Security Now! #998 - 10-29-24
The Endless Journey to IPv6

This week on Security Now!

Apple proposes 45-day maximum certificate life. Please, no. :(SEC fines four companies for downplaying their SolarWinds attack severity. Google adds 5 new features to Messenger including inappropriate content. Does AI-driven local device-side filtering resolve the encryption dilemma forever? The very nice looking "Session" messenger leaves Australia for Switzerland. Another quick look at the question of the EU's software liability moves. Fake North Korean employees WERE found to install backdoor malware. How to speed up an SSD without using SpinRite. Using ChatGPT to review and suggest improvements in code. And Internet governance has been trying to move the Internet to IPv6 for the past 25 years, but the Internet just doesn't want to go. Why not? And will it ever?

There really are no words…

Security News

Apple proposes a certificate life reduction timeline

As our long-time listeners know, many years ago we spent many podcasts looking at the fiasco that was, and sadly still is, certificate revocation, noting that the system we had in place using CRLs – certificate revocation lists – was totally broken. I put GRC's "revoked.grc.com" server online to vividly demonstrate the lie we were being told.

At the time, the OCSP solution – Online Certificate Status Protocol – seemed to be the best solution. But if users' browsers queried for OCSP status that created serious performance and also privacy issues. The solution to that was OCSP stapling, where the website's own server would make the OCSP queries then "staple" fresh results to the certificate it was serving.

But it seems that asking every web server in the world to do that was too high a bar to reach. So, despite its promise and partial success, the CA/Browser forum, which sets the industry's standards, recently decided to backtrack and return to the use of the earlier Certificate Revocation List system which would move all of the website certificate checking to the browser. This had the benefit of allowing us to offer a terrific podcast explaining the technology of Bloom Filters that allow the user browser to locally and very quickly determine the revocation status of any incoming certificates.

For a certificate to be valid two things must be true: We must be between the certificate's "not valid before" and "not valid after" dates, and there must not be any indication that this otherwise valid certificate has nevertheless been revoked for some reason. The conjunction of these two requirements means that CRLs only need to cover any certificates that would otherwise be valid. This means that expired certificates will be distrusted due to their expiration and can thereby be safely removed from the next update to the industry's Bloom Filter based CRL list. If we want to keep the sizes of our Bloom Filter CRLs down, shortening the lives of certificates is the way to do that.

And this brings us to last week's news of a proposal by Apple, an active member of the CA/ Browser forum, to gradually reduce maximum web server certificate life from its current duration of 398 days all the way down to just 45 days.

If this proposal were adopted, certificates would have their lives reduced in four steps starting one year from now and ending in April of 2027, as follows: We're currently at 398 days. That comfortable 398 would be cut nearly in half to 200 days, one year from now in September of 2025. Then a year later, in September of 2026, it would be reduced by half again to 100 days. And the final reduction would occur seven months later, in April of 2027, which would reduce web server certificate maximum lifespans to just 45 days.

The only reason Let's Encrypt's 90-day certificate lifetimes are workable is through automation. So Apple must be assuming that by setting a clear schedule on the plan to decrease certificate lifespans, anyone who has not yet fully automated their server's certificate issuance with the ACME protocol will be motivated to do so.

This creates some potential edge-case problems, however, since it's not only web servers that depend upon TLS certificates. For example, GRC signs its outbound email using a mail server that's manually configured to use the same certificate files that are valid for the grc.com domain.

At the moment, I only need to update those annually, so it's manageable to do that through the email server's UI. I don't know what would happen if I were to change the content of the files out from under it without it knowing. For all I know, since it has private copies of the certificates, it might be holding their files open, thus preventing them from being changed. There's currently no programmatic way to inform the email server that it needs to change its certs since this has never been a problem or a necessity until now. Since I can code, I might wind up having to add a new custom service to watch for my web server autonomously changing its certificates, then shutdown the email server, update its copies of the certs and restart it. My point is, that's what's known as a royal F'ing kludge, and it's no way to run the world.

And make no mistake, my email server is just a trivial example of the much larger problem on the horizon. Think of all the Non-ACME-aware non-web server appliances we have today that have proliferated in the past decade and which now also all need certificates of their own.

Perhaps this is the price we pay for progress, but I question why this should be imposed upon us and upon me, for example? It's **MY** certificate. It represents **MY** domain of GRC.COM. Why is it not also **MY CHOICE** how long that representation should be allowed to endure? If I'm some big organization like Amazon.com, Bank of America or PayPal, where a great deal of damage could be done if a certificate got loose, I can see the problem. So such organizations ought to be given the OPTION to shorten their certificates' lives in the interest of their own security. And, in fact, they can do that today. When I'm creating certificates at DigiCert I'm prompted for the certificate's duration. 398 days is the maximum lifetime allowed – there is no minimum. And DigiCert supports the ACME protocol, so automation for short lived certificates is available. But why are short-life certificates going to be imposed upon websites by the CA/Browser forum and the industry's web browsers? And let's get real here. As we know, revocation has **never** worked. Never. It's always been a feel good fantasy, and the world didn't end when we only needed to reissue certificates once every three years with no effective ability to revoke. Now the industry wants to radically reduce that to every six weeks? How are we not trying to solve a problem that doesn't actually exist, while at the same time creating a whole range of new problems we've never had before? I'll bet there are myriad other instances, such as with my email server, where super-short-lived certificates will not be practical. This sure seems like a mess being created without full consideration of its implications. Do these folks at the CA/ Browser forum fail to appreciate that web servers are not the only things that require TLS connections and the certificates that authenticate them and provide their privacy?

I dropped my use of EV certificates because that became wasted money once browsers no longer awarded those websites any special UI treatment. But I've continued using OV, organization validation certificates, since they were one notch up from the lowest form of certificate, the simple DV, domain validation certs which Let's Encrypt uses. But if we're all forced to automate certificate issuance I can't see any reason why everyone won't be pushed down to that lowest common denominator of domain validation certificates, the issuance of which Let's Encrypt has successfully automated. At that point, certificates all become free, and today's Certificate Authorities lose a huge chunk of their recurring business. How is that good for them? And the fact is, simple Domain Validation provides a lesser level of assurance than Organization Validation. So how is forcing everyone down to that lowest common denominator good for the overall security of the world?

I suppose that Apple, with their entirely closed ecosystem may see an advantage to this. So fine. They're welcome to have super short lived certificates for their own domains. But more than anything, I'm left wondering why the lifetime of the certificates **I** use to assert the validity of **MY** own domain is not **MY** business and why that's not being left up to me?

The SEC levies fines for downplaying the severity of the SolarWinds attacks
One of the rules of the road is that companies that are owned by the public through publicly traded stock have a fiduciary duty to tell the truth to their stockholders when something occurs that could meaningfully affect the company's value.

