Security Now! #1010 - 01-28-25

DNS over TLS

This week on Security Now!

eM Client CAN be purchased outright. An astonishing 5-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 6-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.

What do you mean you: *"Forgot to pack our Australia/New Zealand plug adapter?"*

Errata

A Permanent license for eM Client!

Since there was no way for me to practically individually thank all of our many listeners who took the time to say "Uh, Steve... Your one big gripe about eM Client, which you recently fell in love with, is not actually a thing." I wanted to say thank you to one and all. I have no idea how I missed the very clearly marked slider near the top of eM Client's pricing page, but I certainly did.And now that I've seen it, it's not possible to "unsee" it. So I am now the proud owner of a lifetime license with updates forever.

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. Later, XM merged with Sirius and somewhere along the way the option to purchase a lifetime subscription was eliminated. But I still have mine and I'm very glad that I made that choice many years ago. And back when Tivo's were the way to go, Leo and I would both always purchase the lifetime subscriptions for our Tivo's rather than paying month by month. Since I tend to stick with things until I'm forced to switch, that approach has always worked well for me.

So, just to follow-up on my raves about eM Client last week, I am even more pleased with my switch to eM Client than ever. I heard from many of our listeners who asked "What took you so long?" – these people had discovered eM Client years ago and similarly love it. So, in addition to thanking everyone and who wrote to make sure 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 feel the same way about it that I do. In my opinion, they really got the user-experience right.

Leo perked up upon hearing that it fully supports end-to-end encrypted GnuPG email and address books. So there's that there, too.

And 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 if they, too, were feeling frustrated with their current solution, whatever it might be. And that was also a success. I'll share one quick piece of feedback that's representative of the many I received:

Dan Taylor

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 that obvious on the pricing page, but it's there. I had NO previous knowledge of its existence. In a nutshell, it's wonderful! I have only one Gmail account. I also own 2 domains via Cloudflare, which forwards all email destined for those domains, to my Gmail account. I've configured some aliases (one of which, I'm using to send this to you). It's very cool!

Also, I know you know this. But, you have 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!

Dan's need for only a single gmail domain with others forwarding into it suggests that he may be able to use eM Client's free single-account offering.

