DNS Over TLS

Description: eM Client CAN be purchased outright. An astonishing five-year-old typo in Mastercard's DNS. An unwelcome surprise received by 18,459 low-level hackers. DDoS attacks continue growing, seemingly without any end in sight. Let's Encrypt clarifies their plans for six-day "we barely knew you" certificates. SpinRite uncovers a bad brand new 8TB drive. Listener feedback about TOTP, Syncthing and UDP hole punching, email spam, ValiDrive speed, AI neural nets, DJI geofencing, and advertising in the "New" Outlook. A look into the tradeoffs required to obtain privacy for our DNS lookups.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-1010.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-1010-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. What an amazing find, a five-year-old typo in Mastercard's DNS. They say that's no problem. But is it a problem really? Also, 18,459 script kiddies get pwned. And then is it possible that neural nets like our own brains could, you know, attention could wander? Squirrel! Steve talks about that a whole lot more, next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 1010, recorded Tuesday, January 28th, 2025: DNS Over TLS.

It's time for Security Now!, the show where we cover the latest security news, privacy news, help you protect yourself and your company with this guy right here, the king of the hill, the king of security, Mr. Steve Gibson. Hi, Steve.

Steve Gibson: Back for Episode 1010. Which, as we noted, is binary 8. No, wait, no, 10.

Leo: Whatever it is, it's...

Steve: I've only been doing that for 55 years or something, so yes. Binary 10. 1010, binary 10, for the last episode of January, January 28th.

Leo: Wow, hard to believe. Here we are.

Steve: Where did the year go? Where is it going? What's going to happen? We don't know. Okay. So lots to talk about this week. Today's episode is titled "DNS Over TLS."

I'm going to share - if you were Microsoft, if we were Microsoft, I would call it "my personal learnings" because - I hate that when they...

Leo: Yeah, I don't know why they use that word.

Steve: God, it's...

Leo: It's awful.

Steve: ...so bad, yes.

Leo: Yeah.

Steve: And you can see that we suffered a power failure, which I have not yet reset things up again.

Leo: Oh, there's no blinking lights.

Steve: No, the blinking lights are frozen lights. But I imagine after our first sponsor announcement, the blinking lights will be blinking again because...

Leo: Magically.

Steve: ...that was over the weekend, and I haven't remembered to get them started again. I have to flip, well, you know. You've got blinking lights.

Leo: I have blinking lights, yes.

Steve: And whenever they stop they need a little bit of kick in the blink in order to...

Leo: You've got to reprogram the whole damn thing.

Steve: That's right. That's right.

Leo: Actually one - my PDP has stopped.

Steve: Ah.

Leo: But that doesn't mean it's frozen. It means it's solved the little number problem I gave it, and I have to give it a new one.

Steve: That's right, 42. It just it's...

Leo: Yes, it's 42.

Steve: 42, that's right. Okay.

Leo: So we'll restart. You'll restart yours, and I'll restart mine.

Steve: We're going to be talking about a lot of fun things. eM Client can be purchased outright. We have an astonishing five-year-old typo which was found, discovered by a security researcher in Mastercard's DNS, which, whoa, that was not good, and neither was their response. We have an unwelcome surprise which was received, so far has been received by 18,459 low-level hackers, also known as, in some circles, as script kiddies. DDoS attacks continue to grow, seemingly without any end in sight. We've got news on that front. Let's Encrypt has clarified their plans for six-day "we barely knew ye" certificates.

SpinRite uncovers a bad brand new 8TB drive. A little something I want to explain about that which occurred to a user of SpinRite, and I thought it would be fun to share that. We've also got a ton of listener feedback about TOTP, Syncthing and UDP hole punching, email spam, ValiDrive's speed, AI neural nets, DJI geofencing, and advertising in the new Outlook. And then, as I said, we're going to look into the tradeoffs required to obtain privacy for our DNS lookups.

Leo: Oy.

Steve: And of course, as always, oh, we've got another Picture of the Week that I think everyone will get a kick out of.

Leo: Awesome.

Steve: So, yeah, stand back.

Leo: Another great show. I haven't looked at it. I can see the caption, but I can't see the picture, and I won't. I'm going to preserve my virginity for a moment.

Steve: We're going to get a candidid - a candidid? A candid.

Leo: A candidid.

Steve: A candidid response.

Leo: All right, back we go. I'm ready for the Picture of the Week. Shall I scroll up in front of you and see?

Steve: Yup. As Benito said when he saw this before we began recording...

Leo: Oh, my god.

Steve: ...we've seen things like this before. I gave this the title "What do you mean you forgot to pack our Australia/New Zealand plug adapter?"

Leo: Oh, lord above.

Steve: Now, what we have here is a very clever - you have to give them credit. This is a very clever use of fingernail clippers. You know the kind of old-school chrome-plated fingernail clippers where you can swing out that little nail file part from the top? I mean, I'm sure everybody has seen those. It's like, sort of like, you know, the one design of the can opener which is immortal. Well, this is like that generic chrome-plated fingernail clipper where you can slide out the little filing portion.

Well, somebody apparently did forget their Australian/New Zealand plug adapter. That's the one that's got, you know, they all sort of look like a face. This one's got slanted eye slots and then the little grounding nose clot. But apparently they brought a regular U.S.-style straight-prong plug. Not deterred, however, they managed to use a pair of fingernail clippers to bridge from the slanty slots in New Zealand or Australia to the U.S. straight prong plug. And difficult to describe this. You'll have to see the picture. Anyway...

Leo: It's a mess.

Steve: Yikes.

Leo: And good lord, don't do this.

Steve: And Benito did mention, apparently these switches are - they switch the outlet on and off. And so you certainly would want the option to turn this outlet off...

Leo: That's a good point, yeah.

Steve: ...while you're setting up this disastrous - and, I mean, it's really, it's on the fringe right there. What I can't tell is whether this is a grounded plug that they're connecting to. If so, the ground prong...

Leo: Is missing. You need a paper clip.

Steve: Oh, that's...

Leo: That'll solve it.

Steve: That's right. We need one more exposed bare metal item.

Leo: That's all you need, yeah.

Steve: That's right. Wow.

Leo: Wow is right.

Steve: Anyway...

Leo: All right.

Steve: And I've already got next week's cued up. It's the return of the scissor lift because it turns out there have been some other creative applications. Oh, and Leo, last week's picture of the scissor lift on the float was, you know, some people suggested, yeah, maybe this was photoshopped. I've got pictures of it being set up. Like where it was actually being - this was being established. So anyway...

Leo: Interesting. So it's real.

Steve: We are going to keep having fun with our photos, everybody. Thanks to our listeners. This is entirely listener-generated. So thank you, all of our listeners who are sending email to securitynow@grc.com after registering at GRC.com/mail.

Okay. I have to start with errata because, Leo, thank god I have eM Client to help me manage the number of responses that I received from our listeners, basically saying variations of, "Uh, Steve, you know, that one big gripe you had about eM Client, you know, like which you recently fell in love with, is not actually a thing." So I wanted to say thank you to one and all. I have no idea how I missed the very clearly marked slider up near the top of the EM client's pricing page. But I certainly did. And now that I've seen it, it's impossible to unsee it. You know, every time I go to the page that's all I see is the big slider that says, you know, rent this or purchase it. And I am now, needless to say, the proud owner of a lifetime license with upgrades, updates forever, of eM Client.

And I was thinking about this, Leo. I know that you're, at least as regards TiVos as I am - back in the early days of XM Satellite Radio, they offered a lifetime license, which I purchased since I loved the concept of commercial free streaming music just coming down from the heavens. Later...

Leo: Plus that way you don't get all the emails from them saying, hey, it's time to renew. They're very bad about that. Uh-huh.

Steve: Yeah, yeah. Of course then later XM merged with Sirius.

Leo: Right.

Steve: And somewhere along the way the option to purchase a lifetime subscription, what do you know, it's gone. No longer there.

Leo: But yours is still active?

Steve: Yes.

Leo: Wow.

Steve: I still have mine, and I'm very glad that I made that choice many years ago. And I mentioned to you before, back when TiVos were the way to go, I know that you and I both always purchased the lifetime subscriptions for our TiVos.

Leo: Yeah. Of course it was the lifetime of that hardware, not anything else.

Steve: Yeah, I know. That was annoying, that when, like, because I had, you know, we all had Series 1 TiVos, and they became somewhat endangered at some point. Anyway, since I tend to stick with things until I'm forced to switch, you know, the approach of just, you know, putting the money in upfront and then riding it out a long way, that's always worked well for me.

So anyway, just to follow up on my raves about eM Client last week, I wanted to say I'm even more pleased now with my switch than I was then. And I heard from many of our listeners who were saying things like, "What took you so long?" You know, they had discovered eM Client years ago and similarly love it. So in addition to thanking everyone who wrote to make sure that I knew that it was possible to own it outright, and that it's 100% free to take it out for a spin for 30 days to see whether you might feel the same way about it as I do, anyway, in my opinion, you know, they really got the user experience right. And of course, Leo, you perked up upon hearing that it also fully supports end-to-end encrypted GnuPG email and address books. So that's in there, too.

So anyway, my entire reason for mentioning my own discovery of eM Client last week was to make sure that everyone at least had the opportunity to check it out. And, you know, if they, too, were feeling frustrated with their current solution, whatever they might be using, they would know about it. And that was a success.

Dan Taylor, one of our listeners, said: "Hi, Steve. I realize that you receive a ton of email these days, and your time is valuable. So I'll attempt to keep this short. I just feel the need to thank you for mentioning eM Client on the podcast. I hope you saw my message about the one-time purchase option they have. It's not all obvious on the pricing page, but it's there." And for what it's worth, I did have other people say they didn't see it either. So maybe the eM Client people could do a better job of - although they probably would rather, like, that you paid for it for the rest of your life every month. I think, what, like after four years is the breakeven point or something, so it's like, okay, I'm going to be using this well more than that.