For example, on December 14th, 2020, the day after The Washington Post reported that multiple government agencies had been breached through SolarWinds's Orion software, the company stated in an SEC filing that fewer than 18,000 of its 33,000 Orion customers were affected. Still, 18,000 customers affected made it a monumental breach. And there was plenty of fault to be found in SolarWinds' previous and subsequent behavior.

But I didn't bring this up to talk about them. I wanted to share some interesting reporting by CyberScoop whose headline a week ago was: *"SEC hits four companies with fines for misleading disclosures around SolarWinds hack."* In other words, for misleading the public about the impact their use of SolarWinds' Orion software had on their businesses.

CyberScoop's sub-head read: *"Unisys, Avaya, CheckPoint and Mimecast will pay fines to settle charges that they downplayed in SEC filings the extent of the compromise."* And this is the point that I wanted to make. The management of companies owned by the public need to tell the truth. So let's take a closer look at this. CyberScoop write:

The Securities and Exchange Commission said it has reached a settlement with four companies for making materially misleading statements about the impact of the 2020 SolarWinds Orion software breach on their business.

The regulator charged the four companies — Unisys Corp., Avaya Holdings Corp., CheckPoint Software Technologies and Mimecast Limited — with ***minimizing the compromise or describing the damage to internal systems and data as theoretical, despite knowing that substantial amounts of information had been stolen.***

[In other words, the SEC says they outright lied to their public shareholders.]

The acting director of the SEC's Division of Enforcement said in a statement: "As today's enforcement actions reflect, while public companies may become targets of cyberattacks, it is incumbent upon them to not further victimize their shareholders or other members of the investing public by providing misleading disclosures about the cybersecurity incidents they have encountered. Here, the SEC's orders find that these companies provided misleading disclosures about the incidents at issue, leaving investors in the dark about the true scope of the incidents."

As part of the settlement agreements reached, the companies have agreed to pay fines with no admission of wrongdoing. Unisys will pay $4 million, Avaya $1 million, CheckPoint $995,000 and Mimecast $990,000.

According to the SEC, by December 2020 Avaya already knew that at least one cloud server holding customer data and another server for their lab network had been breached by hackers working for the Russian government. Later that month, a third-party service provider alerted the company that its cloud email and file-sharing systems had also been breached, likely by the same group and through means other than [the SolarWinds] Orion [software].

A follow-up investigation identified more than 145 shared files accessed by the actor, along with evidence that the [Russian] group [known as APT29 aka CozyBear] monitored the emails *of the company's cybersecurity incident responders.*

Despite this, in a February 2021 quarterly report several months later, Avaya described the impact in far more muted terms, saying the evidence showed the threat actors accessed only "a limited number of company email messages" and there was "no current evidence of unauthorized access to our other internal systems." [Avaya knew that this representation was flatly false.]

Unisys' investigation uncovered that, following the discovery of a device running Orion, multiple systems — seven network and 34 cloud-based accounts, including some with administrative privileges — were accessed over [the course of] 16 months. The threat actors also repeatedly connected to their network and transferred more than 33 gigabytes of data.

But the SEC's cease and desist order stated that Unisys had <quote> "inaccurately described the existence of successful intrusions and the risk of unauthorized access to data and information in hypothetical terms, despite knowing that intrusions had actually happened and, in fact, involved unauthorized access and exfiltration of confidential and/or proprietary information." The company also appeared to have no formal procedures in place for identifying and communicating high-risk breaches to executive leadership for disclosure.

Similar investigations by CheckPoint in December 2020 found two infected servers and evidence of the threat actor moving throughout its corporate network and installing malicious software. Still, in SEC filings in 2021 and 2022, Check Point described its level of general cybersecurity risk using near-identical language from past reports, despite knowing it had been victimized by a Russian advanced persistent threat group in a wide-ranging global hacking campaign.

Cloud provider Mimecast discovered evidence the hackers used a stolen authentication certificate to breach five customer cloud platforms, access internal emails and stole code for an encrypted database holding credentials and server configuration information for tens of thousands of customers. While Mimecast publicly disclosed some of these impacts, its reporting to the SEC omitted critical details "including information regarding the large number of impacted customers and the percentage of code exfiltrated by the threat actor."

I'd like to be able to draw a clear moral to this story. But given the extremely modest size of the settlements relative to each company's revenue, it's not clear that the moral of our story is that they should have divulged more. During the heat of the moment, the short term impact upon their stock price may have been more severe than these fines. And coming four years after the event it's reduced to a shrug.

So I doubt that this outcome will wind up teaching any other companies any important lessons. And any companies that did the right thing and were then punished by their stockholders for telling the truth might take away the opposite lesson: "Let's just lie, sweep it all under the rug for now, and if three or four years later we're hit with a modest tax for doing so, after the whole world has moved on, we'll happily pay it, then."

Google updates "Google Messages" with 5 new features.
Last Tuesday, Google's Security Blog posted the news of five new protections being added to their Google Messages app. Although Google's postings are often a bit too full of marketing hype for my taste, I thought this one was worth sharing. Google wrote:

Every day, over a billion people use Google Messages to communicate. That's why we've made security a top priority, building in powerful on-device, AI-powered filters and advanced security that protects users from 2 billion suspicious messages a month. With end-to-end encrypted RCS conversations, you can communicate privately with other Google Messages RCS users. And we're not stopping there. We're committed to constantly developing new controls and features to make your conversations on Google Messages even more secure and private.

As part of cybersecurity awareness month, we're sharing five new protections to help keep you safe while using Google Messages on Android:

1. Enhanced detection protects you from package delivery and job scams.
Google Messages is adding new protections against scam texts that may seem harmless at first but can eventually lead to fraud. For Google Messages beta users, we're rolling out enhanced scam detection, with improved analysis of scammy texts, starting with a focus on package delivery and job seeking messages. When Google Messages suspects a potential scam text, it will automatically move the message into your spam folder or warn you. Google Messages uses on-device machine learning models to classify these scams, so your conversations stay private and the content is never sent to Google unless you report spam. We're rolling this enhancement out now to Google Messages beta users who have spam protection enabled.

2. Intelligent warnings alert you about potentially dangerous links.
In the past year, we've been piloting more protections for Google Messages users when they receive text messages with potentially dangerous links. In India, Thailand, Malaysia and Singapore, Google Messages warns users when they get a link from unknown senders and blocks messages with links from suspicious senders. We're in the process of expanding this feature globally later this year.

3. Controls to turn off messages from unknown international senders.
In some cases, scam text messages come from international numbers. Soon, you will be able to automatically hide messages from international senders who are not existing contacts so you don't have to interact with them. If enabled, messages from international non-contacts will automatically be moved to the "Spam & blocked" folder. This feature will roll out first as a pilot in Singapore later this year before we look at expanding to more countries.