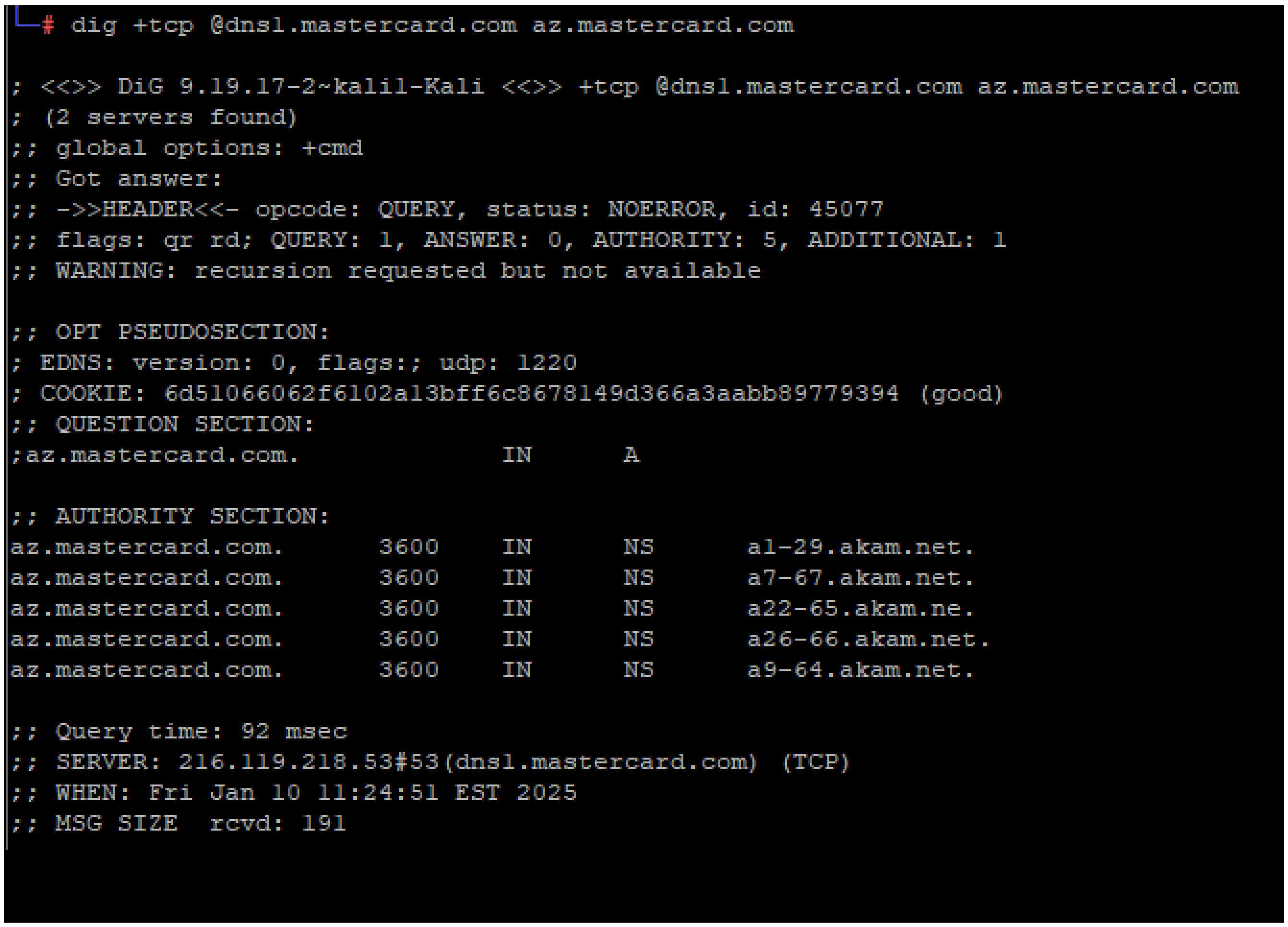
Security News

An astonishing DNS typo

This week's first piece of security news is, as they would say in the UK, "gobsmacking". Our friend, Brian Krebs over at KerbsOnSecurity shared a wonderfully surprising piece of news last Wednesday under the headline "MasterCard DNS Error Went Unnoticed for Years". Before we go any further 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 for years. 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 until a security researcher spent $300 to register the domain and prevent it from being grabbed by cybercriminals.

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:



```
└─# dig +tcp @dns1.mastercard.com az.mastercard.com

; <<>> DiG 9.19.17-2~kali1-Kali <<>> +tcp @dns1.mastercard.com az.mastercard.com
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45077
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 5, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1220
; COOKIE: 6d51066062f6102a13bff6c8678149d366a3aabb89779394 (good)
;; QUESTION SECTION:
;az.mastercard.com.             IN      A

;; AUTHORITY SECTION:
az.mastercard.com.      3600    IN      NS      a1-29.akam.net.
az.mastercard.com.      3600    IN      NS      a7-67.akam.net.
az.mastercard.com.      3600    IN      NS      a22-65.akam.ne.
az.mastercard.com.      3600    IN      NS      a26-66.akam.net.
az.mastercard.com.      3600    IN      NS      a9-64.akam.net.

;; Query time: 92 msec
;; SERVER: 216.119.218.53#53(dns1.mastercard.com) (TCP)
;; WHEN: Fri Jan 10 11:24:51 EST 2025
;; MSG SIZE  rcvd: 191
```

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.

Brian explains:

From June 30, 2020 until January 14, 2025, one of the core Internet servers that MasterCard uses to direct traffic for portions of the mastercard.com network was misnamed.

MasterCard.com 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, edited or updated the DNS record for that mastercard.com domain on June 30th, 2020, which lists the five authoritative DNS nameservers that should be referred to when looking up any IP addresses for mastercard.com, made a tiny and earthshaking mistake – 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.akam.net" and the fifth one is "a9-64.akam.net". But one in the middle of the 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:

This tiny but potentially critical typo was discovered recently by Philippe Caturegli (cat-ur-reg'-li) , 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.

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. 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.

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 (SSL/TLS certs) 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 [Brian Krebs] on his notifications. A few hours later, 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. 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 their systems.

Brian continues:

Meanwhile, Caturegli (cat-ur-reg'-li) 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.

In other words, you know, responsible disclosures and bug bounties.

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 post removed.

Caturegli said while he does have an account on Bugcrowd, he has never submitted anything through the Bugcrowd program, and that he reported this 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."

Most organizations have at least two authoritative domain name servers [That's what I do for GRC. A pair of DNS nameservers are commonly used for redundancy and reliability.]*, but some handle so many DNS requests that they need to spread the load over additional DNS server domains. 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 domains they would only be able to see about one-fifth of the overall DNS requests coming in.*

But Caturegli (cat-ur-reg'-li) explained that the reality is many Internet users are relying at least to some degree on public traffic forwarders or DNS resolvers like Cloudflare and Google.

I would strengthen that statement a LOT to say that there is no one who is **not** relying upon caching resolvers. As we've often discussed on the podcast, caching DNS is crucial. It's the only way a hierarchical system of distributed domain name resolution is able to function.