Anyway, he said, Dan Taylor said: "I had no previous knowledge of its existence. In a nutshell, it's wonderful!" He said: "I have only one Gmail account. I also own two domains via Cloudflare, which forwards all email destined for those domains to my Gmail account." He said: "I've configured some aliases, one of which I'm using to send this to you. It's very cool." He said: "Also, I know you know this, but you've done an outstanding job on SpinRite 6.1. As I type this, my ZimaBoard is churning away on a 256GB flash drive that's been giving me problems. I've already run a Level 3 on another one, which improved its performance. Thanks again."

So Dan's need for only a single domain where he's got the other ones forwarding into it suggests that he may be able to use eM Client's free single-account offering forever and so never need to go - I've got four domains that I need, minimum. So anyway, I just wanted to close the loop on that. Thank you, all of our listeners. With the audience size we have, when I make a mistake like this, I get corrected. And so I'm happy to stand corrected on this because I am so happy that I own this thing.

Okay. This week's first piece of security news, as they would say in the UK, is "gobsmacking." Our friend, Brian Krebs over at KrebsOnSecurity, shared a wonderfully surprising piece of news last Wednesday under his headline "Mastercard DNS Error Went Unnoticed for Years." And before we go any further into that, exactly, into what exactly went unnoticed, I want to first highlight that it wasn't unnoticed for minutes or days or weeks or even months, but literally for years. Which is what, like, puts a sharp point on this.

Brian wrote: "The payment card giant Mastercard just fixed a glaring error in its domain name server settings that could have allowed anyone to intercept or divert Internet traffic for the company by registering an unused domain name. The misconfiguration persisted for nearly five years," he writes, "until a security researcher spent $300 to register the domain and prevent it from being grabbed by cybercriminals."

Now, Brian's article then posts the output of a DNS DIG command which returns the nameservers for a portion of the Mastercard.com domain. I have a screen shot of the command's output in the show notes. Even knowing that something is wrong with this picture, you would need to be sharp-eyed to catch the mistake. I missed it the first time I looked at it. I looked at the screen without having read the text yet because it's big on Brian's page, and I kind of just scanned over it. Okay. Looked okay.

Brian explains. He said: "From June 30th of 2020 until January 14th of 2025, thanks to the work of this security researcher," he said, "one of the core Internet servers that Mastercard uses to direct traffic to portions of the Mastercard.com network was misnamed. Mastercard.com," he says, "relies on five shared Domain Name System (DNS) servers at the Internet infrastructure provider Akamai. All of the Akamai DNS server names that Mastercard uses are supposed to end in 'akam.net' but one of them was misconfigured to rely on the domain 'akam.ne.'"

Yes, whoever created - and this is me talking. Whoever created, edited, or updated the DNS record for that Mastercard.com domain on June 30th of 2020, which lists the five authoritative DNS nameservers that should be referred to when looking up any IP address for Mastercard.com subdomains, made a tiny and earthshaking mistake, just a simple typo, when they were entering the names of the five nameservers. And it's as plain as day once you know what to look for.

The first nameserver is named "a1-29.akam.net." The second one is "a7-67.akam.net." The fourth one is "a26-66," who knows why those are the machine names, but ".akam.net." And the fifth one is "a9-64.akam.net." But the one in the middle of those five, the third one, is "a22-65.akam.ne." The final "t" of "net" was never entered. And boy, does that make a difference. Brian continues to tell the story, writing: "This tiny but

potentially critical typo was discovered recently by Philippe Caturegli, founder of the security consultancy Seralys. Caturegli said he guessed that nobody had yet registered the domain akam.ne, which is under the purview of the top-level domain authority for the West Africa nation of Niger." And Leo, in that picture on the screen, the third item of the five, very clearly, akam.ne. They dropped that final "t."

So Caturegli said it took $300 and nearly three months of waiting to secure the domain with the registry in Niger. After enabling a DNS server on akam.ne, he noticed hundreds of thousands of DNS requests hitting his server each day from locations around the globe. Now, I'm not sure about this. Brian wrote: "Apparently, Mastercard wasn't the only organization that had fat-fingered a DNS entry to include 'akam.ne,' but they were by far the largest."

Now, I don't know. Maybe he was seeing other DNS queries to other domains. Not clear to me. If so, that really makes you wonder how common these sorts of mistakes might be. Like it would be worth - I don't want to give bad guys any ideas; but, you know, there might be others. Brian said had he enabled an email server on his new domain akam.ne, Caturegli likely would have received wayward emails directed toward Mastercard.com or other affected domains. If he'd abused his access, he probably could have obtained website encryption certificates, I'm sure he could have, that were authorized to accept and relay web traffic for affected websites. He may even have been able to passively receive Microsoft Windows authentication credentials from employee computers at affected companies.

But the researcher said he didn't attempt to do any of that. Instead, he alerted Mastercard that the domain was theirs if they wanted it, copying this author, meaning Brian Krebs, on his notifications. A few hours later, okay, quickly, to their credit, Mastercard acknowledged the mistake, but said there was never any real threat to the security of its operations. Uh-huh. Right. A Mastercard spokesperson wrote: "We have looked into the matter, and there was not a risk to our systems. This typo has now been corrected." Okay. Now, I suppose technically it's true that there was not a risk to their systems. But there was certainly a serious risk to anyone who might be relying upon the security of Mastercard's systems, since that flew out the window with this typo, and that was five years ago.

Brian continues, writing: "Meanwhile, Caturegli received a request submitted through Bugcrowd, a program that offers financial rewards and recognition to security researchers who find flaws and work privately with the affected vendor to fix them." You know, in other words, responsible disclosures and bug bounties.

Brian says: "The message suggested his public disclosure of the Mastercard DNS error via a post on LinkedIn after he'd secured the akam.ne domain was not aligned with ethical security practices, and passed on a request from Mastercard to have the LinkedIn post removed. Caturegli said he does have an account on Bugcrowd, has never submitted anything through the Bugcrowd program, and that he reported the issue directly to Mastercard."

Caturegli wrote in reply: "I did not disclose this issue through Bugcrowd. Before making any public disclosure, I ensured that the affected domain was registered to prevent exploitation, mitigating any risk to Mastercard or its customers. This action, which we took at our own expense, demonstrates our commitment to ethical security practices and responsible disclosure."

Now, most organizations have at least two authoritative domain name servers. And that's true. That's what Brian wrote, and that's what I do for GRC. And that's typical. That's why most people will see two DNS servers in their own computers. This has always been done to create some redundancy for the sake of DNS lookup reliability. Brian says: "But

some handle so many DNS requests that they need to spread the load over additional DNS server domains." Which is also true. And in fact DNS deliberately responds when there's a list of available DNS servers. It will send them in round-robin fashion so that successive requests get a differently ordered list of nameservers in order to further cause them to get spread out.

So, you know, if they always listed the first one first, then everyone would just choose that one, and so you wouldn't really get much effect of having five. So in Mastercard's case, that number is five, so it stands to reason that if an attacker managed to seize control over just one of those five domains, they would be able to see about one-fifth of the overall DNS requests coming in. But Caturegli explained that the reality is many Internet users are relying at least to some degree - and this is what Brian is writing - on public traffic forwarders or DNS resolvers like Cloudflare and Google.

Okay, now, I would strengthen that statement a LOT to say that there is, I would argue, no one who is not relying upon caching resolvers. As we've often discussed on the podcast, caching DNS is critical. It's the only way this hierarchical system of distributed domain name resolution is able to function. You know, when you turn on your computer for the first time in the morning, and you go to Amazon.com, you're not hitting Amazon.com's nameserver to find out a list of IP addresses. Your ISP has obtained that from any other of the customers, of its customers who you are sharing the ISP's DNS server with. So it's in the DNS server's cache for, you know, eight hours a day, who knows how long. So caching is crucial for this whole process.

Caturegli said: "So all we need is for one of these resolvers to query our name server and cache the result." And here's the key. "By setting their DNS server records with a long TTL, which is the Time To Live, a setting that can adjust the lifespan of data packets on the network actually it's the lifespan of the DNS record which is cached throughout the DNS hierarchy - an attacker's poisoned instructions for the target domain can be propagated by large cloud providers."

He said: "With a long TTL, we may reroute a LOT more than just one fifth of the traffic." Okay, and so that's absolutely true. Typical TTLs are maybe an hour or two. Depends upon - it's entirely up to the discretion of the person who's setting up an entity's DNS. The longer the TTL that you publish, that is, how long you are telling the rest of the DNS-caching hierarchy out on the Internet, it can wait before it comes back to refresh your IP address. But the longer that is, the fewer requests you're going to get; right? Because a greater percentage of the request will be handled by all the caching out on the Internet.

So back in the, you know, two decades ago when GRC was first being a victim of DDoS attacks, I would decrease our TTL so that I could change IPs. Well, that's no longer feasible because it's not about changing IPs. Today's attacks just swamp the bandwidth. So there's no point in doing anything except just waiting. But if an organization's IP addresses are very stable, then it can make sense to set a TTL to it to 24 hours, for example. And many of them are. So, and in fact, if you try to set it too low, many caching resolvers will ignore a too-low setting and just set their own minimum, ignoring what you have asked for.

Anyway, Caturegli said he'd hoped that Mastercard might thank him, or at least offer to cover the cost of buying the domain. He wrote in a follow-up post on LinkedIn regarding Mastercard's public statement: "We obviously disagree with this assessment. But we'll let you judge. Here are some of the DNS lookups we recorded before reporting the issue." And then his post, which Brian quoted and has a picture of in Brian's own reporting, shows a sobering list of the queries that were coming into his .ne domain. We can see West Europe, East U.S., West U.S., AU Southeast, AU East, Australia East and more.

And remember that this DNS record was last changed and had been incorrect for the past four and a half years. So let's just say that if this had fallen into the hands of a malicious Russian or Chinese attacker, who have repeatedly demonstrated that they're looking for any advantage they can find over the West, the story we would be reporting today would have a very different ending.

That said, mistakes happen, and anyone can make an innocent mistake. I'm sure that's all this was. This was just that. At least Mastercard had the good sense and grace not to threaten this researcher who helped them significantly in return for nothing other than some recognition for his sharp eyes and the demonstration of his own integrity within his community. But wow. And again, the thing that really caught me out here was the suggestion that this wasn't just Mastercard.com queries that were coming in as a result of this typo, that that would suggest that there were other places where this Azure stub domain was being referred to, and somebody who was referring to - somebody else had it as .ne in their own DNS, not just Mastercard. Which, again, you really sort of wonder how many typos exist in DNS, and how many opportunities there are to get up to some real mischief.