4. Sensitive Content Warnings give you control over seeing and sending images that may contain nudity.
At Google, we aim to provide users with a variety of ways to protect themselves against unwanted content, while keeping them in control of their data. This is why we're introducing Sensitive Content Warnings for Google Messages.Sensitive Content Warnings is an optional feature that blurs images that may contain nudity before viewing, and then prompts with a "speed bump" that contains help-finding resources and options, including to view the content. When the feature is enabled, and an image that may contain nudity is about to be sent or forwarded, it also provides a speed bump to remind users of the risks of sending nude imagery and preventing accidental shares.All of this happens on-device to protect your privacy and keep end-to-end encrypted message content private to only sender and recipient. Sensitive Content Warnings doesn't allow Google access to the contents of your images, nor does Google know that nudity may have been detected. This feature is opt-in for adults, managed via Android Settings, and is opt-out for users under 18 years of age. Sensitive Content Warnings will be rolling out to Android 9+ devices including Android Go devices with Google Messages in the coming months.

5. More confirmation about who you're messaging.
To help you avoid sophisticated messaging threats where an attacker tries to impersonate one of your contacts, we're working to add a contact verifying feature to Android. This new feature will allow you to verify your contacts' public keys so you can confirm you're communicating with the person you intend to message. We're creating a unified system for public key verification across different apps, which you can verify through QR code scanning or number comparison. This feature will be launching next year for Android 9+ devices, with support for messaging apps including Google Messages.

I want to skip back to that 4th new feature: Sensitive Content Warnings. Apple announced their Sensitive Content Warnings in iOS 15 where the smartphone would detect probably-sensitive content and warn its user before displaying it. Despite that potentially privacy invading feature which has now been in place for several years we're all still here. Not only did the world not end when Smartphones began looking at what their users were doing, it didn't even slow down. So the idea of device-side image recognition and detection has not proven to be a problem, and Google has clearly decided to follow. But I believe there may be a larger story here. I suspect that this will be the way the world ultimately resolves the thorny end-to-end encryption dilemma:

As we know, Apple initially really stepped in it by telling people that their phones would be pre-loaded with an exhaustive library of the world's most horrible known CSAM – Child Sexual Abuse Material. No one wanted anything to do with having that crap on their phone, and even explaining that it would be fuzzy matching hashes rather than actual images did nothing to mollify those who said *"Thanks anyway. I'll go get an Android phone before I like you put anything like that on my phone."* Apple received that message loud and clear and quickly dropped that effort.

But then, right in the middle of the various European governments' and especially the UK's very public struggles with this issue, facing serious pushback from every encrypted messaging vendor saying that they would rather leave than compromise their user's security, AI suddenly emerges on the scene and pretty much blows everyone's mind with its capabilities and with what it means for the world's future.

If there's been any explicit mention of what AI will mean for highly effective, local, on-device filtering of personal messaging content I've missed it. But the application seems utterly obvious, and I think this solves the problem in a way that everyone can feel quite comfortable with. The politicians get to tell their constituents that "next generation AI will be watching everything their kid's smartphones send and receive and will be able to take whatever actions are necessary without ever needing to interfere with or break any of the protections provided by true, full, end-to-end encryption." So everyone retains their privacy with full encryption and the bad guys will soon learn that they're no longer able to use any off-the-shelf smartphones to send or receive that crap. It seems to me this really does put that particular intractable problem to rest... and just in the nick of time!

I'll note one more thing about this. It's foreseeable that the behavior recognition provided by AI-based on-device filtering will eventually be extended to encompass additional unlawful behavior. We know that governments and their intelligence agencies have been credibly arguing that terrorists are using impenetrable encryption to organize their criminal activities. So I would not be surprised if future AI-driven device-side detection were not further expanded to encompass more than just the protection of children. This, of course, raises the specter of Big Brother behavior monitoring and profiling, which is creepy all by itself. And I'm not suggesting that's entirely a good thing, because it does create a very slippery slope. But at least there we can apply some calibration and implement whatever policies we choose as a society. What **IS** an entirely good thing is that those governments and their intelligence agencies who have been insisting that breaking encryption and monitoring their populations is the only way to be safe will have had those arguments short circuited by AI. Those arguments will finally be put to rest with encryption having survived intact.

“Session” leaves Australia for Switzerland

While we’re on the subject of encrypted apps, an app known as “Session” is a small but increasingly popular encrypted messaging app. Session announced that it would be moving its operations outside of Australia after the country’s federal law enforcement agency visited an employee’s residence and asked them questions about the app and about a particular user of that app. As a result of that nighttime intrusion, Session will henceforth be maintained by a neutral organization based in Switzerland.

404 Media noted that this move signals the increasing pressure on maintainers of encrypted messaging apps, both when it comes to governments seeking more data on app users, as well as targeting messaging app companies themselves. They cited the recent arrest of Telegram’s CEO in France last August. Alex Linton, the president of the newly formed Session Technology Foundation (STF) which will publish the Session app from Switzerland told 404 Media in a statement: *“Ultimately, we were given the choice between remaining in Australia or relocating to a more privacy-friendly jurisdiction, such as Switzerland. The app will still function in Australia, but we won’t.”*

I wasn’t aware of the “Session” messaging app, but it looks quite interesting and I wanted to put it on everyone’s radar. It appears to be what you would get if you were to combine the ultra-robust and well proven Signal protocol – which Session forked – with the distributed IP-hiding Tor-style Onion routing which we briefly discussed again recently. And on top of all that, Session is 100% open source and living on Github.

Since all of this piqued my curiosity I tracked down a recent White Paper describing “Session” from July of this year. It’s titled: *“Session: End-To-End Encrypted Conversations With Minimal Metadata Leakage”*. The White Paper’s Abstract described Session:

Session is an open-source, public-key-based secure messaging application which uses a set of decentralized storage servers and an onion routing protocol to send end-to-end encrypted messages with minimal exposure of user metadata. It does this while providing the common features expected of mainstream messaging applications, such as multi-device syncing, offline inboxes, and voice/video calling.

Huh. Wow. I would imagine that the Australian Feds were probably left quite unsatisfied by the answers anyone knowledgeable of Session’s design would have provided. They would have explained that Session’s messaging transport was deliberately designed like Tor’s to hide each endpoint’s IP address through a multi-hop globally distributed server network, and that the entire content of the messages used the impenetrable Single protocol used by Signal and WhatsApp to exchange authenticated messages between the parties.

And if this didn’t already sound wonderful, listen to the system’s mission statement from the White Paper’s Introduction:

Over the past 10 years, there has been a significant increase in the usage of instant messengers, with the most widely-used messengers each having amassed over 1 billion users. The potential privacy and security shortfalls of many popular messaging applications have been widely discussed. Most current methods of protecting user data privacy are focused on encrypting the contents of messages, an approach which has been relatively successful. The widespread deployment of end-to-end encryption does increase user privacy; however, it largely fails to address the growing use of metadata by corporate and state-level actors as a method of tracking user activity.