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

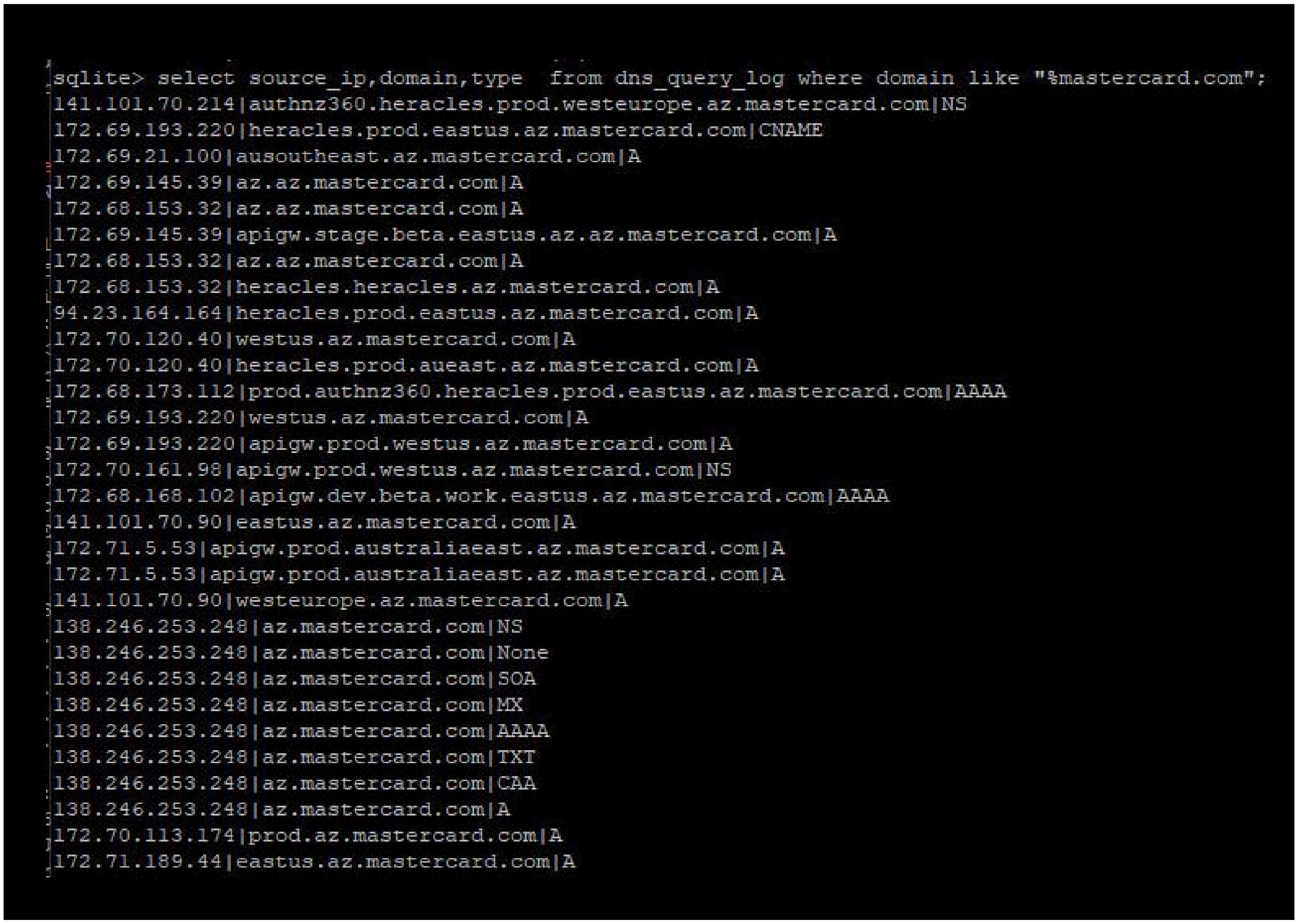
He said: "With a long TTL, we may reroute a LOT more than just 1/5 of the traffic."

And that's absolutely true. Typical TTL's are an hour or two. Sometimes longer. But since the TTL field is a 32-bit unsigned integer, a caching period up to 136 years can be specified. It's unclear whether such a long TTL would be honored, and a nameserver reboot would be likely to flush any cache. But many months of caching could be requested. That would mean that no one who received such a long cache life from a malicious nameserver would come back soon to obtain any connection.

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 shows a who's who of DNS queries:



```
sqlite> select source_ip,domain,type  from dns_query_log where domain like "%mastercard.com";
141.101.70.214|authnz360.heracles.prod.westeurope.az.mastercard.com|NS
172.69.193.220|heracles.prod.eastus.az.mastercard.com|CNAME
172.69.21.100|ausoutheast.az.mastercard.com|A
172.69.145.39|az.az.mastercard.com|A
172.68.153.32|az.az.mastercard.com|A
172.69.145.39|apigw.stage.beta.eastus.az.az.mastercard.com|A
172.68.153.32|az.az.mastercard.com|A
172.68.153.32|heracles.heracles.az.mastercard.com|A
94.23.164.164|heracles.prod.eastus.az.mastercard.com|A
172.70.120.40|westus.az.mastercard.com|A
172.70.120.40|heracles.prod.aueast.az.mastercard.com|A
172.68.173.112|prod.authnz360.heracles.prod.eastus.az.mastercard.com|AAAA
172.69.193.220|westus.az.mastercard.com|A
172.69.193.220|apigw.prod.westus.az.mastercard.com|A
172.70.161.98|apigw.prod.westus.az.mastercard.com|NS
172.68.168.102|apigw.dev.beta.work.eastus.az.mastercard.com|AAAA
141.101.70.90|eastus.az.mastercard.com|A
172.71.5.53|apigw.prod.australiaeast.az.mastercard.com|A
172.71.5.53|apigw.prod.australiaeast.az.mastercard.com|A
141.101.70.90|westeurope.az.mastercard.com|A
138.246.253.248|az.mastercard.com|NS
138.246.253.248|az.mastercard.com|None
138.246.253.248|az.mastercard.com|SOA
138.246.253.248|az.mastercard.com|MX
138.246.253.248|az.mastercard.com|AAAA
138.246.253.248|az.mastercard.com|TXT
138.246.253.248|az.mastercard.com|CAA
138.246.253.248|az.mastercard.com|A
172.70.113.174|prod.az.mastercard.com|A
172.71.189.44|eastus.az.mastercard.com|A
```

West Europe, East US, West US, 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 malicious Russian or Chinese attackers, who have repeatedly demonstrated that they're looking for any advantage they can find over the West, the story we'd be reporting would have had a very different ending.

That said, mistakes happen and anyone can make an innocent mistake. I'm sure this was one. 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 integrity within his own community.