You know, we've talked often, I mean, when Dan Kaminsky discovered that DNS recursive resolver queries had insufficient randomization in their queries, which allowed for their caches to be poisoned by bad guys guessing what a query would be and inserting a malicious response, that panicked the entire industry, so much that in a matter of 24 hours all DNS resolvers were updated like in a pre-planned staged secret update. I mean, it was that big a deal. This is that scale. So I hope the news gets out and people check their DNS records because, you know, a typo, as we've seen here, can go unseen for five years and could cause some real damage. Again, not to the company, but to the people who are relying on the security of its services. Wow.

And Leo, we're a little after, a little more than half an hour in. Let's take a break. And then we're going to look at what happens when script kiddies think they're getting away with something.

Leo: I like, what did you say, "low-level hackers"?

Steve: Low-level hackers. That's right.

Leo: All right, Steve. On we go. I've got a pie chart here.

Steve: Huh, yes. Last Friday the security firm CloudSEK, spelled S-E-K, disclosed the details of their investigation into an interesting attack that I don't think we've seen before. Get a load of what they shared. They wrote: "A trojanized version of the XWorm RAT builder - where RAT is the common abbreviation for Remote Access Trojan - has been weaponized and propagated. It is targeted specifically toward script kiddies who are new to cybersecurity and directly download and use tools mentioned in various tutorials, thus showing that there is no honor among thieves."

Okay, now, not that anyone ever thought there was any. Rather than going with the no honor among thieves theme, I think I might have chosen there's no such thing as a free lunch because these script kiddies think that they've found a hacked version of a commercial XWorm RAT builder tool. I saw one of the postings somewhere that said, you know, this is a cracked version, so you get to use it for free. Uh-huh. Right. Anyway, so the article goes...

Leo: How stupid do you have to be?

Steve: Well, that's what the hacker sites are full of, right, is this or that has been cracked, or here's the key for using it and so forth.

Leo: Right. You only do that once, I think.

Steve: Yeah. So the article says the malware is spread primarily through a GitHub repo, but also uses other file-sharing services, specifically the well-known mega.nz, upload.ee, two Telegram channels, and several hacker sites. It has so far compromised, and here it is, 18,459 devices globally, is capable of exfiltrating sensitive data like browser credentials, Discord tokens, Telegram data, and system information. The malware also features advanced functionality including virtualization checks, that is, to check to see whether it's running in a virtual machine and is thus being analyzed by researchers, virtualization checks, registry modifications, and a wide range of commands enabling full control over infected systems. Thus Remote Access Trojan, as the name goes, or RAT for short. Top victim countries include Russia, USA, India, Ukraine, and Turkey.

The malware uses Telegram as its command-and-control infrastructure, leveraging bot tokens and API calls to issue commands to infected devices and exfiltrate stolen data. Analysis revealed the malware has so far exfiltrated more than 1GB of browser credentials from these 18,459 devices globally.

Okay. So these wannabe hackers really are being hacked. Browser credential theft, as we know, allows the actual bad guys behind this to impersonate them on any websites where they're logged on. The article continues: "Researchers also identified the malware's 'kill switch' feature, which was leveraged to disrupt operations on active devices. Disruption efforts targeted the malware's botnet by exploiting its uninstall command. While effective for active devices, limitations such as offline machines and Telegram's rate-limiting posed challenges. Attribution efforts linked the operation to a threat actor" - now, so this is the guy behind the creation of the malicious malware, the mal-malware, he uses aliases "@shinyenigma" and "@milleniumrat" as well as GitHub accounts and a ProtonMail address.

They wrote: "The rise of sophisticated Remote Access Trojans has amplified cyber threats, with XWorm emerging as a significant example. Recently, a Trojanized XWorm RAT builder has been identified, being propagated by threat actors via multiple channels such as GitHub repositories, file-sharing services, and others." This was specifically targeted toward script kiddies who are new to cybersecurity and use tools mentioned in various tutorials. So, for example, YouTube tutorials, we're saying go here and get this. So this was a serious campaign deliberately looking for these, you know, as we said, low-level hackers.

They said: "Our analysis aims to provide detailed insights into the delivery, functionality, and impact of this Trojanized XWorm RAT builder. By leveraging data exfiltrated via Telegram," these researchers said, "we uncovered the infection sources, mapped its command-and-control mechanisms, and identified the breadth of its capabilities and the affected devices. Additionally, we conducted disruption activities targeting the botnet infrastructure to mitigate its operations." So they went further than just being a passive observer. They got proactive, which, you know, the legal issues there are a little shaky. Apparently you're able to do it, I think the last time we checked in, if you had some state-level agreement to do so.

But otherwise, you know, even if you're disinfecting other people's machines, technically you're still affecting other people's machines without their permission. So that's a little sketchy. But the malware that the script kiddies inadvertently installed and hosted on their own machines, believing that they were obtaining a cracked copy of the well-known XWorm Rat builder, is able to obey commands such as the /browsers command, which steals saved passwords, cookies, and autofill data from their browsers; /keylogger, what its name sounds like, records everything the victim types on their computer; /desktop captures the victim's current screen; /encryptpassword encrypts all files on the system using a provided password.

/Processkill terminates specific running processes, which would typically be security software. And then there's the upload file, which exfiltrates specific files from the infected system. And 50 other commands, in total. So it's, you know, a very complete command set.

What struck me is that there is such a large, okay, this was like first blush. Such a large and thriving ecosystem of low-level hackers who apparently aspire to be running their own botnets. 18,459 specific known instances where this trojan trojan was downloaded, installed, and run. 2,478 of them are located in Russia. But the U.S. is the runner up with 1,540 installed instances. Now, I suppose when you consider the size of the world and the number of kids who are probably enamored of the idea of being, you know, a stealthy Internet hacker, it's understandable. And when you consider the viewpoint of the more sophisticated hacker who created this double-cross, your targets are easily baited with low-hanging fruit. They think they're getting something for nothing. And, well, boy, are they! They are installing really bad malware into their own machines, thinking that they're getting a malware builder for free. So anyway.

Leo: Wait a minute. Let me get this straight. Script kiddies who wanted to install a remote access trojan on their systems installed a remote access version on their systems.

Steve: Exactly.

Leo: Okay. Bit by their own swords.

Steve: They thought, exactly, hoisted by their own petard.

Leo: That's hysterical.

Steve: They thought that they were going to be getting a worm-based remote access trojan system in order to create their own botnets.

Leo: Yeah.

Steve: And they became, you know, a victim of somebody else's effort to infiltrate their systems. So whoopsie.

Leo: Unbelievable. Yeah.

Steve: Speaking of botnets generating widespread attacks, Leo, we have set a new record.

Leo: Oh.

Steve: Yeah. Last Tuesday, Cloudflare updated the world on the state of Internet DDoS attacks by publishing their 20th quarterly report since they began quarterly reporting in 2020. I've got a link on the next page, top of page 9. You may want to just bring that up on the screen while I'm talking about this because this thing, I'm only going to touch on it, that's why I've got the link, and I've mentioned it several times because there are so many interesting charts and graphs in this thing.

Okay. So today's DDoS attacks records appear to be - the DDoS attack records, the size of today's DDoS attacks at this point appear to be broken just for the sake of breaking them. By that I mean that hitting anyone with, get this, 5.6 trillion bits of traffic per second - per second. 5.6 trillion bits of attack traffic per second, well, it's massive overkill. I mean, the only exception to this would be if one were stubbornly trying to attack a site that was being protected by a leading DDoS mitigation service, you know, such as Cloudflare. And this is their quarterly report.

And in fact that is what happened. During the week of Halloween, at the end of October 2024, Cloudflare's DDoS defense systems - and to me this is astonishing - successfully and autonomously detected and blocked that 5.6 terabit per second DDoS attack, registering the largest attack ever reported. And somewhat incredibly, the company paying for Cloudflare's DDoS attack prevention services remained online and blissfully unaware that anything had even happened.

Leo: Wow. That's amazing.

Steve: It's incredible. So in their report, which as I said I've linked to in the show notes for anyone who's interested, they note that in 2024, Cloudflare's autonomous DDoS defense systems blocked around - and here's a number that'll sober you up quickly - 21.3 million DDoS attacks, 21.3 million DDoS attacks representing a 53% increase compared to 2023. So 2024 saw a 53% increase in number of attacks compared to 2023. And it's the botnets; right? I mean, it's - unfortunately there are lots of botnets, and it's not difficult to enlist them just to throw garbage at a given IP, and to knock those IPs off the 'Net. They said on average in 2024, Cloudflare blocked 4,870, okay, 4,870 DDoS attacks per hour. Nearly 5,000 DDoS attacks per hour.

Okay. And that's not all of the Internet; right? That's not all the Internet. That's only the attacks against Cloudflare, its infrastructure, and its customers. That means that worldwide the DDoS attack rate will be many, many times more, since Cloudflare is only protecting a tiny subset of the entire Internet. Nonetheless, nearly 5,000 attacks per hour, 21.3 million DDoS attacks last year, just for Cloudflare.

Also they noted: "In the fourth quarter, over 420 of those attacks" - 420 in the fourth quarter of 2024 - "were what they're now terming 'hyper-volumetric,' exceeding rates of one billion packets per second, and over one terabit per second. So one billion packets per second, and one terabit per second. 420 of those were hyper-volumetric. And the number of attacks exceeding one terabit per second grew by a staggering 1,885% quarter over quarter." In other words, there's been an explosion in the number of these

high-volume, greater than one terabit per second attacks from the same quarter in 2023 compared to the fourth quarter in 2024.

And about this record-breaking attack, they wrote: "On October 29th, a 5.6 Tbps UDP DDoS attack launched by a Mirai-variant botnet targeted a Cloudflare Magic Transit customer, an Internet service provider from Eastern Asia. The attack lasted only 80 seconds and originated from over 13,000 IoT devices. Detection and mitigation were fully autonomous by Cloudflare's distributed defense systems. It required no human intervention, did not trigger any alerts, and did not cause any performance degradation. The systems worked as intended."

