solution provides scalable security with 256-bit end-to-end encryption," which of course everybody has. And, they said, it's free for up to 50 endpoints. So Tailscale was generous with their free plan at a 20-node free limit. But if that was a little bit binding for someone, you can use ZeroTier - very much like Tailscale, I think, in every way - and go up to 50. But I know that there'll be lots of people who are interested in the free and open source alternative, so I wanted to share those.

Anyway, in summary, we've got Innernet, looking like a free and open source solution providing much of what Tailscale offers, both of them being built on top of WireGuard to give WireGuard a bunch of needed features. Nebula is a free and open source SDN written from scratch by the Slack guys because nothing that existed at the time did the job they needed. And Nebula is built using the same crypto framework as WireGuard, thus inheriting many of the same guarantees as WireGuard, and it also forms the backbone of Slack. So I would suggest that Nebula has already been proven to be scalable at global scale. Not that any of us need that for our own personal networks, but it's kind of cool. So anyway, there's four so-called Software Defined Networks, overlay networks. And I imagine that one of those will probably appeal to all of our listeners.

Okay. So two closing-the-loop bits. Bryan noted in GRC's Security Now! newsgroup, he wrote: "One thing Steve didn't mention is that in 2016 Avast also purchased AVG for 1.3 billion. So now Norton gobbled up Avera, Avast, and AVG." Anyway, I'm glad that Bryan

mentioned that. It happened on our watch in July of 2016. Avast purchased its fellow Czech cybersecurity company AVG for 1.3 billion, as Bryan noted. So we are seeing a clear merging and consolidation of the independent security add-on AV companies. For better or worse, that's happening.

And I received a DM from an individual, so I thought I would share it. He wrote: "Can I ask for an advice? Where do you see the future of IT? I've been working for the last eight years in different positions mainly in IT infrastructure, network and system. But now I want to invest my time and plan my future. Your podcast is a great library for understanding security. I want to start from the bottom and learn my way up."

And of course given where we are here with this podcast, while I may be somewhat biased, I can't imagine any segment of the IT industry that is safer to invest in for the future than where we are right now with security. I really believe this is it, that it's not like we're going to solve these problems anytime soon. There is no reason to believe that we're going to figure out how to do this within anyone who's listening's lifetime. So, yeah, I would say that an investment in IT security is one that would be long-lasting.

Leo: As we head into the final quarter of this episode, Steve is going to do a little thought experiment.

Steve: Okay. Not long after we finished recording last week's podcast, which as we all know was titled "Microsoft's Culpable Negligence," I had another thought. Last week I said over and over, and drove the point home, that given Microsoft's effectively unlimited resources, and having clear knowledge of new highly critical vulnerabilities handed to them, and of the devastating impact the exploitation of those vulnerabilities would have, I was unable to see any rational explanation for Microsoft's behavior, other than that they had to be performing a brutal cost-benefit analysis and rationally deciding not to fix those vulnerabilities, you know, not to take the time to fix them. One way or another, for one reason or another, this had to be a deliberate decision because there was no way to parse the history that we have all been witness to that wouldn't cause any unbiased observer to conclude that what has been happening was exactly what Microsoft had decided should be happening, insane as that at first appears.

Last week I stated that there could be no other reason. Then I thought of one. There is an explanation that perfectly maps onto all of the evidence and exactly predicts the behavior we're all witnessing from Microsoft. And in this proposed model, the driving motivation is, indeed, a brutal cost-benefit analysis, but one that's even more brutal than we imagined. It's just not the obvious cost-benefit analysis I was focused upon and described last week. Last week I was assuming that it would only be hostile and malicious adversaries who would be attacking users of Microsoft's software. Thus the "cost" in the cost-benefit analysis would be the attacks themselves. But what if, instead, attacks were the benefits? And what if those benefits arising from attacks were so beneficial that they outweighed the cost to Microsoft, which we've already determined to be effectively negligible?