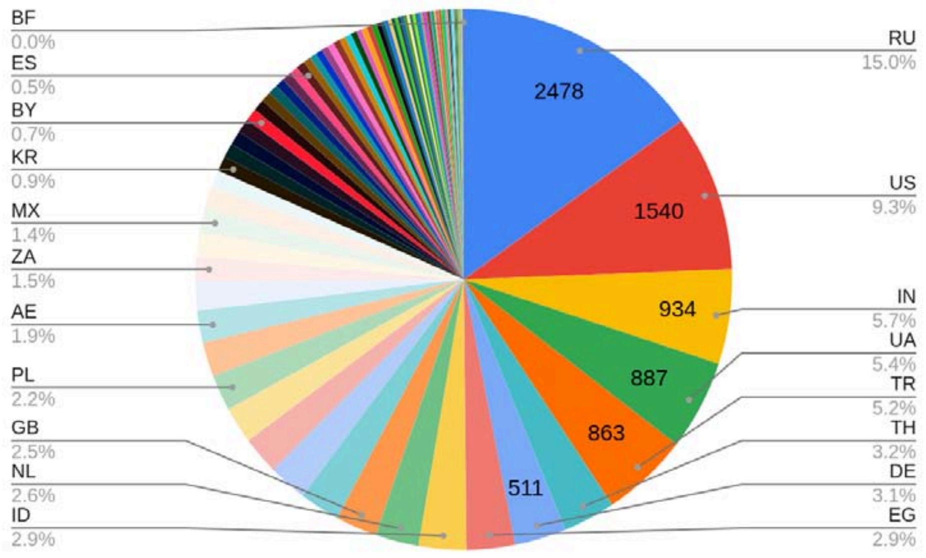
https://krebsonsecurity.com/2025/01/mastercard-dns-error-went-unnoticed-for-years/

18,459 low-level wannabe hackers get some of their own medicine
Last Friday, the security firm CloudSEK (spelled SEK) 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:

A Trojanized version of the XWorm RAT build – where R.A.T. is the common abbreviation for Remote Access Trojan – has been weaponized and propagated. It is targeted specifically towards script kiddies who are new to cybersecurity and directly download and use tools mentioned in various tutorials, thus showing that there is no honour among thieves.

Not that anyone ever thought there was any. Rather than going with *"no honor among thieves"* I might have chosen: *"There's no such thing as a free lunch."*

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 over 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, registry modifications, and a wide array of commands enabling full control over infected systems. Top victim countries include Russia, USA, India, Ukraine, and Turkey.



The malware uses Telegram as its command-and-control (C&C) 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 1 GB of browser credentials from multiple devices.

So the wannabe hackers really are being hacked. Browser credential theft allows the actual bad guys behind this to impersonate them on any websites where they were logged on.

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 using aliases like "@shinyenigma" and "@milleniumrat" as well as GitHub accounts and a ProtonMail address.

The rise of sophisticated Remote Access Trojans (RATs) 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, Telegram channels, and forums. This was specifically targeted towards script kiddies who are new to cybersecurity and use tools mentioned in various tutorials. This builder provides attackers with a streamlined tool to deploy and operate a highly capable RAT, which features advanced capabilities like system reconnaissance, data exfiltration, and command execution.

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, we uncovered the infection sources, mapped its command-and-control (C&C) 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.

The malware that these 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:

- /browsers – Steal saved passwords, cookies, and autofill data from web browsers
- /keylogger – Record everything the victim types on their computer
- /desktop – Capture the victim's active screen
- /encrypt*<password> - Encrypt all files on the system using a provided password
- /processkill*<process> - Terminate specific running processes, including security software
- /upload*<file> - Exfiltrate specific files from the infected system

... and 50 additional commands.

What struck me is that there is such a large and thriving ecosystem of low-level hackers who apparently aspire to be running their own botnets. 18,459 known instances, where this Trojan Trojan was downloaded, installed and run.

2478 of them are located in Russia. But the US is the runner up with 1540 installed instances. I suppose when you consider the size of the world and the number of kids who are probably enamored of the idea of being 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 low-hanging fruit. They think they're getting something for nothing ...
... and boy, are they!

Speaking of BotNets generating widespread attacks – we have a new record

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.

Today's DDoS attacks appear to be broken just for the sake of breaking them. By that I mean that hitting anyone with **5.6 trillion bits of attack traffic per second** is massive overkill.

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 such as Closeflare. And, in fact, that's what happened. During the week of Halloween, at the end of October, 2024, Cloudflare's DDoS defense systems successfully and autonomously detected and blocked a 5.6 Terabit per second (Tbps) 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 happened. That's incredible.

In their report, which I've linked to in the show notes for anyone who is interested, (https://blog.cloudflare.com/ddos-threat-report-for-2024-q4/) they note that:

- In 2024, Cloudflare's autonomous DDoS defense systems blocked around 21.3 million DDoS attacks, representing a 53% increase compared to 2023. On average, in 2024, Cloudflare blocked 4,870 DDoS attacks every hour.

Hold ... 4870 DDoS attacks per hour. And that's not all of the Internet. That's only the attacks against Cloudflare, its infrastructure or 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. 4,870 attacks per hour or 21.3 million DDoS attacks – just for Cloudflare.