Then they added about this attack: "While the total number of unique source IP addresses was around 13,000, the average unique source IP addresses per second was 5,500. We also saw a similar number of unique source ports per second. In the graph below" - and I have this below on our next page in the show notes - "each line represents one of the 13,000 different source IP addresses. And as portrayed, each contributed less than 8 Gbps per second on average. The average distribution of each IP address per second was around 1 Gbps." And this is just, I have it at the top of page 10 in the show notes. It's just a beautiful chart. So you need to see the show notes to appreciate this.

But every line is one of the bots. And so there's 13,000 of these little thin lines. And I have to say this also represents astonishingly good control, you know, I don't want to give credit to the bot herders, the bot masters. But like to bring up an attack - the earlier chart that you showed from their page, Leo, that showed just basically a big square wave. The attack began with a sharp edge. It almost immediately came to full strength. It lasted for 80 seconds, and then it immediately shut off.

Leo: Oh, so that's only 80 seconds.

Steve: Yes.

Leo: So it's a test.

Steve: Well, yes. Exactly. And in fact in some other reading that I've done, DDoS attacks are often being aimed at people who are capable of measuring them because they want to know...

Leo: We just want to show we can do this.

Steve: Yes. And when you think about it, they don't know.

Leo: Right, sure.

Steve: They're commandeering routers. They're grabbing routers and NAS boxes and random crap.

Leo: These are from Mirai. This is all a Mirai bot. That's amazing.

Steve: Yes, a 13,000-agent Mirai botnet did this. And I mean, this melts wires. I mean, it's crazy.

Leo: A lot of data.

Steve: And it just gets - this is crazy.

Leo: It's also very impressive, and of course that's why Cloudflare writes the blog post, that they were able to mitigate this 100%.

Steve: Yes. If you were like a gambling site or, you know, because...

Leo: Big ad for them.

Steve: It is a big ad for them. I would argue they deserve it. And of course they're not the only people who are able to do DDoS attack mitigation. We've named a bunch of them before. I think Akamai has a service. I think Microsoft offers a service.

Leo: Amazon does, yeah.

Steve: Yes, Amazon does. So, you know, there are alternatives. But, wow, just 5.6 terabits, trillion bits per second. Per second.

Leo: Yeah.

Steve: Wow. Twelve days ago, on January 16th, Let's Encrypt posted their formal announcement, which we had a preview of a few weeks before that which worried me a bit. On the 16th they posed their formal announcement of their plans for 2025. And a sincere "Thank you" to one of our listeners for pointing me to this. I'm glad to know this and to be able to share this.

The opening paragraph of their announcement says: "This year we will continue to pursue our commitment to improving the security of the Web PKI by introducing the option to get certificates with six-day lifetimes (short-lived certificates). We will also add support for IP addresses in addition to domain names. Our longer-lived certificates, which currently have a lifetime of 90 days, will continue to be available alongside our six-day offering. Subscribers will be able to opt in to short-lived certificates via a certificate profile mechanism being added to our ACME API."

Okay. So I am grateful for this welcome clarification. As our listeners know, I question whether this is actually solving a real problem with the industry's PKI, our Public Key Infrastructure. And it exposes, you know, it does expose its users to some threat of connectivity outage if anything should occur to prevent a timely ACME certificate renewal. But that said, why not offer it, as long as it's not mandatory? This places a huge burden on anyone offering such short-term renewals. It's very much like the analogy I

just drew with DNS. DNS depends on caching in order not to load down the DNS nameserver. If it didn't have it, it would have to be fielding all these requests.

Well, certificate lifetime is very much like caching the credentials out on the web server, which otherwise has to come back and get updated credentials within, you know, before its cached credentials, the lifetime of its certificate expires. So if you're shortening that, you're shortening, you know, you're requiring all of the web servers that are opting to do this to come back much more often. But okay. If they want to do it, fine. As long as they don't make everybody do it.

So, you know, again, I just - I don't know what's driving this. The fact that they're willing to put this huge burden on themselves suggests that there must be some problem. Maybe there are people who are being kept up at night worrying about the theft of their web server authentication certificates and who place no faith in the ongoing move to client-side Bloom-filter-based revocation enforcement, which we talked about last year, toward the end of last year, and which is in place and working and being increasingly relied on.

Anyway, the Let's Encrypt statement included a timeline. They said: "We expect to issue the first short-lived certificates to ourselves in February of this year." So in a few days. "Around April we will enable short-lived certificates for a small set of early adopting subscribers. We hope to make short-lived certificates generally available by the end of 2025." So not tomorrow. Hope to make short-lived certificates generally available by the end of 2025. Again, this is going to require some scaling up of their infrastructure in order to pull this off. And they finished: "Once short-lived certificates are an option for you, you'll need to use an ACME client that supports ACME certificate profiles and select the short-lived certificate profile, the name of which will be published at a later date."

So this is, you know, very much still nascent and on its way. I did hear from a listener of ours who received the show notes last night where I was talking about this. He said that something had changed just recently with the Let's Encrypt certbot because he was having an email connectivity problem. It turned out that they defaulted, they changed the default to elliptic curve certificates from RSA. And it was necessary to explicitly specify that you wanted RSA certificates because he was having connectivity problems with other servers who were not able to support elliptic curve crypto.

So just a heads-up for anybody who might be caught out by the same thing. I don't have a sense of timeframe for when this happened to him, but I got the sense that it had just happened, and he was having an email outage as a consequence of an updated Let's Encrypt certificate having changed its certificate in a way that other email servers were having a problem connecting to. So there's another sort of gotcha for that.

I want to share a SpinRite story that I think everybody will find interesting, Leo. But we're at an hour in, so let's tell our listeners why we're still here.

Leo: Yes. Indeed, indeed.

Steve: On the air, as it were.

Leo: On the air. How does it manage to stay on the air? SpinRite story.

Steve: Well, I haven't mentioned SpinRite for quite a while since I haven't had anything new to share. We all know of the discovery that the fronts of SSDs, where the operating

system files live, slow way down after years of use, and that a single Level 3 SpinRite pass will restore the drive's original performance. I receive ongoing reports of that, and I've posted some of them over on SpinRite pages. But, you know, it becomes redundant after a while.

I'm mentioning SpinRite today because last week we received a report that I did want to share. A generic SpinRite user wrote to my tech support guy Greg. He said: "Hi, Greg. I bought four Western Digital Red Plus 8TB hard drives for a Zima Cube and wanted to check their operation before installing. The first two" - that is, the first two of his four drives - "passed SpinRite Level 3 in about 28 hours, each with no errors. The third got 80% through, but then started showing problems through the SMART screen. By 94%, which took 106 hours, there were 216 bad sectors, 379 minor issues, 6,845 command timeouts with the status screen showing four 'B's' for bad regions." He said: "I'm running the fourth WD 8TB drive on a Zima board. Like the first two drives, it's having no trouble at 68% and should finish before the bad third drive," which I guess was still chugging away and struggling.

So then he had questions. He said: "Questions. Would you return this third drive showing the problems? What do Command Timeouts mean? How do I know how many spare sectors remain for future swapping out?"

Okay, now, the big news here is the picture that he included. He took a picture of that third drive's SMART system monitor page in SpinRite. Now, this is what this one drive was showing him about itself. And what we see here is a brand new drive that's in serious trouble. The whole SMART system, you know, S-M-A-R-T, Self Monitoring Analysis and Reporting Technology, has always been a mixed blessing because it's never been a strong standard. In fact, it's an extremely weak standard. I would argue it's really not much of a standard at all. What's standardized is the way to access the drive's SMART data.

What's never been standardized, because there was never any way to force its standardization, is the precise meaning of the various things a drive may choose to report about itself. As a result, large databases have been assembled by volunteers, and they're being maintained on a volunteer basis to show what this or that specific drive's make and model means with this or that SMART parameter.

But that said, the one thing that is universally understood is that the drive's summary "Health" parameter has the meaning that the more positive it is, the better. You know, UP is good, DOWN is bad. So the screen that we see tells an unambiguous story. It shows us that the drive ITSELF is - this is not SpinRite saying this, and that's what's key here. The SMART is Self-Monitoring Analysis and Reporting Technology. The drive itself is saying that three clearly crucial parameters - the amount of ECC error correction being needed, the rate of bad sector relocations, and the number of relocation events, those are those three red bars shown here - they are reflecting a drive that is in serious trouble. That is, the drive itself is saying I am in serious trouble.

SpinRite is showing those three SMART parameters in RED bars because what it does is it holds the maximum positive health value it has seen since it was started. And any subsequent drop in those values, which again, down is bad, up is good, so any subsequent drop in those values is shown in red because that's never good.

The screenshot also shows us that many other SMART health parameters the drive is reporting have remained pinned at their peak of 100%. Sector seek errors, recalibrate retries, cabling errors, uncorrectable errors, write errors, and pending sectors are not worrying the drive at all. They're all sitting at 100 out of 100 or 200 out of 200. But "ECC corrected" has dropped to -50 out of 149, "Sectors Relocated" is at 30 out of 200, and "Relocation Events" is down to one out of 200. These all reveal that something is very

wrong with this drive. So the question is not "Should I return it?" but "How quickly can I return it and get it replaced?" I mean, this was just, you know, it's a bum drive.

And this brings me to the first of two points I want to make. If a drive is just sitting there doing nothing but spinning happily away, it will be quite fine. Many other SMART monitoring tools have been created, and they can be useful. But it's important to really understand that if a drive is not being asked to do any work, if it's just sitting there happily spinning away, then the drive's sunny disposition doesn't have the same meaning as when it's still smiling while doing what a drive is there to do, which is reading and writing data. You know, human doctors who want to test someone's cardiac function put their patient on a treadmill because it's only when the patient's heart is under some load that its response to that work can be determined. Resting state is also useful, but it doesn't tell the whole story.

And here's the second point I wanted to make: This SpinRite user purchased four drives, and only one of the four was brought to its knees just by asking the drive to read and write during a Level 3 SpinRite pass. It's not as if this is some sort of torture for a drive. SpinRite is not abusing a drive in any way. It's just saying, "How would you feel about doing some reading and writing?" You know? Three of those identical drives, all purchased - all four purchased at the same time. Three of them respond by saying "Sure thing," while one of the four is really very unhappy about being asked to do what it was designed to do.