So then we have to ask, how could attacks on users of Microsoft's proprietary software be beneficial? Such attacks would be in the U.S. national interest if they were being conducted by the United States domestic intelligence services against U.S. foreign adversaries. I recall mentioning on this podcast many years ago that Microsoft routinely tipped off our U.S. intelligence agencies about recently discovered and not-yet-patched flaws in Windows, and in their various other products. On Security Now! Episode 426, which we recorded on October 16th, 2013, I quoted Bruce Schneier from a piece he wrote for The Atlantic titled "How the NSA Thinks About Secrecy and Risk." I'm going to

read directly, verbatim, the first five paragraphs of that piece, which Bruce wrote nearly eight years ago.

He said: "As I report in The Guardian today, the NSA has secret servers on the Internet that hack into other computers, codename FOXACID. These servers provide an excellent demonstration of how the NSA approaches risk management, and exposes flaws in how the agency thinks about the secrecy of its own programs. Here are the FOXACID basics," Bruce wrote. "By the time the NSA tricks a target into visiting one of those servers, it already knows exactly who the target is, who wants him eavesdropped on, and the expected value of the data it hopes to receive. Based on that information, the server can automatically decide what exploit to serve the target, taking into account the risks associated with attacking the target, as well as the benefits of a successful attack.

"According to a top-secret operational procedures manual provided by Edward Snowden, an exploit named Validator might be the default, but the NSA has a variety of options. The documentation mentions United Rake, Peddle Cheap, Packet Wrench, and Beach Head, all delivered from a FOXACID subsystem called Ferret Cannon." He says: "Oh, how I love some of these code names." Then he says, in parens, "(On the other hand, EGOTISTICALGIRAFFE has to be the dumbest code name ever.)"

He says: "Snowden explained this to Guardian reporter Glenn Greenwald in Hong Kong. If the target is a high-value one, FOXACID might run a rare zero-day exploit that it developed or purchased. If the target is technically sophisticated, FOXACID might decide that there's too much chance for discovery, and keeping the zero-day exploit a secret is more important. If the target is a low-value one, FOXACID might run an exploit that's less valuable. If the target is low-value and technically sophisticated, FOXACID might run an already known vulnerability."

And here's the line: "We know that the NSA receives advance warning from Microsoft of vulnerabilities that will soon be patched." So he continues: "There's not much of a loss if an exploit based on that vulnerability is discovered. FOXACID has tiers of exploits it can run, and uses a complicated trade-off system to determine which one to run against any particular target. This cost-benefit analysis doesn't end at successful exploitation. According to Snowden, the TAO" - that's the Tailored Access Operations, of course we were all talking about that eight years ago - "operators running the FOXACID system have a detailed flowchart, with tons of rules about when to stop. If something doesn't work, stop. If they detect a PSP, a personal security product, stop. If anything goes weird, stop. This is how the NSA avoids detection, and also how it takes mid-level computer operators and turns them into what they call 'cyberwarriors.' It's not that they're skilled hackers, it's that the procedures do the work for them."

Okay. So in that fourth paragraph of that longer piece, famous security expert Bruce Schneier said: "We know that the NSA receives advance warning from Microsoft of vulnerabilities that will soon be patched." The revelations made by Edward Snowden and WikiLeaks stripped us of our innocence and matured our understanding of the true nature of the global cyber-intelligence world. Sometimes the need to gather intelligence requires, how shall I put it, an extreme lack of passivity.

So once again I'm being entirely serious about this. Think about it for a moment. Microsoft receives notification of a critical vulnerability from any of the world's many white hat hackers who are poking and prodding at their products. Say they get notice of a horrifically exploitable flaw in their email Exchange Server. The exploit is not publicly known, and its discoverer has agreed to keep it to themselves until sometime after it has been fixed.

So here's how this proposed timeline would play out. Microsoft thanks the security researcher hacker and promises to graciously throw them a bone by mentioning their

discovery in their eventual disclosure. Perhaps they'll also receive a bug bounty, but of course only if they remain silent until the problem has been fixed and a sufficient number of systems have been patched.