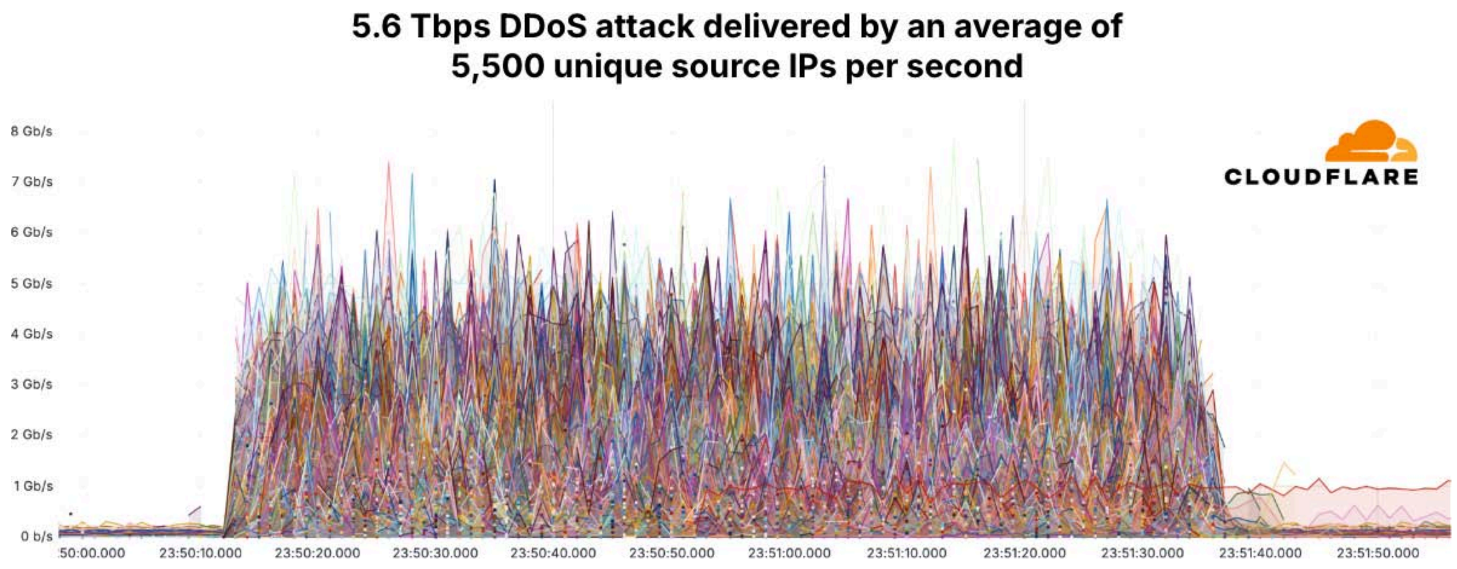
- In the fourth quarter, over 420 of those attacks were hyper-volumetric, exceeding rates of 1 billion packets per second (pps) and 1 Tbps. And, the number of attacks exceeding 1 Tbps grew by a staggering 1,885% quarter-over-quarter.

About this record-breaking attack, Cloudflare write:

On October 29, a 5.6 Tbps UDP DDoS attack launched by a Mirai-variant botnet targeted a Cloudflare Magic Transit customer, an Internet service provider (ISP) 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.

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 shown (below on the next page), each line represents one of the 13,000 different source IP addresses, and as portrayed, each contributed less than 8 Gbps per second. The average contribution of each IP address per second was around 1 Gbps (~0.012% of 5.6 Tbps).



Cloudflare's 20th quarterly update has much more, which I recommend to anyone who's curious: https://blog.cloudflare.com/ddos-threat-report-for-2024-q4/

A Clarification from Let's Encrypt
Twelve days ago, on January 16th, Let's Encrypt posted their formal announcement of their plans for 2025 – and a sincere "Thank you" to one of our listeners for pointing me to this. The opening paragraph 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.*

So, I'm 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 does expose its users to a 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; which suggests that something must be driving this. There must be those who are 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.

The Let's Encrypt statement included a timeline:

We expect to issue the first valid short-lived certificates to ourselves in February of this year. 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.

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).

SpinRite

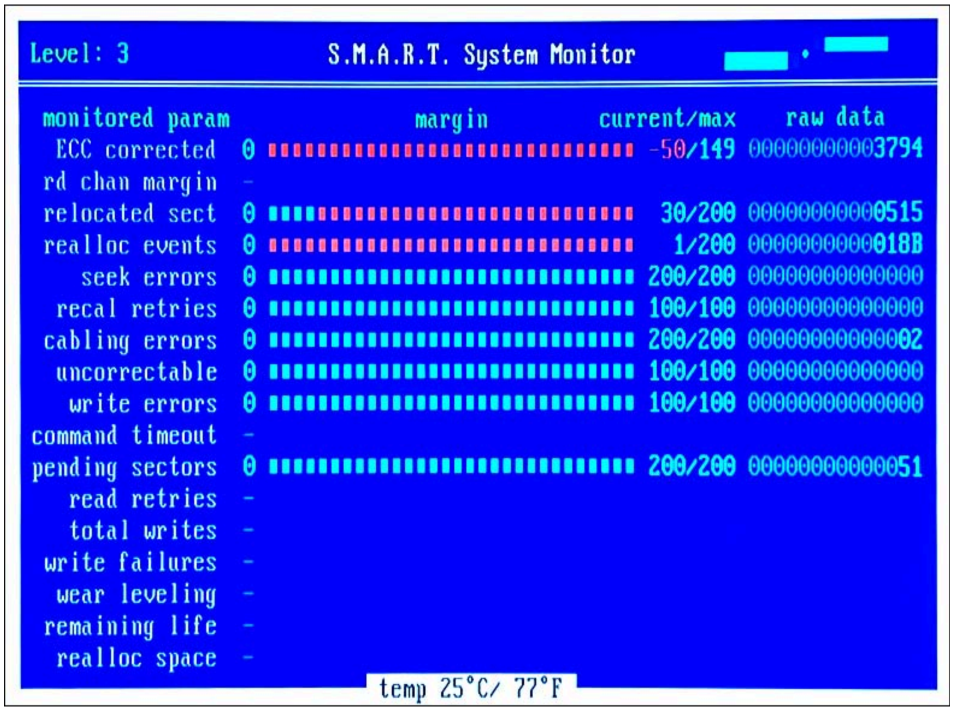
I haven't mentioned SpinRite for quite a while, since I haven't had anything new to share. We all know of the discovery that the fronts of SSDs, where the operating system files live, slow way down after years of use, and that a single Level 3 SpinRite pass will restore the drive's original performance. I receive ongoing reports of that and I've posted some on SpinRite's pages. But that 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