In the context of private messaging, metadata can include the IP addresses and phone numbers of the participants, the time and quantity of sent messages, and the relationship each account has with other accounts. Increasingly, it is the existence and analysis of this metadata that poses a significant privacy risk to journalists, protesters, and human rights activists.

Session is, in large part, a response to this growing risk: it provides robust metadata protection on top of existing cryptographic protocols which have already been proven to be effective in providing secure communication channels.

Session reduces metadata collection in three key ways:

Firstly, Session does not require users to provide a phone number, email address, or other similar identifier when registering a new account. Instead, pseudonymous public-private key pairs are the basis of an account's identity.

Secondly, Session makes it difficult to link IP addresses to accounts or messages sent or received by users, through the use of an onion routing protocol.

Thirdly, Session does not rely on central servers; a decentralized network of thousands of economically incentivised nodes performs all core messaging functionality. For those services where decentralization is impractical, like storage of large attachments and hosting of large group chat channels, Session allows users to self-host infrastructure, and rely on built-in encryption and metadata protection to mitigate trust concerns.

In other words... wow. As we know, Pavel Durov the Telegram guy, freed himself by agreeing to, where warranted, share IP addresses and whatever other metadata Telegram collected with law enforcement. And we know that Apple, Signal and WhatsApp all similarly keep themselves out of hot water with governments and law enforcement by cooperating to the degree they're able to. And they **are** able to. They're able to provide IP addresses and related-party identifiers. They may not be able to peer into the content of conversations, but the fact of those conversations and the identity of the parties conversing is knowable, sharable and shared.

So I suppose we should not be surprised that the guys who married the Signal messaging protocol with Tor's Onion routing to deliberately create a hyper-private messaging system saw the clear handwriting on the wall and decided – after that visit from their local Feds – that they would need to move away from Australia sooner or later, so it might as well be sooner.

The messaging app is called "Session" and it's available in several flavors for Android and iOS smartphones as well as for Windows, Mac and Linux desktops. From here it appears to be a total win. Establishing an anonymous identity with a public/private key pair is exactly the right way to go and that's exactly what they do plus much more and all with their source code being openly managed on Github.
In addition to the full 34-page technical White Paper there's a highly accessible 5-page "Light

Paper" which carries their slogan: "Send messages, not metadata." The URL is "getsession.org" where you'll find a software download page (https://getsession.org/download) as well as links to both that 5-page "LightPaper" and the full 34-page "WhitePaper".

Software Liability

The EU's proposed wholesale revision of the software liability has, not surprisingly, drawn a huge amount of attention from the tech press. We gave it enough attention here last week, but I was glad to see that I didn't misstate or misinterpret that new EU Directive. It really is what it appears to be. One reporter about this wrote:

The EU and US are taking very different approaches to the introduction of liability for software products. While the US kicks the can down the road, the EU is rolling a hand grenade down it to see what happens.

Under the status quo, the software industry is extensively protected from liability for defects or issues and this results in systemic underinvestment in product security. Authorities believe that by making software companies liable for damages when they peddle crapware, those companies will be motivated to improve product security.

I also want to share part of what another writer wrote for The Record:

Six years after Congress tasked a group of cybersecurity experts with reimagining America's approach to digital security, virtually all of that group's proposals have been implemented. But there's one glaring exception that has especially bedeviled policymakers and advocates: a proposal to make software companies legally liable for major failures caused by flawed code.

Software liability was a landmark recommendation of the Cyberspace Solarium Commission, a bipartisan team of lawmakers and outside experts that dramatically elevated the government's attention to cyber policy through an influential report that has seen roughly 80% of its 82 recommendations adopted. Recent hacks and outages — including at leading vendors like Microsoft and CrowdStrike — have demonstrated the urgent need to hold software companies accountable, according to advocates for software liability standards.

But despite the Solarium Commission's high-profile backing and the avowed interest of the Biden administration, this long-discussed idea has not borne fruit. Interviews with legal experts, technologists and tech-industry representatives reveal why: Software liability is extremely difficult to design, with multiple competing approaches, and the industry warns that it will wreck innovation and even undermine security.

Jim Dempsey, senior policy adviser at Stanford University's Program on Geopolitics, Technology and Governance said: "The Solarium Commission and Congress knew that this was going to be a multi-year effort to get this done. This is a very, very, very hard problem."

A recent spate of massive cyberattacks and global disruptions — including the SolarWinds supply-chain attack, the MOVEit ransomware campaign, the Ivanti hacks, the CrowdStrike outage and Microsoft's parade of breaches — has shined a spotlight on the world's vulnerability to widely distributed but sometimes poorly written code.

Dempsey added: "There is a widespread recognition that something's got to change. We're way too heavily dependent on software that has way too many vulnerabilities."

The software industry's repeated failures have exasperated experts who see little urgency to address the roots of the problem. But bringing companies to heel will be extremely difficult.

An associate professor at Fordham School of Law who specializes in cybersecurity and platform liability said: "We have literally protected software from almost all forms of liability, comprehensively, since the inception of the industry decades ago. It's just a golden-child industry."

Virtually all software licenses contain clauses immunizing vendors from liability. Policymakers originally accepted this practice as the cost of helping a nascent industry flourish. But now that the industry is mature and its products power all kinds of critical services, it will be an uphill battle to untangle what Dempsey called the "intersecting legal doctrines that have insulated software developers from the consequences of the flaws in their products."

Cosmos Blockchain & DPRK
A recurring event in security news has been the industry's inadvertent hiring of fake IT workers, generally from North Korea. Hopefully this had not been happening for long undetected, since there suddenly seems to be a lot of it going around – at least now that everyone is on the lookout for it and should be aware. And I shared the hoops I had to jump through during that one-way video conference with a DigiCert agent, following his instructions as I moved my hands around my face, holding up and displaying my government issued ID.

As far as I know, the coverage of this hasn't actually revealed any malfeasance on the part of these North Korean employees. It's illegal to hire them, but that had been the extent of the trouble... until now. The creator of the Cosmos blockchain has admitted that the company inadvertently hired a North Korean IT worker. The company says the FBI notified the project about the North Korean worker, but the individual who received the notification did not report the incident to his managers. Huh?!?! And, moreover, Cosmos says that the code contributed by the North Korean worker contained at least one major vulnerability. The company is now performing a security audit to review the code for other issues.

Hopefully, these continuing revelations will lead to many more real-time video conferences such as I had with DigiCert.