You know, and I've shared the story before, both from hearsay and also from people who have reported from having been there themselves, that in the early days the famous IBM PC cloning company, Compaq, would over-order the number of drives they needed, then use SpinRite to pre-test those drives before putting them into service. Any drives that didn't make the grade were returned. Since those drives technically worked and would have passed the manufacturer's QA testing, I imagine somebody else wound up with Compaq's rejects. But nobody wants that.

So it's interesting that, even though today's technology could hardly be more different, and we're talking about 8TB drives, eight trillion bytes on a drive, rather than 30 or 40MB back in those early Compaq days, some things have still not changed, and SpinRite has remained useful for performing pre-deployment hard drive testing. And actually I know that that's what a lot of SpinRite's users do with it. So just a perfect case in point of that, you know, yeah, you can look at a drive's SMART data when, you know, you turned it on, and it's been sitting there for a while. That'll tell you a few things. But you need to ask it to do some work and see how it feels about its own ability to do that. And this drive, you know, this needs to be replaced.

Okay. So a listener of ours, Stephen, says: "Hi, Steve. Another incredible podcast breaking down one-time passwords, but I'd like to drop a spanner in the machine. Sorry. If an attacker is trying to brute force a one-time password, they already have the user's creds, which means the code space is reduced to one million, the weakest link in the chain." Okay, now what he means is that there's only - there is one in a million possible correct answers if you're trying to log in. We know that's true, six digits.

He says: "In theory, a bad actor could easily spin up a few hundred cloud instances and distribute the two-factor authentication attempts across them. Multiple simultaneous attempts within the 30-second time window doesn't have to get the one-time password the first time, but given enough resources would likely succeed. Obviously the server could throttle login attempts per account, but no server admin is perfect. Just a thought. Best regards, Stephen."

Okay. So a number of our listeners shared variations on this theme. So I wanted to take a moment to mention that last week's challenge was not so much about defeating

multifactor authentication once in order to log in as a user, but rather to examine the theoretical requirements for cracking an authenticator's secret key. That was what we were trying to do. After writing and sharing that last week, I've been thinking about it since. I realized that there's a somewhat clearer and simpler, cleaner way to think about the entire thing. Since it's a different construction of the same solution, I want to share it. Won't take long. I think it's sort of a distillation of what we talked about.

Okay. So first, we once again assume that we have some set of sample outputs from an authenticator, where each output is a six-digit code and the time of that code, that code's timestamp. So for any given 80-bit candidate key, there will be a one-in-a-million chance that the candidate key will produce the same code as the authenticator for the same timestamp. The key we seek is the one that produces the proper authenticator code for every timestamp. So we get a new candidate key, and we start testing it against each of the authenticator samples we have, authenticator output samples we have. The right key will match all of them. And since there's always a one-in-a-million chance for any match, that means that non-matching is always a near certainty. Except for one in a million times, we're not going to get a match.

So as we test a new candidate key against our set of samples, each successful match allows us to be one million times more certain that we have found the one proper key that will match every sample we can test. Since 80 bits allows for - and here it comes - 1.2 million million million million keys, this makes very clear why we need at least four sample matches, and why a few more would be good, just to make sure.

Anyway, that seems like a distillation of my longer exposition of this last week. Every sample that you can test against makes you a million times more sure that you've got the right key since there's only a one-in-a-million chance that the right key will work. And since there's four millions times 1.2, if you're able to test four different keys, you're a million times more sure, four times. So you're getting pretty sure at that point. But a few more would be good. Anyway, I wanted to acknowledge Stephen's other point, which was that the authentication service on the receiving end of many failed guesses would be expected to limit and throttle the number of those a user would be allowed to make. It would seem a bit far-fetched for that not to be done if we hadn't recently covered Microsoft's own multifactor authentication systems having made exactly that mistake. So some great points from our listener, as always.

Joe Havlat, he said, on the subject of Syncthing and UDP hole punching: "Hi, Steve. Thank you for all the time and effort you and Leo put into the Security Now! podcast. I look forward to listening to it every week. I end up using a lot of software and services you mention on the show, and Syncthing is one of them. In the past I've used Tailscale to access my 'internal' devices remotely, including devices I use Syncthing on. I recently decided to try something other than Tailscale. And after I removed it from my devices, to my surprise, Syncthing continued to work!" Right.

"After looking at the settings and doing a bit of reading, it appears that Syncthing was making QUIC connections leveraging STUN for a 'direct connection.' I believe this is similar to how Tailscale gets around NATs? Anyway, as my eyes were glazing over while reading about STUN, I thought this might make a good topic for one of your 'propeller hat' discussions. If you could find the time to discuss this in one of your future episodes, it would be greatly appreciated. If not, no big deal. You always seem to come up with something that piques my interest. Thanks again, Joe."

Okay. So I was certain that we once had a podcast titled "STUN & TURN," but I was unable to locate it. I did locate a reference to that, that phrase, in podcast #443, which was titled "Sisyphus," where I said: "And they use, in order to do NAT traversal, we've talked about NAT traversal in the past, there's the so-called 'STUN' and 'TURN' protocols." But given my inability to locate a podcast with that title, perhaps I've only

ever referred to it in passing. So, Joe, if that's the case, I agree that it would make a terrific and still very relevant deep dive topic because NAT traversal is something as important today as it ever was. So thank you for that.

Jason Harris said: "Hi, Steve. After hearing you talk about switching to eM Client for email, I decided to check it out. Currently I'm using the built-in Mail apps on macOS and iOS to manage my personal Gmail and Yahoo accounts. While they work fine for my needs, I'm curious about what other email clients have to offer. That leads me to a question." And Leo, this would be one I'd like to hear you weigh in on. He asked: "Do you have any recommendations for email providers? Over the years I've noticed that my Yahoo account in particular has been receiving more and more spam. I suspect this might be due to how long I've had the address and how many services I've linked to. Thanks for any insights you can share. Best regards, Jason."

Okay. So I first want to say that many, many years ago, and I know that you and I talked about this at the time, Leo, I spent some time looking at the spam problem. A very techie coder buddy of mine, Mark Thompson, and I developed a Bayesian filter for spam that was pretty much state of the art at the time. Now, this was back in the famous John Dvorak "I get no spam!" days where, as I recall, John was stating that his ISP was so good that he got no spam. Meanwhile, I was being buried under an avalanche of spam since my email address at the time was just "steve@grc.com." Yikes.

I will never forget the time I enabled real-time logging for GRC's email server and watched foreign SMTP servers connecting to GRC and just running down an alphabetic list of account names using people's proper first names. I mean, starting with "A," running through, you know, like Abigail and Annette and so forth. I realized that it wasn't only that my email address had "leaked," though I'm also sure by then that it had. It was that my email account name was just likely to be valid because it was just my name. So it was clear that I needed something uncommon.

The other thing I wondered was how long it would take for an uncommon email address to escape into a spammer's hands, or the Internet's spammers' hands widely. And this is where Jason's thought of "I suspect this might be due to how long I've had the address and how many services I've linked to" comes in. What I started doing at least 15 years ago is changing, deliberately changing my email address annually. I'll keep forwarding all previous years' email account names into my current email so that I don't miss those. But anything I generate will be from the current year, so an awareness of my current email tends to migrate forward sort of organically. And if at some point some annoying spammer does start using an older email account, and if I'm unable to unsubscribe from that, I'll just delete that old account's forwarding into my current account.

And here's the surprising breakthrough that this allowed me to discover: I don't understand why, to this day I don't, but it appears to take spammers many years to obtain and/or to begin using an email address.

I often remember John Dvorak's "I get no spam!" proclamation with a smile since now that's also true for me. GRC runs with zero spam filtering. None. And spam is not any problem for Sue or Greg or me because all of our email addresses are rotated annually. I truly do not understand why this is so, that is, that it works as well as it does. But it does. And it's also been confirmed by others with whom I've shared this simple discovery. So if you're able to periodically change your email account, I believe you'll be quite surprised to see how long it takes for that new account to be discovered and despoiled by the world's email abusers. A few years from now, let me know. And Leo, any thoughts about email services?

Leo: Well, yeah, I mean, most people can't do that because, you know, that would mean that they wouldn't get email, basically. I mean, you don't care, I guess. But we rely on email for so many things, and it's not convenient to say to everybody who sends us email, oh, change our address every year. So people keep the same email. They're going to do that. And honestly, this guy, if possible, I don't think there's any service that provides effective email filtering. Dvorak's "I get no spam" goes back many years to this company. And if you look at their website, you can see how many years old this is. I think they're still around, Junkemailfilter.com. So it was on top of his email provider.

I think spam is for most of us just a fact of life, and there are all sorts of ways, I mean, what I do is I have an email box that checks against my contact list, and that box is the first one I look at. But inevitably I have to go through the spam folder, you know, every few weeks to make sure I haven't missed anything. I think spam is just - I don't know if there's any real way to avoid spam except do what you do, but which is impractical for 90% of our...

Steve: No, all my previous years still come to me, Leo. That's what I said. I'm forwarding all of those previous emails.

Leo: But don't you get spam on that email?

Steve: No. That's what's bizarre.

Leo: On the older email.

Steve: I don't understand why. So people still write to me on old addresses, comes through with no trouble at all.

Leo: Oh, that's interesting.

Steve: Anything I generate goes out on today's email. So anyway, I invite our listeners to give it a try.

Leo: There's a puzzle there. That's an interesting idea. So you still get all the old email, but no spam comes on your address from 2008.

Steve: No.

Leo: I think you're just lucky. I don't know how you do that.

Steve: Just reporting what works for me.

Leo: Yeah, that's interesting.

Steve: And has worked for others.

Leo: That's interesting.

Steve: Yeah.

Leo: All right.

Steve: A customer of ours, Jeff Parrish - customer. I don't mean a customer. A listener and also a user of freeware of GRC's. Jeff Parrish wrote: "I purchased a 10-pack of PNY 16GB thumb drives. This is the results I received on two of them so far. I will be checking all 10."