Hi Greg, I bought four Western Digital Red Plus 8-Terabyte hard drives for a Zima Cube and wanted to check their operation before installing. The first 2 passed spinrite level 3 in about 28 hours each with no errors. The third got 80 percent through, but then started showing problems through the SMART screen. By 94%, (which took 106 hours) there were 216 bad sectors, 379 minor issues and 6845 command timeouts with the status screen showing four "B's" (bad sectors). I am running the 4th WD 8 Terabyte drive on a Zima board. Like the first two drives, it is having no trouble at 68% and should finish before the bad 3rd drive.

Questions:
1. Would you return this third drive showing the problems?
2. What do Command Timeouts mean?
3. How do I know how many spare sectors remain for future swapping out?



```
Level: 3                 S.M.A.R.T. System Monitor

 monitored param               margin          current/max    raw data
   ECC corrected  0                              -50/149  00000000003794
  rd chan margin  -
  relocated sect  0                               30/200  00000000000515
  realloc events  0                                1/200  0000000000018B
     seek errors  0                              200/200  00000000000000
    recal retries 0                              100/100  00000000000000
  cabling errors  0                              200/200  00000000000002
   uncorrectable  0                              100/100  00000000000000
    write errors  0                              100/100  00000000000000
 command timeout  -
 pending sectors  0                              200/200  00000000000051
    read retries  -
    total writes  -
  write failures  -
    wear leveling -
  remaining life  -
   realloc space  -
                           temp 25°C/ 77°F
```

This SpinRite user included a screenshot of what this one drive was showing him about itself. What we see here is a brand new drive that's in serious trouble. The whole SMART system – Self Monitoring Analysis and Reporting Technology – has always been a mixed blessing, because it's never been a strong standard. It's an extremely weak standard. In fact, it's really not 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 and are maintained by volunteers to show what this or that specific drive make and model “means” with this or that SMART parameter.

But 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 – UP is good, DOWN is bad. So the screen above tells an unambiguous story. It shows us that the drive ITSELF is saying that three clearly crucial drive parameters: the amount of ECC error correction being needed, the rate of bad sector relocation, and the number of relocation events, are reflecting a drive that is in serious trouble. SpinRite is showing those three SMART parameters in RED bars because it holds the maximum positive health value it has seen since it was started, and any subsequent drop in those values below the maximum seen is shown as a RED drop downward, which is never good.

This screenshot also shows us that many other SMART health parameters the drive is reporting have remained pinned at their peak of 100%. Seek errors, recalibrate retries, cabling errors, uncorrectable errors, write errors and pending sectors are not worrying the drive at all. They are 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 a 1 out of 200. These all reveal that something is very wrong with this drive.

The question is not “should I return it” but “how quickly can I return and replace it?”

And this brings me to one of the two points I wanted to make: If a drive is just sitting there doing nothing but spinning 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, 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. When human doctors want to test someone’s cardiac function they 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 cannot reveal 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. SpinRite is not abusing a drive in any way. It’s just saying “How would you feel about doing some reading and writing?” Three of those identical drives respond saying “Sure thing!” while one of the four is really very unhappy about being asked to do what it was designed to do.

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. And 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 someone else wound up with Compaq’s rejects.

It’s interesting that even though today’s technology could hardly be more different, and we’re talking about 8 terabyte drive – 8 trillion bytes – rather than the 30 or 40 megabytes back in those early Compaq days, some things have not completely changed, and SpinRite has remained useful for performing pre-deployment hard drive testing.

Closing the Loop

Stephen

Hi Steve, Another incredible podcast breaking down OTP but I'd like to drop a spanner in the machine, sorry! If an attacker is trying to brute force a OTP, they already have the user's creds, which means the code space is reduced to 1 million... the weakest link in the chain.

In theory a bad actor could easily spin up a few hundred cloud instances, and distribute the 2FA attempts across them. Multiple simultaneous attempts within the 30 second time window doesn't have to get the OTP 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

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. After writing and sharing that last week, I've been thinking about it since. I realized that there's a somewhat cleaner and simpler way to think about the entire thing.
Since it's a different construction of the same solution, I want to share it.

First, we once again assume that we have some set of sample outputs from an authenticator, where each output is a 6-digit code and that code's timestamp. 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. 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. So, as we test a new candidate key against our set of samples, each successful match allows us to be one million times **more** sure that we have found the **one** proper key that will match every sample we can test. Since 80 bits allows for 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 be sure.