Closing the Loop

Brian

Please add me to your Security Now podcast GRC list. I am an occasional listener and appreciate all of your information and tips shared. Regards, Brian

That's all he said, but I wanted to share this because Brian is a mode of listener who can obtain a value of GRC's weekly podcast synopsis mailing that I had never considered. I often hear from listeners who have fallen behind in listening or who aren't always able to find time to listen. So Brian's note made me realize that the weekly mailings which, at the moment, 11,716 people received, for example, yesterday evening before the podcast, can come in quite handy when making a determination about how to invest one's limited time.

Martin in Denmark,

Hi Steve. Love the podcast, been with you guys from episode 0! I have a question about the stuff SpinRite does when "speeding up an SSD." My computer is due for a reformat and reinstall of Windows. Windows is slowing down, as it does, but it seems worse than "usual" so I think my SSD could use a little help.

Since I'm going to "nuke" the drive anyway, is there a way to do the same stuff that SpinRite does without it? I assume that using Windows installer or diskpart to "clean" the disk just wipes the filesystem/partition table and does nothing else? Am I right that a "poormans" solution would be to delete the partitions on the drive and make a new one and then fill it with random data? I don't own SpinRite (money reasons yada yada :-)) and was just wondering if there is another way as I don't care about the data on the drive. Regards, Martin in Denmark

So here' s what I wrote in reply to Marin:

Hi Martin, You don't need SpinRite for that at all. The only magic SpinRite does, aside from perhaps helping (hugely) to recover from any trouble encountered in the process, is rewriting the data on an existing SSD. But it's the "writing", not the "rewriting" that's the key here. So if you're going to be reinstalling Windows, that act of re-installation will inherently be overwriting and thus – writing (which is the goal).

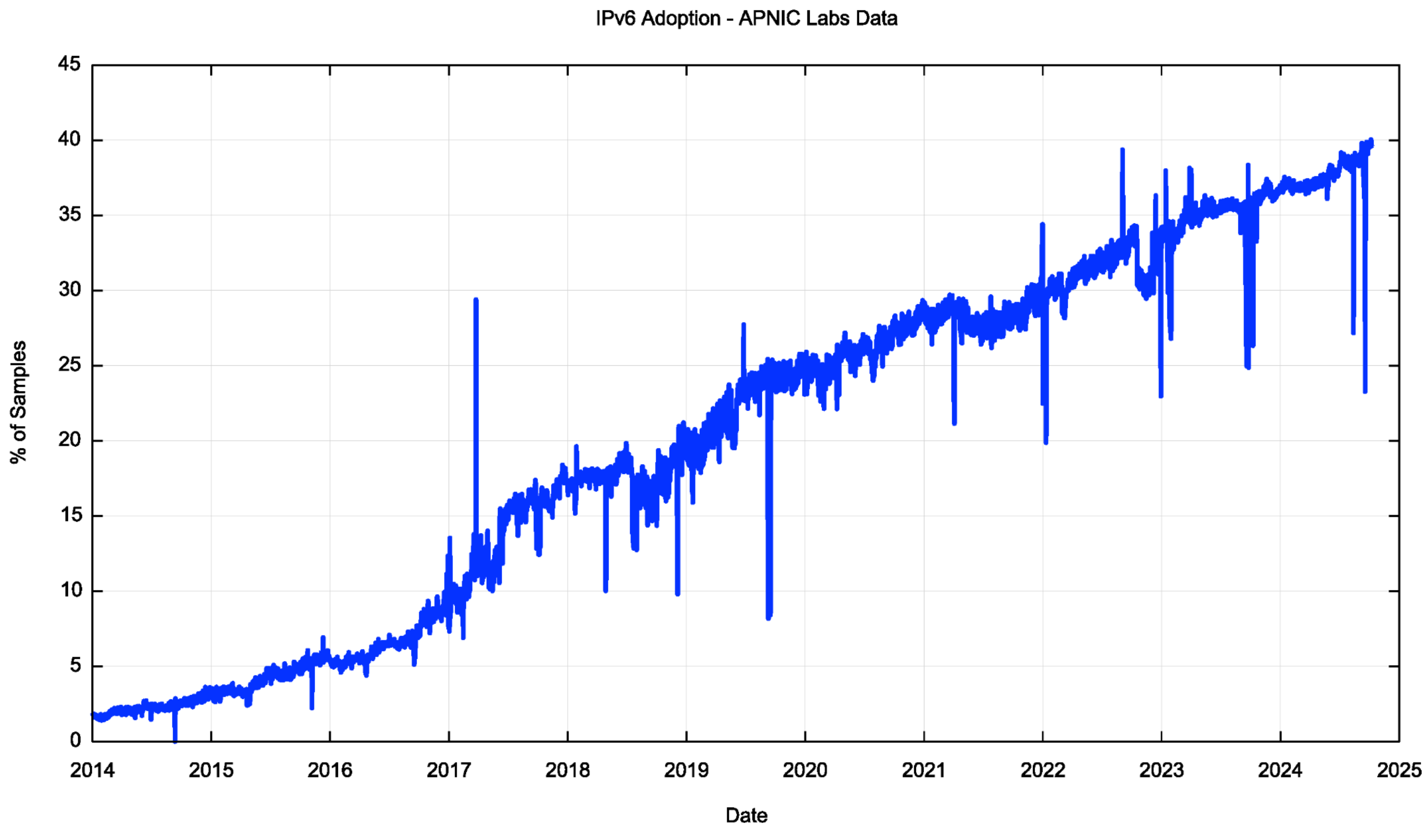
We've discovered that SSDs can grow surprisingly slow without otherwise complaining as the years go by without regions of their media ever being rewritten. SpinRite makes refreshing SSDs with data in place easy. But if retaining an SSDs current data is not needed then neither is SpinRite. A standard re-installation of Windows will entirely do the trick for you!

Alain Gyger / Radio: K6ACG

Hi Steve, I'm really liking the emails versus X ... thank you for switching! I do lots of Python programming and really like the code creation process. So I don't use ChatGPT to write my initial code, but I use it ***after*** *I've written a function. I just paste it in and ask ChatGPT to describe what it does. If I like the result I ask if there is any way to improve the code I've written. I do have my ChatGPT customized so that it prefers readability, descriptive function/variable names, etc. over shorter (and potentially more cryptic) code. This process fits well into my development flow and results in higher quality code. I hope this can help other people, it's been working well for me, Alain*

One of the things coders are always being told is that there's no better way to improve one's craft than to spend serious time reading others' code. Successful novelists will have always spent their earlier lives reading other people's novels and music composers grew up listening intently to endless compositions that preceded them. So it should be no surprise that reading others' code would be every bit as valuable. It's for this reason that I think Alain's idea is very interesting and useful. ChatGPT has been trained by reading vast quantities of other people's code. So I think it absolutely makes sense to ask an AI like ChatGPT whether it can see any way to improve upon some code that was just written. That appeals to me far more than asking it to do the work, first. If you're coding for the sheer pleasure of doing so, then don't give that up. But then also take the opportunity to learn by testing your creation against the distilled wisdom of everyone who previously posted their code to the Internet and influenced ChatGPT's training model.