Next, Microsoft then uses their well-established quiet backchannel to pass the researcher's findings on to the NSA and the CIA. Microsoft takes no active part in the development of an exploit because that would be crossing a line. And should it ever become common knowledge that this early information was provided to U.S. intelligence services, Microsoft is simply being a good citizen and helping our own domestic intelligence agencies to guard against attacks which might exploit this now-discovered flaw, yet not publicly disclosed.

And now Microsoft sits on it. Remember how Victor Mata, who reported a remote code execution vulnerability with full system privileges, tweeted Wednesday before last, on August 11th: "Hey guys, I reported the vulnerability in December 2020, but haven't disclosed details at MSRC's request. It looks like they acknowledged it today due to the recent events with print spooler." So when Microsoft finds themselves in receipt of a valuable vulnerability, do they immediately assign a CVS number? No. That would raise suspicion and speculation and might start people looking. Do they post a note about a new vulnerability that needs fixing and their intention to do so? Nope. Not only do they remain completely mum, they do not patch it. It is only of use to our own, shall we say, "proactive" intelligence agencies who have been informed of it, so long as it remains unknown and unpatched.

All evidence suggests that the reasoning here is that as long as it has not been discovered and publicly reported as being actively exploited in the wild, it's just like any of all those other undiscovered vulnerabilities that exist within Windows, and we all know there's certainly no shortage of those. In the juicy case of Microsoft's Exchange Server, it's been sitting there, previously undiscovered, for more than the past 10 years. So what's a few more months? And just think of all the benefit that our domestic intelligence agencies can reap from it until, and if, it eventually comes to light on its own. Or perhaps, as Bruce observes, as an unfortunate side effect of its active use by our own NSA or CIA. What was it that we heard about the Exchange Server exploit? That it could allow an adversary access to all of the server's local communication history? Think that might be of interest to some of our snoopier spooks?

And, finally, consider that it wasn't until the ProxyLogon vulnerabilities were suddenly found to be exploited by adversaries actively attacking users of Exchange Server that Microsoft finally jumped to attention and rushed out an emergency fix for the problem. Remember that they claimed to be getting ready to release the fix with the next Patch Tuesday? Uh-huh. And remember that something didn't smell quite right about that at the time? And then, since this sudden disclosure apparently caught them by surprise, and they were unable to deny how long ago they had been originally been informed about it by our old friend Orange Tsai, they then hinted that someone they told might have leaked the details. Right. More like someone they told might have been caught privately exploiting it.

I frequently hear that we have listeners who have been with us from the start, for all of these past 16 years. And those people will have heard every podcast we've produced, and they will know that they've never heard me once jump onto a conspiracy theory. That's not the way I roll, and I certainly have no intention of doing so now. I have no firsthand evidence of this, and I'm not particularly interested in digging up any. As we all know, I've got much better things to do. But we also know that absence of proof is not proof of absence. And any system such as this would be well designed to remain off the books and under the radar.

I've got a chart here in the show notes. It was made on March 24th, several weeks after Microsoft's emergency patch release for the ProxyLogon flaws. In it we see, okay, this is two weeks after the patch was made available, on March 24th. On the chart we see 2,496 still vulnerable Exchange servers in Russia, and 1,473 servers still vulnerable in China. Again, that's weeks downstream. I wonder how many Exchange servers in Russia and China might have been vulnerable before Microsoft's revelation?

If nothing else, this resolves the apparent mystery of Microsoft's culpable negligence by converting apparent negligence into reasoned neglect. I guess the question remains, which would we rather have, a negligent Microsoft or a diabolically neglectful Microsoft? Neither places the interests of their own customers first.

Leo: That makes sense.

Steve: It does.

Leo: I mean, you never know, but...

Steve: It's sad.

Leo: ...it makes sense, yeah. Well, I mean, it's only a possibility. It's not necessarily what's happening. But it makes sense.