Now, he attached to his email a screen shot from ValiDrive's display for two of the 10-pack of the 16GB PNY thumb drives he purchased. He pointed out that whereas he believed he was only purchasing 16GB drives, what he received were 32GB drives that fully pass ValiDrive's scrutiny. So that was cool. I mean, you know, he got twice the drive for the price. And really it makes sense because sub-terabyte thumb drives have become commodity items. So there's actually no cost difference to the supplier between 16GB and 32GB media. You know, who would ever imagine the day that that would be true? And frankly, this is one of the reasons why Apple's device pricing always rubs me the wrong way. They are charging so much more for double or four times the memory, as if there was any marginal cost difference for them. Or nearly that. There just isn't. But, you know, that's the game they're playing.

Okay. But aside from that, what really stopped me in my tracks about Jeff's thumb drives was the total time spent reading and writing. ValiDrive performs a pseudorandom spot-test by reading and writing 1152 4K regions, 4Kbyte regions, uniformly spread across the drive's self-declared size. As the drive, you know, the size the drive declares itself to be, which is if it's faking its size, we see whether it's telling the truth or not and find that we're unable to read and write spots that it says should be valid. And thus ValiDrive's purpose. So ValiDrive reads and writes, rereads and rewrites and finally reads again each location, gathering statistics while it's doing this. During this process, a grand total of 3.6 seconds, that is, on Jeff's drive, 3.6 seconds total was spent reading, whereas 1307.8 seconds was spent writing. Okay. 3.6 seconds spent reading; 21.8 minutes spent writing.

Now, we know that NAND flash memory is fast to read and slower to write. But this is 362 times slower to write. I believe we're going to find that the better way to express this is that the bulk of this time was spent waiting to begin writing. We know that writing to NAND flash memory requires pushing electrons through an insulating barrier so that those electrons are then stranded as an electrostatic charge on an insulated floating gate. In order to read bits, it's easy to sense that charge. That's what field effect transistors do. They are affected by the field.

But changing that charge requires generating a sufficiently high voltage to create an electrostatic potential that will strongly attract or repel those electrons to break down that floating gate's insulation. That high voltage charge must be dumped before the data can be read. But it takes no time to dump the charge. But then, when immediately switching back to writing, that charge must first be built up again from scratch. And that's where all the time goes, waiting to be able to start writing after reading.

So this inexpensive thumb drive is very, very slow to switch from reading to writing. It's crazy that this first release of ValiDrive took nearly 22 minutes to validate that 32GB

thumb drive, which explains why I cannot wait to get back to work on ValiDrive to create version 2.

In order to create Beyond Recall, which will be GRC's super-secure mass storage drive wiping tool, I'm going to need to develop a bunch of technology I don't have yet. So my plan is for the second release of ValiDrive to be the development test bed for that new technology. ValiDrive 2 is doing to take a different approach to solving this problem. It's going to read and store the data from all of those 1,152 4K locations, then switch into writing mode and write them all with signature data. Then it will switch back to re-read and verify them all. Then it will switch to writing to replace all of the drive's original data, then perform one final read confirmation of the replaced data.

So this will mean two switchings from reading to writing for ValiDrive 2, whereas ValiDrive 1 is doing that 2,304 times. 2,304 times it's switching. So I suspect ValiDrive 2 is going to be much faster, more sure of its conclusions, since it will lay down signature data across the entire drive at once, and much more pleasant to use as a result. It's the thing I plan to start working on as soon as the DNS Benchmark is finished and ready.

Leo: Take a break?

Steve: Leo, yeah, let's take a break. We've got some more - we've got a bunch more really great feedback from our listeners. So now would be a good time.

Leo: I really want you to figure out why you're not getting spam. This just bothers me because if, I mean, I thought the whole purpose of your changing your email was to cast aside the previous year's email addresses.

Steve: Never comes in. The spam never catches up.

Leo: So why do you create a new email address every year?

Steve: Because I want to stay ahead of the pack.

Leo: I mean, I understand if you do that and then say, well, if you don't know this year's email address, you can't email me. But if you're accepting email to all the previous email addresses, I don't get it. I don't understand, A, why it would prevent spam; and B, why even bother? I mean, unless you believe that it prevents spam somehow.

Steve: I don't get any.

Leo: I'm really trying to figure out why that would...

Steve: So I think I probably have maybe about the last 10 years. And as I said, if I start getting spam on some prior year, and I think maybe like three or four years ago someone started spamming me, and I was unable to unsubscribe, then I just killed that one year's forwarding.

Leo: Oh, okay. So you kill addresses if you start getting spam.

Steve: If they start getting abused. But right now about eight of the past 10 years are just - they've never been discovered.

Leo: Probably, I'm going to guess, it's because you very rarely use email for anything. In other words, you're not exposing your email to people particularly. Most of the rest of the world, we use our email address all the time.

Steve: Well, it's true, I'm not in a position where my email address is being scraped. And I do, it's like my, you know...

Leo: But when you buy something, do you give them an email address?

Steve: Yeah.

Leo: Yeah. Do you give them a special email address or your regular email address?

Steve: Often my regular email address.

Leo: Well, I don't get it, then. We'll have to figure out what is Steve doing, and how can we duplicate that?

Steve: Well, as I said to my listeners, give it a try, see what happens. You may be surprised. Set up a new email account, forward the one into your new one so you don't lose anybody, and then, you know, see how long it takes.

Leo: Well, I mean, I do that. I do create new email addresses all the time. But it is very quick for them to start getting spam. But then that's probably because I use them in a variety of places that may be exposed. I don't know. It's an interesting question. If you can just bottle that, Steve, I think you have a future. You could be the new Dvorak.

Steve: Just wanted to share that no one in my company gets any spam.

Leo: Yeah, yeah.

Steve: And we don't have any filtering.

Leo: It's fascinating. All right, Mr. I Get No Spam. On we go.

Steve: Okay. So as we know, I've only studied AI briefly and enough to satisfy my desire to have some sense for what the heck is going on. So I claim no deep expertise in AI. But I have spent a great deal of time, more than our listeners know, you have some idea of it, Leo, quietly studying human brain function, and I've developed a deep appreciation for its complexity.

Over the weekend, a question was posed in GRC's Security Now! Newsgroup which I thought was very much worth asking and very much worth answering. The poster wrote: "Just wondering. If AI developments rely heavily on neural networks, and as they start to approach the human brain in capability, can they also suffer from some of the same weaknesses of the human brain? With experience, could they start to get distracting thoughts and produce more confused output? A case where adding training data might actually lead to a deterioration in performance?"

Okay. So first of all, yes. I think we already see some of that behavior which those working in the field take very seriously. But I wanted to take a moment to address some of the implications of the questioner's phrase: "If AI developments rely heavily on neural networks, and as they start to approach the human brain in capability...."

One thing our discussion of AI and neural networks never touched upon is the fact that today's current generation of AI uses structures that we call "neural networks," while at the same time we all learn in elementary school that our own human brains are filled with richly interconnected neural cells that create networks of neurons. I am 100% certain that no one listening to this podcast imagines that there's anything more than a very loose notion of a network of interconnected "somethings" that AI and our brains might have in common. But I wanted to take this opportunity created by the question to make absolutely certain that even those listeners here who may have not been following all of this very closely appreciate, without any shadow of a doubt, that the only thing an AI's so-called neural network has in common with a biological brain's neural network is the name.

The truth is that calling the addition and multiplication operations that are organized into networks of propagating values "neural networks," where the use of the term "neural" is in any way intended to suggest that any of this bears any resemblance whatsoever to the operation of biological brains, is just a joke, a total joke, really. It should almost be an embarrassment to the AI community for anything they're doing to be called "neural" in any way. You know, but it's certainly true that calling them "high-speed GPU networks," that's far less sexy.

A long time ago, when these artificial "neural" networks were laboratory curiosities, it didn't matter what they were called because they were busy learning how to win at tic-tac-toe and to play the game of NIM. But things have changed radically since that time. Neural networks have obviously moved from the lab into daily mainstream life. So to me, and I've talked to, you know, my friends and neighbors, it's a little worrisome that the neural network moniker has stuck around because it could be so misleading. And that's beginning to matter as this becomes a commonly used term. Everyone in the AI field is very clear that there is nothing whatsoever neural in the sense of a biological neuron about performing massive numbers of factor-scaling multiplications, summating additions, and thresholding.

But it's easy to see how the general public could begin to get somewhat creeped out by the idea that our brains are being emulated in some way. They're not. We do not even begin to have the capability or capacity to emulate the tiniest fraction of the complexity of a biological brain. In fact, we don't even have an accurate emulation of a single solitary biological neuron, not even one, because no two are the same, and every neuron's precise operation is unique, involving and including a hair-raising number of intrinsic and extrinsic factors.

I'd say that the only behavior shared between these artificial and biological networks is the surprisingly emergent property of their ability to self organize. They both have that. And that behavior over on the artificial side was discovered and applied more than 50 years ago. That's not new. Since then, the work has been about scaling and research to discover the best "pre-organization" to apply to these untrained artificial networks.

But anyway, you know, just I'm sure everyone's clear about this, but I just kind of wanted to dot the "I" here. You know, there's a collision of naming where both artificial networks and biological networks employ the term "neural," but that's it. There could not be anything further from the truth that anything about an artificial neural network relates to our biological brains. All they share is a name, and nothing else. So I thought it was a neat question because, you know, the idea being, oh, if our artificial neural networks start approaching our complexity, what's going to happen? Well, nobody knows how to make anything like a biological brain. And what we have today, which is surprising people, is incredibly simple by comparison.

And the fact that they both use the word "neural" is just kind of a coincidence of history, rather than anything else. Back 50 years ago it was a joke to call them neural networks. It was like, well, okay, let's call them that. Doesn't mean anything. Doesn't mean they're like human neurons at all, biological neurons.

Lyle Haylett said: "I've been a listener of 1009 Security Now! podcasts, so obviously highly appreciate the work you and Leo do to bring it to us listeners. I felt the need to comment on the DJI Geofencing 'unlocking' issue. I am an FAA certified Part 107 commercial remote pilot, a drone operator, as well as a certified Private and Instrument Rated pilot." Okay, so he flies both drones and planes.