The Endless Journey to IPv6



I know that the majority of our listeners need no introduction to the difference between IPv4 and IPv6. But I want to share some of a wonderful recent blog posting made by APNIC Labs. And since it assumes complete comfort with the IPv4 vs IPv6 distinction, I want to first share a very quick orientation.

IP stands for Internet Protocol and version 4 of the Internet Protocol is the original version that took off and became the world wide standard. By the mid 1990's the folks who created this first successful Internet were already starting to worry about its growth. So they started working on its successor replacement. That became known as IPv6 – or version 6 of the Internet Protocol. Although IPv6 changes many things from IPv4, the most prominent and significant is IPv6 addressing. Internet addresses are expressed by a set of binary bits, and any set of binary bits can only have so many possible combinations. The original IPv4 protocol uses 32 bits. Back before the Internet happened, when it was still just a "what if" experiment, it was believed that these 32-bits, which allowed for 4,294,967,296 individual Internet addresses, was believed to be more than ample. But today, around 20 billion devices are attached to the Internet and many people feel that the Internet is in trouble. If anyone wonders how this is possible, consider the number of Internet-connected devices in the average home, and that thanks to the miracle of NAT routing – network ADDRESS translation – they are all able to comfortably share the household's single ISP-assigned IP address.

The way to think about this is that the IPv4 protocol also set aside 16 bits for port numbers. Thus at any given 32-bit IPv4 address, an additional 16 bits are then used to specify the port number at that address.

So, when you think about this, if you think about the Internet as publicly addressing by PORT number rather than by HOST IP, port-based addressing yields an effective 48 bits of total addressing – 32 bits for the IP plus 16 bits for the port at that IP. Thus, what NAT routing does is borrow bits from IPv4's port numbering and reuse them as additional addressing bits.

This works, but it ***really*** upsets the Internet purists. These guys ***hate*** the idea (with a passion) of addressing by port because <quote> "That's not the way we designed it to work!"

Okay, so refocusing upon today's topic. Everyone agrees that IPv4 is being stretched and stretched way past its expected end of life. But why? We've had IPv6 since the last 1990's. So what's the holdup? At this point, two podcasts away from episode 1000, would any of our listeners be surprised to learn that it's nothing more than resistance and inertia and the fact that port addressing works well enough?

Okay. So first of all, who are the people who wrote this blog posting? What is APNIC? APNIC is the regional Internet address registry for the Asia–Pacific region – thus "AP". It's one of the world's five Regional Internet Registries, abbreviated RIR. So we can think of this as where the IP address assignments come from... because, well, there's where they come from. So here's what the guys in charge of the IP address space have to say as of one week ago, last Tuesday.

Since Geoff writes in the first person, it only seems right to introduce him by name as Geoff Huston. He's the Chief Scientist at APNIC, where he undertakes research on Internet infrastructure, IP technologies, and address distribution policies among other topics. He's widely regarded as the preeminent researcher on IPv4 exhaustion, and is routinely referenced by international agencies and frequently quoted by the media. So Geoff is the guy we want to hear from about this. Here's what he has to say:

I wrote an article in May 2022, asking 'Are we there yet?' about the transition to IPv6. At the time, I concluded the article on an optimistic note, observing that we may not be ending the transition just yet, but we are closing in. I thought at the time that we wouldn't reach the end of this transition to IPv6 with a bang but with a whimper. A couple of years later, I'd like to revise these conclusions with some different thoughts about where we are heading and why.

The state of the transition to IPv6 within the public Internet continues to confound us. RFC 2460, the first complete specification of the IPv6 protocol, was published in December 1998, over 25 years ago. The entire point of IPv6 was to specify a successor protocol to IPv4 due to the prospect of depleting the IPv4 address pool. We depleted the pool of available IPv4 addresses more than a decade ago, yet the Internet is largely sustained through the use of IPv4. The transition to IPv6 has been underway for 25 years, and while the exhaustion of IPv4 addresses should have created a sense of urgency, we've been living with it for so long that we've become desensitized to the issue. It's probably time to ask the question again: How much longer is this transition to IPv6 going to take?

At APNIC Labs, we've been measuring the uptake of IPv6 for more than a decade now. We use a measurement approach that looks at the network from the perspective of the Internet's user base. What we measure is the proportion of users who can reach a published service when the only means to do so is by using IPv6. The data is gathered using a measurement script embedded in an online ad, and the ad placements are configured to sample a diverse collection of end users on an ongoing basis. The IPv6 adoption report, showing our measurements of

IPv6 adoption across the Internet's user base from 2014 to the present, is shown in the Figure. [I placed it at the beginning of this topic. Geoff continues:]

On the one hand, the figure is one of those classic 'up and to the right' Internet curves that show continual growth in the adoption of IPv6. The problem is in the values in the Y-axis scale. The issue here is that in 2024 we are only at a level where slightly more than one-third of the Internet's user base can access an IPv6-only service. Everyone else is still on an IPv4-only Internet.

I'll interrupt to note that their approach is clever. They have scattered ads around the Internet as a means of running a bit of their own script in the user's browser. The script probably queries two servers, one using IPv4 addressing and another using IPv6. And presumably the visitors whose browsers pull these ads and run this script are widely diverse. Anyway, Geoff continues:

This seems to be a completely anomalous situation. It's been over a decade since the supply of 'new' IPv4 addresses has been exhausted, and the Internet has not only been running on empty but also being tasked to span an ever-increasing collection of connected devices without collapsing. In late 2024 it's variously estimated that some 20B devices use the Internet, yet the Internet's IPv4 routing table only encompasses some 3.03B unique IPv4 addresses.

Note that the reason for the disparity between the total number of addresses in 32 bits, which is nearly 4.3 billion and the Internet's current routing table at 3.03 billion is management overhead and the fact that network allocations always leave a bit of headroom. So here comes the "purist" argument. Geoff writes:

The original 'end-to-end' architecture of the Internet assumed that every device was uniquely addressed with its own IP address, yet the Internet is now sharing each individual IPv4 address across an average of seven devices, and apparently, it all seems to be working! If end-to-end was the sustaining principle of the Internet architecture then as far as the users of IPv4-based access and services are concerned, then it's all over!

IPv6 was meant to address these issues, and the 128-bit wide address fields in the protocol have sufficient address space to allow every connected device to use its own unique address. The design of IPv6 was intentionally very conservative. At a basic level, IPv6 is simply 'IPv4 with bigger addresses'. There are also some changes to fragmentation controls, changes to the Address Acquisition Protocols (ARP vs Neighbor Discovery), and changes to the IP Options fields, but the upper-level transport protocols are unchanged. IPv6 was intended to be a largely invisible change to a single level in the protocol stack, and definitely not intended to be a massive shift to an entirely novel networking paradigm.