I did want 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 MFA systems having made exactly this mistake.

Joe Havlat *Subject: 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 have 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!

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.

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 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," S-T-U-N and T-U-R-N, 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!

Jason Harris

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: 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 it to. Thanks for any insights you can share! Best regards, Jason

Many many years ago, 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 the state of the art at the time. 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."

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 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. So it was clear that I needed something uncommon.

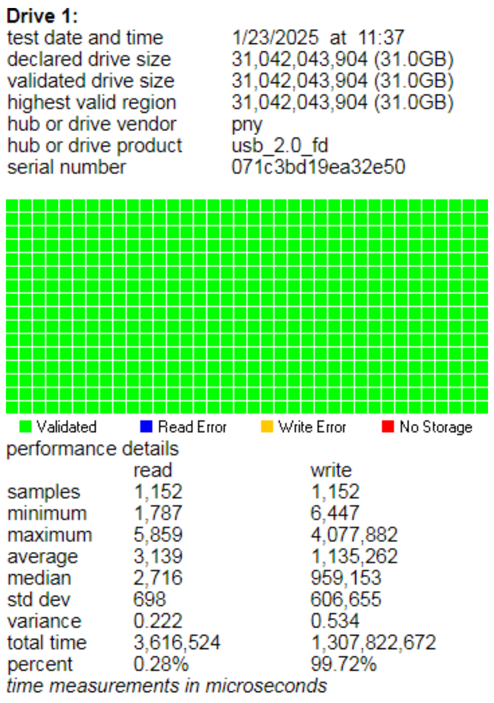
The other thing I wondered was how long it would take for an uncommon email address to escape into spammer's hands. And this is where Jason's thought of "*I suspect this might be due to how long I've had the address and how many services I've linked it to.*" What I started doing at least 15 years ago, is changing my email address annually. I'll keep forwarding all previous years' email into my current email so that I don't miss it. But anything I generate will be from the current year so awareness of my current email tends to migrate forward organically. And if at some point some annoying spammer does start using an older email account 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, 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's 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. But it is. And it's been confirmed by others with whom I've shared this simple discovery. 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. Let me know a few years from now!

Jeff Parrish

I purchased a 10 pack of PNY 16GB thumb drives, this is the results I received on 2 of them so far. I will be checking all 10.



Drive 1:

test date and time	1/23/2025 at 11:37
declared drive size	31,042,043,904 (31.0GB)
validated drive size	31,042,043,904 (31.0GB)
highest valid region	31,042,043,904 (31.0GB)
hub or drive vendor	pny
hub or drive product	usb_2.0_fd
serial number	071c3bd19ea32e50

🟩 Validated 🟦 Read Error 🟨 Write Error 🟥 No Storage

performance details

	read	write
samples	1,152	1,152
minimum	1,787	6,447
maximum	5,859	4,077,882
average	3,139	1,135,262
median	2,716	959,153
std dev	698	606,655
variance	0.222	0.534
total time	3,616,524	1,307,822,672
percent	0.28%	99.72%

time measurements in microseconds

Jeff's email included two screenshots of ValiDrive's display for two of the ten-pack of 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. And 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. Who would have ever imagined this day? (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.)

But aside from that, what really stopped me in my tracks about Jeff's thumb drives was the total time reading and writing. ValiDrive performs a pseudo-random spot-test by reading and writing 1152 4-Kbyte regions uniformly spread across the drive's self-declared size. It reads, writes, re-reads and re-writes each location, gathering statistics while it's doing this.

During this process, a grand total of 3.6 seconds was spent reading whereas 1307.8 seconds was spent writing. That's **3.6 seconds** spent reading and **21.8 minutes** spent writing.

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, 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 the floating gate's insulation. That high voltage charge must be dumped before data can be read. 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 write 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 – 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 work. 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 1152 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 2 switchings from reading to writing for ValiDrive 2 whereas ValiDrive 1 is doing that 2,304 times. 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. It's the thing I plan to work on as soon as work on the DNS Benchmark is finished.

AI Nervous Breakdowns?

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've spent a great deal of time 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?*

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..."* and so on.

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 the opportunity created by this 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 a total joke. Really. It should almost be an embarrassment to the AI community for anything they are doing to be called "neural" in any way. But it's true certainly true that calling them "high-speed GPU networks" is 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. 53 years ago, when I worked at Stanford's AI lab between 1971 and '73, we had what we referred to as the "Hand-Eye Table." In fact, it was by writing code for that vision that I earned and landed a multi-year job there while and after I was in high school. The hand-eye table had two anodized robot arms and a pair of video cameras. We knew that they bore no resemblance to actual hands or eyes. So it was just a bunch of computer geeks giving things fun and loosely meaningful names.

But things have changed a great deal since then, and "neural networks" have obviously moved from the lab into daily mainstream life. So it's somewhat worrisome that the "neural network" moniker has stuck around – because it can be so misleading – and that's beginning to matter. 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.

They're not. We do not even begin to have the capacity or capability 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 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 surprising and emergent property of their ability to self organize. And **that** behavior over on the artificial side was discovered and applied more than 50 years ago. So that's nothing new. Since then, the work has been about scaling and research to discover the best "pre-organization" to give to these untrained artificial networks.