He says: "I utilize two DJI drones and a home-built drone to do commercial 3D mapping, photography, and videography for the construction, real estate, and other businesses." I imagine maybe wedding photography. He says: "Drones that are considered 'enterprise' or 'commercial,' as well as lower-priced drones that are considered 'consumer' or 'recreational,' can and are routinely used for these business purposes." I just, I love our listeners. This is so great. Here's somebody who's right in the middle of all this. Thank you, Lyle.

He continues: "I wanted to clarify that, to my knowledge, no other drone manufacturers have ever limited where a drone can fly. Any other drone could fly over any of those restricted areas you mention, subject only to the will of the operator. The DJI restricted zones were never well-aligned with where someone could legally fly a drone in the United States. In many cases, their restrictions applied to areas where it's perfectly legal and safe to fly." He says: "And I believe in some cases they even permitted flying in areas where it is not legal to fly.

"This has been a frustration for pilots like me since I can get FAA (LAANC) authorization to fly (almost instantly), only to find, when going to the site of a job, that there was some DJI Geo Zone that needed to be unlocked. If Internet access was not available, I would be unable to fly. In addition, I had instances where the Geo Zone kicked in after taking off, limiting my control..."

Leo: That's not good.

Steve: Oh, boy, "of the drone." He said: "GPS isn't perfect and can sometimes be widely inaccurate. Combine that with a function that takes control of, or limits, manual control of the drone, that creates a hazard. Moreover," he said, "my biggest concern with the old DJI Geo Zones is that many, particularly recreational flyers, believe that if they are okay

according to DJI Geo Zones, then they're safe and legal to fly, when oftentimes they are not. In many of these areas they would need to get FAA LAANC approval to be legal and safe to fly, and they simply don't know. Now, since DJI has aligned their warning zones with the FAA areas that need approval, at least pilots will be properly warned to make sure they're legal and safe to fly. I think that, on balance, the new system is better for everyone, particularly since no other drone manufacturer to my knowledge has ever been doing anything like this.

"I'm an avid proponent of safe drone flying and probably somewhat obnoxious to people recreationally flying drones when I try to educate them on what they should and should not be doing. I don't know if you have a drone, but I do know that Leo has one. So as part of my drone safety soapbox, I hope he (or you, if you have a drone) have taken the FAA 'TRUST' test and are legal. Sincerely, Lyle from Tennessee."

Leo: Hmm. I'd better get going.

Steve: So Lyle, thank you so much. It is so valuable to receive feedback from someone who has a broader perspective and experience with the subject. It seems very clear that DJI was really not giving anyone "the middle finger," as some in the press and on the Internet suggested, and that they were aligning with the rest of the industry and, hopefully, making drone operators more responsible by aligning their warning zones with the FAA's guidelines. So, you know, thank you for bringing us a reality check.

Leo: I was just ignorant. I had no idea, yeah.

Steve: Huh? Yeah.

Leo: I was just ignorant.

Steve: Most people, you know? Unless we know from somebody who's got experience and doesn't have their own cross to bear. I hope it doesn't needlessly harm DJI. As we know, they're the best drones, and we would like to still have access to them.

Leo: Yeah.

Steve: Tim Clevenger said: "Hi, Steve. I heard you talking about the Sponsors page on TWiT's website. Club members can also find the links to the current show's advertisers in the episode's description in their podcatcher. Thank you for the show. It helped me to not only ace the interview when I moved from IT into Cybersecurity a few years ago; it also helped me pass my CISSP certification exam last April. Tim." So, Tim, thank you. And I wanted to share that news with anybody else who is looking for where to find the sponsors. We talked about this last week, that it's on the TWiT.tv website in the upper right of the menu.

Leo: Yes.

Steve: And finally, George Adamopoulos said: "Dear Steve, I'm a Security Now! subscriber for several years. Thank you for all the hard work. I have a remark about the forced Outlook update that you talked about in Security Now! #1009." So that was last week. "As Leo mentioned, Windows already had an email client, Windows Mail. What you did not mention is that this is being deprecated in favor of this new Outlook. In fact, when I tried to open Mail just now, I got a warning that 'Support for Windows Mail, Calendar, and People will end on December 31st, 2024.'" He says: "(Yes, that's in the past)." He said: "Next to it is a button to 'Try the new Outlook.' Even if I press nothing, a few seconds later, the new Outlook opens automatically. To add insult to injury, the new Outlook displays ads. Anyway, thank you once again for the excellent work that you do. Kind regards, George Adamopoulos."

Well, there's not much more I can add to that other than to say "Thank goodness for eM Client." I have no idea whether eM Client would work for the enterprise, but it looks like it checks a lot of the boxes. They bill it as "all-compatibility tool support" and so forth we talked about last week. Google Workspace, Outlook 365, Office, Exchange and all that. So anyway, you know, I appreciate that. And Leo, this is what Microsoft's doing now; right? I mean, we've talked about Edge and the enshittification of all of this.

Leo: Yeah.

Steve: And so here's New Outlook.

Leo: It's not the first time, even. Remember Outlook Express? Then they changed it and turned it into Live Mail.

Steve: Right.

Leo: And then they - I think there have been a couple others since then. They just kind of do this on a regular basis. I guess it makes sense. If you have a lot of technical debt built up in a mail client, maybe sometimes it's good to start over.

Steve: Yeah. Yeah. So DNS Over TLS. I wanted to share my experiences thus far with the implementation of GRC's DNS Benchmark which, as we all know, I'm in the process of updating to support IPv6 and the various encrypted DNS protocols that are increasingly being used to protect the privacy of users' web accesses. And I think everybody's going to find this interesting and a little surprising. What I discovered was initially surprising to me until I sat back and thought about it a bit. And I believe at least for intellectual curiosity's sake it'll be of use to our listeners.

As I've mentioned before, GRC's original DNS Benchmark, which I first wrote 16 years ago, provided a complete solution at the time for determining the performance of the DNS servers that everyone could choose to use. But as we know, times change. That first release was strictly IPv4, and there was no notion of encrypting DNS for privacy. All of that has changed during the intervening 16 years. IPv6 is slowly but steadily coming online with all recent operating systems, most ISPs now, and the intervening equipment such as consumer routers now supporting IPv6. So it's on the desktop.

During the past 16 years we've also witnessed a massive transformation in the monetization of the Internet's users. Who we are, who and what interests us, and where we go is all up for sale. That information is being used to generate additional revenue for

everyone at every stage of the pipeline, from the websites we visit and the advertisers to our ISPs who connect us to the Internet. Since many who use the Internet would prefer to do so with as much privacy as possible, the ability to encrypt DNS queries, which otherwise advertise our every whim and desire, is of growing interest. In response to this growing interest, all of the major public DNS providers such as Google, Quad9, Cloudflare, and many others already offer fully

encrypted DNS services. Our routers and web browsers offer support, and it's already built into Windows 11. So it's easy to have.

To the best of my knowledge, no one has ever answered the question of how much DNS query performance is sacrificed to obtain the privacy offered by encryption. How do encrypted DNS lookups using encrypted TLS or HTTPS connections compare to traditional in-the-clear DNS over UDP? And even if this weren't a concern, I could hardly offer an updated DNS Benchmark today that didn't also benchmark IPv6, DoT, and DoH in addition to traditional IPv4.

As I mentioned before when Leo and I were talking about the work I've been doing recently, the first major change was restructuring the entire DNS Benchmark to use any protocols other than IPv4. Since IPv4 addresses are all 32-bits long, and since the DNS Benchmark was written for Windows Win32 API, 16 years ago I took advantage of the ability to hold any DNS nameserver's IP in a native machine 32-bit register. The switch to IPv6's 128-bit addresses, not to mention DoT and DoH nameservers which are addressed by URLs just like web pages, meant that needed to change. 32-bits no more. Today's DNS Benchmark is now, as a consequence of the updating work I've done so far, completely protocol agnostic. Any protocol can be added to its underlying structure, which has largely been rewritten. So it's now ready to handle today's newer DNS protocols and whatever else the future might hold going forward.

After the Benchmark's fundamental redesign, the first thing I did was to add support for IPv6 nameservers since that was just a matter of adding more nameserver address bits, making room for longer IP addresses in the user interface and teaching the Benchmark about the funky zeroes compression that's used to shorten the many IPv6 addresses that contain one or more words of all zeroes.

Then it was on to TLS, and things suddenly became quite a bit more interesting. Windows has an API known as Secure Channel, or "Schannel" for short. Using the API takes some getting used to, since it was designed to provide an interface, sort of a generic interface to a large collection of very different underlying secure protocols, of which TLS (Transport Layer Security) is only one. So this requires the user to do weird things like repeatedly call into the API until we're told that its needs have been satisfied, whatever they may be. It's all deliberately opaque. So as a coder you just have to sort of shrug and say "okay," follow the weird rules, and hope for the best.

However, no one explained the API to me like that. In fact, the entire thing is woefully under-documented. So I spent some time staring at what few examples I could find online, wondering whether what I was seeing could possibly be correct since, as I said, it's really quite weird. I've been documenting my journey through all of this in GRC's public newsgroups, and I'm currently at the fifth generation of this TLS support system. The code that I finally have is actually quite lovely, and I'm proud of it. It's far more clear and clean than anything I've found online.

And someday, after I've pulled the plug on GRC, and I release all of the source code of my work, which is my eventual plan, I'll be glad to have contributed to cleaning up the mess that Microsoft created with this weird Schannel API. And I will make a point of inviting the world's AI's over to dig around in that source code so that they might be able to help others quickly get to where I wound up. So my point is, I have TLS working

beautifully now. But that's where some real surprises, that Microsoft had nothing to do with, were encountered.

When GRC's DNS Benchmark is started, when you start the program, fire it up, it loads the list of DNS nameservers it will be testing. For every nameserver, it sends a couple of test DNS queries to verify that the nameserver is online and reachable from the user's current location and connection. It also uses the system's standard DNS nameservers, whatever nameservers are configured on the Windows desktop, to query a couple of public databases to obtain the ownership information about the IP address space housing the nameserver to create a richer experience and provide more background information about all these IP addresses, you know, who owns them, because it's not otherwise clear from an IP address. The URLs, which the encrypted name servers use, does tell a much richer story.