In the sense of representing a very modest incremental change to IPv4, the IPv6 design achieved its objective, but in so doing it necessarily provided little in the way of any marginal improvement in protocol use and performance.

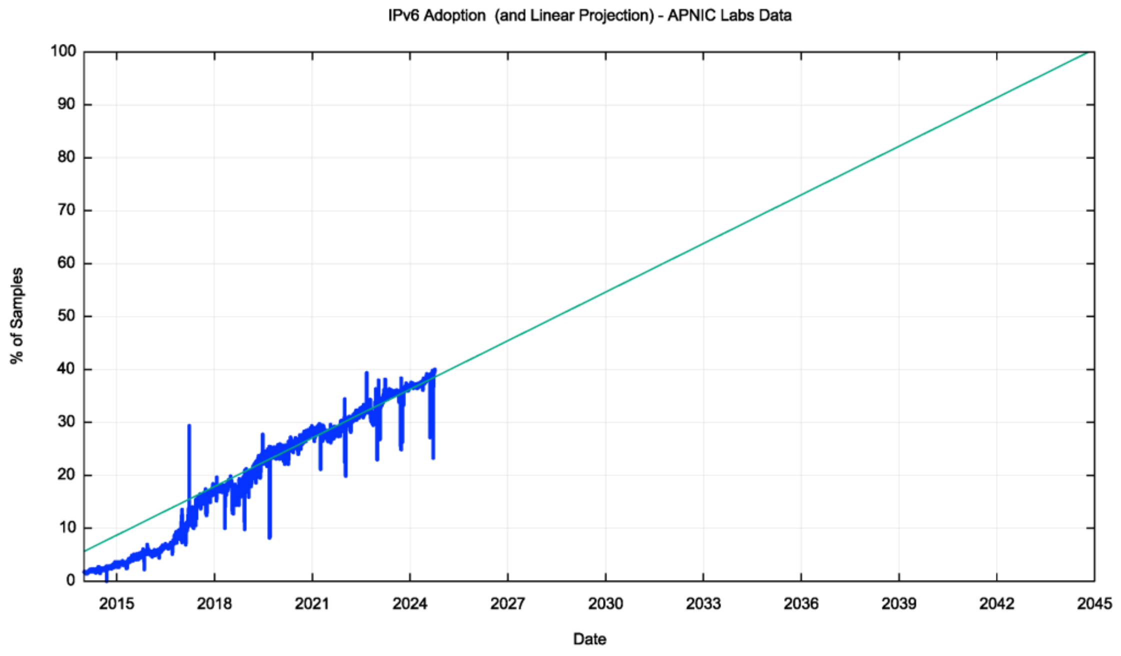
IPv6 was no faster, no more versatile, no more secure than IPv4. The major benefit of IPv6 was to mitigate the future risk of IPv4 pool depletion.

In most markets, including the Internet, future risks are often heavily discounted. The result is that the level of motivation to undertake this transition is highly variable given that the *expenditure to deploy this second protocol does not realize tangible benefits in terms of lower cost, greater revenue or greater market share. In a networking context, where market-based coordination of individual actions is essential, this level of diversity of views on the value of running a dual-stack network leads to reluctance on the part of individual actors and sluggish progress of the common outcome of the transition. As a result, there is no common sense of urgency.*

When we refers to a "dual stack" he means using a machine that simultaneously runs both IPv4 and IPv6 protocols. Everyone running modern desktop machines today is probably running a dual stack. If I open a command prompt on the Windows 7 machine that's in front of me and enter the command "ipconfig" I see that my machine has both IPv4 and IPv6 addresses as well as both IPv4 and IPv6 default gateways. So that means my ISP, COX Cable is providing both IPv4 and IPv6 support which is flowing through my cable modem into my pfSense firewall router which is distributing both flavors of the Internet to all of the machines in my local network. Thus, dual stack.

So Geoff's point here is that the only significant thing IPv6 was **intended** to provide, aside from minor fixes around the edges, was significantly greater addressing space. And, inertia being what it is, that wasn't sufficient to drive its adoption. My guess is that what we're seeing is what I would call "adoption by attrition". The same way we're getting Windows 11 when Windows 10 machines die and it's impossible to get another Windows 10 machine – in other words, for reasons other than desire or demand. Geoff says:

To illustrate this, we can look at the time series shown in the figure and ask the question: 'If the growth trend of IPv6 adoption continues at its current rate, how long will it take for every device to be IPv6 capable?' This is the same as looking at a linear trend line placed over the data series used in the Figure, looking for the date when this trend line reaches 100%. Using a least-squares best fit for this data set from January 2020 to the present day, and using a linear trend line, we come up with Figure 2.



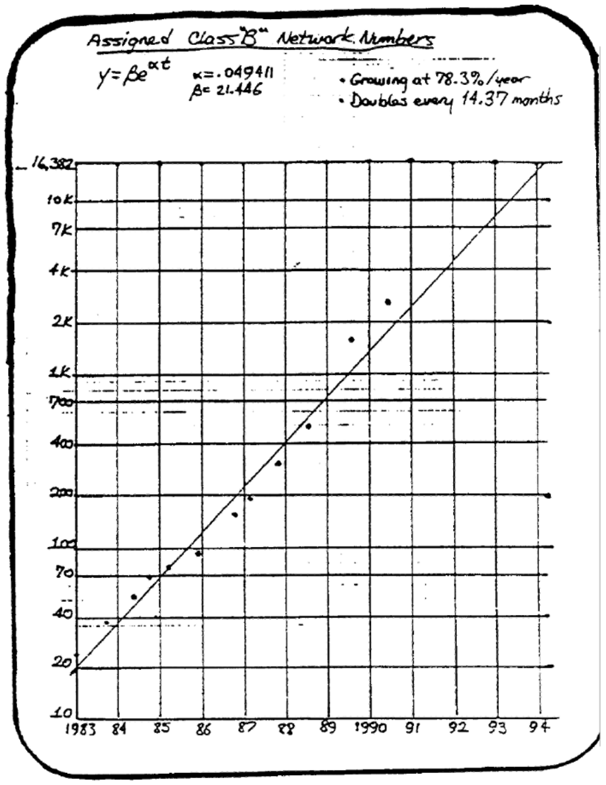
This exercise predicts that we'll see completion of this transition in late 2045, or some 20 years into the future. It must be noted that there is no deep modelling of the actions of various service providers, consumers, and network entities behind this prediction. The only assumption that drives this prediction is that the forces that shaped the immediate recent past are unaltered when looking into the future. In other words, this exercise simply assumes that 'tomorrow is going to be a lot like today.'