But make no mistake about it, the collision of naming, where both artificial networks and biological networks employ the term "neural" could not be any further from the truth on the artificial side. Artificial? 100%. Neural? Not by any stretch of rationality.

Lyle Haylett

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 (drone operator) as well as a certified Private and Instrument Rated pilot. I utilize 2 DJI drones and a home-built drone to do commercial 3D mapping, photography and videography for the construction, real estate, and other businesses. 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.

This is great! I love our listeners! Here's someone who is right in the middle of all this. Lyle 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 is perfectly legal and safe to fly, 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 of the drone. 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, creating a new hazard.

Moreover, and my biggest concern with the old DJI Geo Zones, is that many, particularly recreational flyers, believe that if they are OK according to DJI Geo Zones, then they are 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 are 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 TN

Thank you so much, Lyle! It's so valuable to receive feedback from someone who has a broader perspective and experience with the subject. It seems very clear that no DJI was really not giving anyone "the middle finger" was 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. Thanks again, Lyle.

Tim Clevenger

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

Thanks, Tim.

George Adamopoulos

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 talk about in GRC #1009. As Leo mentioned, Windows already had an email client (Windows Mail). What you did not mention is that this is being deprecated in favour 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 31, 2024" (yes, that's in the past). 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

There's not much that can be added 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 an "All-compatible productivity tool", claiming "eM Client's compatibility is unmatched. Connect to any email service – it will work. Even outlook business email. And they highlight its interoperability with Google Workspace, Office 365, Outlook and Exchange. And as I noted last week,

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 user's web accesses. What I discovered was initially surprising until I sat back and thought about it a bit. And I believe that what I've learned will be of at least intellectual curiosity to many of 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 on line with all recent operating systems, most ISPs and the intervening equipment such as consumer routers now supporting IPv6.

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 their advertisers to our ISPs who connect us to the Internet. Since many who use the Internet 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.

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 utility 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.

Today's DNS Benchmark is protocol agnostic – any protocol can be added to its underlying structure. So it's now ready to handle today's newer DNS protocols and whatever else the future might hold.

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 0's compression that's used to shorten the many IPv6 addresses containing one or more words of all 0's.

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 to a large collection of very different underlying secure protocol, 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, 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 5th generation of this system. The code that I finally have is actually quite lovely and I’m proud of it. It’s far more clear and clean that 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 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, 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 to query a couple of public databases to obtain the ownership of the IP address space housing the nameserver to create a richer experience.

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 lookup the IP address for 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 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, so they’ll simply ask all of the nameservers they’re configured for and accept the first reply they receive.

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 that 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. So it offers no privacy.

We know what encryption requires. At the bare minimum it requires that the entities at each end of any conversation share a secret that no one else can possibly know. They 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 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 is by moving from the minimal elegance of UDP – the User Datagram Protocol – to the reliable delivery system provided by TCP, the Transmission Control Protocol.

So that's what I built.

For every remote nameserver that the DNS Benchmark will be testing, it looks up the IP address for that 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 by negotiating encryption protocols, receiving and verifying the remote nameserver's certificate, agreeing upon a shared secret key, then bringing up their encrypted tunnel. (That's the whole weirdly opaque SChannel API stuff I spoke about earlier.)

At this point – Whew! – Yay! – we have a connection to a remote DNS over TLS nameserver which should allow us to send and receive DNS queries.

So it was with great joy and celebration that I got all that working, whereupon the remote nameservers began unceremoniously disconnecting and dropping their connections without warning or reason and with prejudice.

I tried it a few times and the same thing kept happening. It seemed that these nameservers were impatient for queries. And they were not uniformly impatient. Some would drop the connection after a second, some would wait five seconds. 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 DNS resolutions. So I started having the benchmark send them DNS queries to answer over the 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. 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 were dropping connections.

After sitting back and thinking about this for a few minutes the reason for all of this became obvious: Compared to unencrypted UDP queries and replies, TCP – and especially TLS over TCP connections – are incredibly expensive, not only to establish, but also 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 is 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 or received, is 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 be handled 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 version that were negotiated and the shared secret key for encrypting and decrypting, and the state of all of the many options that have been added to TLS from the start of SSL through TLS v1.3.

In other words, it's a lot. And now consider all that in comparison to plain old original 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. Zero. Nothing to remember between queries. No state to preserve. No connections.

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 single second of every single day. For UDP, that's no problem. They just do it. 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 that, 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 DNS 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 this all 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 browser's use of DNS is inherently bursty.

When a user clicks a link to jump to a new web page that it has 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 to 150 DNS domain names. Those are the domains for the advertisements, the script libraries, the images, the various tracking servers 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 an open 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 addresses it needs to fetch all 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 to generate another brief flurry of queries. And so it goes.

This means that bringing up relatively short-lived (and very expensive to maintain) TCP-TLS connections winds up being cost effective. It's true that doing all of this connecting, establishing and negotiating takes time and multiple – many – packet round trips. But once that's been done, the DNS queries and replies are able to occur with the same speed as regular DNS, even though now they're 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 burst 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 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 re-connects to the same server. The server decrypts the blob, which no one else has any way to use. And if everything matches up, the client and server are able to bypass all of the time-consuming TLS negotiation to pick up right where they left off.

Having thus understood what's going on with TLS nameservers, GRC's benchmark is now working with every one of them I've found. And since DNS over HTTPS just wraps the DNS query and response inside HTTP protocol which 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. I imagine that DNS over UDP will eventually go the way of good old unencrypted HTTP.