Steve: No, it's not. But if Bruce is right, and Microsoft is giving tips to the NSA, and the NSA has an interest in actively exploiting, I mean, and being proactive in their intelligence gathering, and I think we must all, given what we learned from Snowden and WikiLeaks, we must all understand that that's the case, I mean, we saw pictures of NSA nodes at central points of the Internet, you know, monitoring all the traffic that went through. We've seen pictures of the massive server farms next to sources of cold water so they're able to cool themselves. I mean, you know, this is big business. So how could they not, I mean, it would almost be malpractice for the NSA not to take advantage while the opportunity is there of the ability to get into remote Exchange servers and use them for gathering intelligence. How could they not? And then really, again, what's Microsoft's hurry? If it's really, really valuable, if it's useful for the U.S., if it's secret, and no one has discovered it in 10 years except one lonely researcher who's promised to keep it quiet with the carrot of a bug bounty once it's disclosed?

Leo: I don't see why Microsoft would do it. But okay.

Steve: Again, how else do we, you know, last week we explored how could it possibly be that they're not fixing this thing in three months.

Leo: Right, right.

Steve: And then when it's an emergency, they have a fix overnight. They immediately push out an emergency fix.

Leo: I think there are other explanations possible, as well, including just a sluggishness.

Steve: And not caring.

Leo: I would hope that Microsoft wouldn't just bend over to the NSA or FBI and the interest of national security over the security of all of its users. And that seems - certainly we know Apple has said no to the FBI in the past. That just seems like a pretty craven thing to do. But maybe they did. Maybe they did. It's possible.

Steve: Anyway...

Leo: We don't have any evidence of it, we should point out.

Steve: Nope. No evidence. I had no explanation for it last week. It seemed like, you know, how could they just not care to that degree? Because lord knows they have the resources to fix anything that they're, I mean, it's not - they don't even have to find them. They're being given these. How can they not fix it? I just - you know? And the only thing I can think is, well, that sure would be handy. Again, if there were nearly 2,500 still vulnerable Exchange servers operating in Russia two weeks after the patch had been issued, imagine how many there were before.

Leo: Right. I also think the NSA is unlikely to speak to a company and say we'd like you not to fix that for a while. I'm sure they have enough other exploits they could take advantage of. But maybe. Maybe. I'm not saying it's not true. It's a fascinating theory.

Steve: Interesting fodder, if nothing else.

Leo: Good fodder, absolutely. Thank you for a great show, as always. Steve does this every Tuesday, believe it or not. He's a hard-working fellow.

Steve: Woohoo.

Leo: You can watch us record the show after MacBreak Weekly. That's usually around 1:30 to 2:00 p.m. Pacific time, which would be 4:30 to 5:00 Eastern time, or 20:30 UTC. If you want to watch us live, TWiT.tv/live has a live audio or video stream. You can chat live with us at irc.twit.tv. After the fact, Steve's got copies of the show at his website, GRC.com. He has two unique kinds of versions of the show, a 16Kb audio version for people who really don't want to download anything too big. There's also an I think probably even smaller text transcription of it. He also has the 64Kb audio. That's all at GRC.com.

While you're there, check out SpinRite, Steve's bread and butter, world's best mass storage maintenance and recovery utility, now version 6; 6.1 is on the way. You can buy it now and get 6.1 for free, plus participate in the development of the next version of SpinRite. Lots of other great free stuff at GRC.com. Check it out. Leave

him feedback at GRC.com/feedback. Or you can tweet him. His DMs are open, as you heard, @SGgrc. We have copies of the show, audio and video, 64Kb audio and video, at our website, TWiT.tv/sn. There's also a Security Now! YouTube channel.

And of course the easiest thing to do is subscribe in a podcast application. That way you'll get it automatically the minute it's available.

Steve, thank you so much. Have a great week, and I'll see you next time on Security Now!.

Steve: Thanks, buddy.