So here's where we first encounter the biggest difference between traditional DNS and any form of encrypted DNS. Traditional DNS is carried over the UDP protocol. UDP stands for User Datagram Protocol. When a user's computer wishes to look up the IP address of a domain name, that domain name is packaged into a single Internet UDP packet, and it's sent to whatever DNS nameserver the user's computer has been configured to use. And that's it. Package the domain name into a packet and send it out onto the Internet with the destination IP of one of the user's configured nameservers. Hopefully, the packet arrives at its destination. When it does, the nameserver examines it, takes whatever actions may be needed to obtain the IP address that's been requested, and eventually replies by appending the answering IP to the user's DNS query, which also fits into a single packet.

The original DNS protocol designers understood the value of keeping everything tucked into single packets. So DNS doesn't miss a trick when it comes to quick hacks to eliminate any redundancies in DNS queries and their replies. If the sender of the query doesn't receive a reply within a reasonable length of time, either the query or the reply packets may have been dropped by a router along the way. They'll simply ask all of the nameservers they've been configured for and accept the first reply they receive. They just try again. But typically on a retry they ask everybody.

What we have as a result is a truly elegant and minimal system. One Internet DNS query packet goes out, finds its way across the Internet, and is received by the user's designated DNS nameserver. That nameserver makes it its mission to get the answer to the user's DNS query. And once it has it, you know, it might just be, as I talked about earlier, it's got, you know, Amazon.com. Got the IP right there in its cache. It just immediately sends the answer back. Either way, once it has the answer, it sends the reply back in another single packet. It's beautiful. Yes, it is.

Unfortunately, what it also is, is ruthlessly hostile to encryption. It offers no privacy. Now, we know what encryption requires. At the bare minimum, encryption requires that the entities at each end of any connection share a secret that no one else can possibly know. They then use that shared secret to encrypt and decrypt the messages they send back and forth. So how do they obtain that secret? We know that there are key exchange mechanisms that make establishing a shared secret in full view of the public possible. But they're vulnerable to man-in-the-middle attacks. And we know that the only way to prevent a man-in-the-middle attack is to be able to positively authenticate the identity of the party we're connecting to.

The way that's done, using the technology we currently have, requires a certificate. And certificates are large, like between 3 and 6K. What this all means is that just asking for a tiny little bit of privacy here for our DNS queries and their replies completely blows all of the original elegance of DNS's fast and lightweight single-packet queries and replies out of the water. All we want is for a single packet not to be eavesdropped on. But the

realities of the Internet means that in order to do that we have no choice other than to drag all of the massive overhead of connection security along for the ride.

The other thing I didn't explicitly mention is that, with all of this back and forth exchange of certificates and handshaking and encryption protocol enumerations and agreements, on top of all of that we cannot just have packets getting lost along the way. So the only way to carry on this dialog, which has suddenly become much more complicated, is by moving from the minimal elegance of single-packet UDP, the User Datagram Protocol, to the reliable delivery system provided by TCP, the Transmission Control Protocol. So that's what I built. That's TLS on top of TCP. For every remote nameserver that the DNS Benchmark will be testing, it looks up the IP address for that nameserver's domain name because, again, remember, encrypted nameservers are referred to by domain names, just like web pages. They've got URLs.

So we look up the IP address of the nameserver's domain name. Whereas the original standard port for DNS is port 53, the standard port for TLS encrypted DNS is 853. So the Benchmark establishes a TCP connection to the remote nameserver's port 853. It then initiates a TLS connection negotiation, negotiating encryption protocols, receiving and verifying the remote nameserver's certificate because that's part of TLS, agreeing upon a shared secret key, and then bringing up the encrypted tunnel. That's that whole weirdly opaque Schannel API stuff that I spoke about earlier. Okay. At this point - whew, yay! - we have a connection to a remote DNS nameserver over TLS which should allow us to send and receive DNS queries.

So it was with great joy and celebration that I got all of that working, whereupon the remote nameservers began unceremoniously disconnecting and dropping their connections without warning or reason and with prejudice. I thought, what? I tried it a few times, and the same thing kept happening. It seemed that these nameservers were, I don't know, impatient for queries. And they were not being uniformly impatient. Some would drop the connection after a second. Some would wait five seconds, or in between. But without fail, the connections would be dropped. So I figured that perhaps they were getting annoyed with me for getting them on the line and not immediately asking them for some DNS resolutions.

So I started having the Benchmark send them DNS queries to answer over this newly created connection. This maybe worked a little better. Things were definitely working. The connection was up, and TLS was running. I was able to use Wireshark to observe the transactions, the packets moving back and forth across the wire. And I was receiving valid answers to the Benchmark's queries. So we were on the right track. But without warning, even in the midst of DNS queries and replies, the remote ends were still getting fed up with my questions and dropping connections.

After sitting back and thinking about this for a few minutes, the reason for this all became obvious. Compared to unencrypted UDP queries and replies, TCP - and especially TLS over TCP connections - are incredibly expensive, not only to establish, but to maintain. Traditional UDP DNS nameservers have been so spoiled compared to almost all other servers. They receive a UDP query packet to which they reply with an answering UDP reply packet. And that's it. Period. Mission accomplished.

Thank you very much.

We've talked about all of the back and forth that's required to establish a TCP connection, and then even more for TLS once the TCP connection is established. But there's another significant cost to maintaining a connection. Both TCP and TLS require each end to maintain a great deal of "state" information. Since TCP numbers every byte that's sent and received, it's responsible for providing reliable delivery of anything sent and acknowledging the receipt of everything received. It needs record-keeping to make all of

that happen. And that also means that the TCP/IP stack needs to be aware of the existence of all of the many various connections to everywhere so that the incoming and outgoing packets can all be routed appropriately.

And once the packets pass through the TCP/IP layer, the TLS protocol has a bunch more of its own "state." It needs to retain the knowledge of the specific TLS encryption protocol and the version that was negotiated with the end, and the shared secret key for encrypting and decrypting the data, and the state of all the many options that have been added to TLS from the start of SSL up through TLS 1.3. In other words, a lot. And now consider all that in comparison to plain old standard DNS queries over UDP, which has none of that. None. A packet arrives, and a reply is returned. DNS over UDP has no state. Nothing to remember between queries. No state to preserve. No connections. Nothing.

Okay. So now we switch back to those big iron DNS servers that are being operated by Quad9, Google, Cloudflare, and many others. Think of how many thousands or tens of thousands of clients' queries they may be handling every second of every single day. For UDP, that's no problem. Packet in, packet out. They just do it. Done. They reply to every query and forget about it. But for DNS queries that need to establish a TCP connection, then negotiate a TLS secure tunnel on top of that - all before even the first DNS transaction - that's one heck of a lot of overhead. And now imagine, with this expensive connection established, the client expects this busy, widely shared public nameserver to just sit there, with a TCP connection established and TLS crypto negotiated, and wait for the client to ask a question. Not happening. There's no way busy and super-popular nameservers can possibly afford that.

They cannot afford to tie up their precious RAM memory with all of the state tables and flags and options that every single one of these connections requires, only to have the client not immediately needing and using its services. So it should come as no surprise that these nameservers are exhibiting very little patience with inactive connections, and that even with active connections, they're only able to give anyone who asks a limited amount of their time.

Given all of this, you might be inclined to wonder why all of this works at all. How can encrypted DNS, which is so much more expensive than good old DNS over UDP, be the future? The answer is that web browsers' use of DNS is inherently bursty. When a user clicks a link to jump to a new web page that it's never visited before, and assuming that the browser or the operating system is configured to use DNS over TLS or DNS over HTTPS, a connection will be brought up to the remote nameserver to obtain the IP address of the site. Once the IP address is obtained, the browser will immediately connect to that remote web server to obtain the destination web page.

Today, in 2025, fully populating a typical web page requires the resolution of an average of between 50 and 150 DNS domain names. Those are the domains for the advertisements, the script libraries, the images, the various tracking gizmos, and all of the other goop that runs today's web. So upon downloading and obtaining the destination webpage, the user's web browser, which would very likely still be holding open the connection to the remote nameserver, will send off a blizzard of those 50 to 150 DNS queries over the previously negotiated secure and encrypted TLS tunnel. And that will pretty much be it for a while. The user's web browser will have collected all of the IP address responses it needs to fetch all of the rest of the page's contents. So if either it or the far end decides to drop the expensive-to-maintain TCP/TLS connection, who cares?

This is what I meant when I said that DNS queries are inherently bursty. They generally arrive in a very brief flood with the display of a new page, which the browser then renders, and the user examines and ponders, before eventually clicking another link, which generates another brief flurry of queries. And so it goes. This means that bringing

up a relatively short-lived, and very expensive to maintain, TCP/TLS connection winds up being cost effective.

It's true that doing all of this connecting, establishing, and negotiating takes time and multiple - many - packet roundtrips. But once it's been done, the DNS queries and replies are able to occur with the same speed as regular DNS, even though they're now encrypted with the same state-of-the-art crypto protocols we use to protect all of our other crown jewels. And if 50 to 150 queries are being sent in a burst, the time required to set up the connection can be amortized across all of the DNS work that can get done once the connection is ready. The user will not experience any different page-loading performance than before.

Also, the TLS protocols offer session resumption features where the answering remote server bundles up all of its post-negotiation TLS state information, encrypts it under its own local secret key, and hands it back to the client to keep at the end of their initial connection negotiation. This allows the client to cache that opaque blob which it's then able to return and offer to the server the next time it reconnects to that same server. The server receives the blob, decrypts it using its own private key which no one else has. And if everything matches up, the client and the server are able to bypass all of the time-consuming and expensive TLS renegotiation to pick up right where they left off.

Having thus understood what's going on with nameservers, GRC's benchmark is now working with every one of them I have found. I've got a long list. And since DNS over HTTPS just wraps the DNS query and its response inside HTTP protocol which also runs inside TLS, I expect to have that added and running shortly. And now everyone has a much better sense for how the industry is moving forward to encrypt the last of the simple plaintext protocols which has survived until now. I imagine that DNS over UDP will someday go the way of good old unencrypted HTTP, which we hardly use any longer.