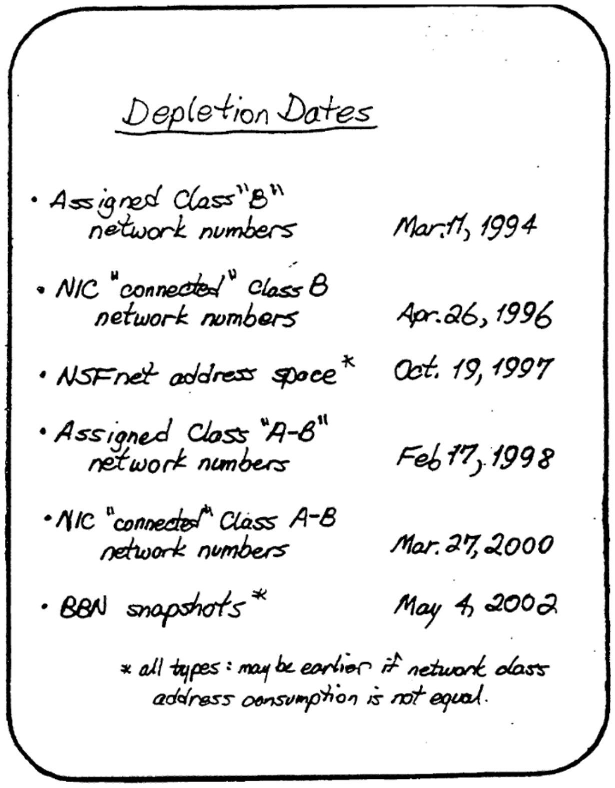
The projected date in the second Figure is less of a concern than the observation that this model predicts a continuation of this transition for a further ***two decades****. If the goal of IPv6 was to restore a unified address system for all Internet-connected devices, but this model of unique addressing is delayed for 30 years (from around 2015 to 2045), it raises questions about the relevance and value of such a framework in the first place! If we can operate a fully functional Internet without such a coherent end-device address architecture for three decades, then why would we feel the need to restore address coherence at some point in the future? What's the point of IPv6 if it's not address coherence?*

Something has gone very wrong with this IPv6 transition, and that's what I'd like to examine in this article.

So now let's look back a bit to see what these Internet pioneers saw during the 1990's:

By 1990 it was clear that IP had a problem. It was still a tiny Internet at the time, but the growth patterns were exponential, doubling in size every 12 months. We were stressing out the pool of Class B IPv4 addresses and in the absence of any corrective measures this address pool would be fully depleted in 1994 (Figure 3).





Depletion Dates

• Assigned Class "B" network numbers	Mar.11, 1994
• NIC "connected" Class B network numbers	Apr.26, 1996
• NSFnet address space*	Oct. 19, 1997
• Assigned Class "A-B" network numbers	Feb 17, 1998
• NIC "connected" Class A-B network numbers	Mar. 27, 2000
• BBN snapshots*	May 4, 2002

* all types: may be earlier if network class address consumption is not equal.

"Internet Growth" Frank Solensky, Proc. IETF, Aug 1990

Geoff explains that the IETF was panicking in the 1990's because the Internet's original design was destined to collapse. He explained:

There was a collection of short- medium- and longer-term responses that were adopted in the IETF to address the problem. In the short term, the IETF dispensed with the class-based IPv4 address plan and instead adopted a variably sized address prefix model. Routing protocols, including BGP, were quickly modified to support these classless address prefixes. Variably sized address prefixes added additional burdens to the address allocation process, and in the medium term, the Internet community adopted the organizational measure of the Regional Internet Registry (RIR) structure to allow each region to resource the increasingly detailed operation of address allocation and registry functions for their region.

These measures increased the specificity of address allocations and provided the allocation process with a more exact alignment to determine adequate resource allocations that permitted a more diligent application of relatively conservative address allocation practices. These measures realized a significant increase in address utilization efficiency. The concept of 'address sharing' using Network Address Translation (NATs) also gained some traction in the ISP world. Not only did this dramatically simplify the address administration processes in ISPs, but NATs also played a major role in reducing the pressures on overall address consumption.

The adoption of these measures across the early 1990s pushed a two-year imminent crisis into a more manageable decade-long scenario of depletion. However, they were not considered to be a stable long-term response. It was thought at the time that an effective long-term response really needed to extend the 32-bt address field used in IPv4. At the time the transition from mainframe to laptop was well underway in the computing work and the prospect of further reductions in size and expansion of deployment in smaller embedded devices was clear at the time. An address space of 4B was just not large enough for what was likely to occur in the coming years in the computing world.

We should remember that the original Internet divided networks on byte boundaries. IPv4 addresses have four 8-bit bytes. So a class A network had its most significant byte as the network ID and the remaining 24 bits as the host machine within the network. Class B networks had two bytes for network ID and 16 bits for individual host machines within each network. And Class C networks had three bytes for network ID and one byte for host machines within the network.

The problem that Geoff is referring to is that this created massively granular network allocations. The adoption of so-called "Classless Inter-Domain Routing" (CIDR), where the division between the network ID on the left and the host machine within that network on the right could fall on any boundary within the IPv4 32 bits, massively increased the load on the Internet's routers and routing tables, but in return it meant that the size of individual network allocations could much more closely track the number of host machines within that network. So that was a win.

But Geoff mentioned the emergence of NAT routing, and a fascination of mine has always been "what's wrong with NAT?" It works. Here's what Geoff has to say about NAT:

At this point, there was no choice for the Internet, and to sustain growth in the IPv4 network while we were waiting for IPv6 to gather momentum we turned to NATs. NATs were a *challenging subject for the IETF. The entire concept of coherent end-to-end communications was to eschew active middleware in the network, such as NATs. NATs created a point of disruption in this model, creating a critical dependency upon network elements. They removed elements of network flexibility from the network and at the same time reduced the set of transport options to TCP and UDP.*

The IETF resisted any efforts to standardize the behavior of NATs, fearing perhaps that standard specifications of NAT behavior would bestow legitimacy on the use of NATs, an outcome that several IETF participants were very keen to avoid. This aversion did not reduce the level of impetus behind NAT deployment. We had run out of IPv4 addresses and IPv6 was still a distant prospect, so NATs were the most convenient solution. What this action did achieve was to create a large variance of NAT behaviors in various implementations, particularly concerning UDP behaviors. This has exacted a cost in software complexity where an application needs to dynamically discover the type of NAT (or NATs) in the network path if it wants to perform anything more complex than a simple two-party TCP connection.

Despite these issues, NATs were a low-friction response to IPv4 address depletion where individual deployment could be undertaken without incurring external dependencies. On the other hand, the deployment of IPv6 was dependent on other networks and servers also deploying IPv6. NATs made highly efficient use of address space for clients, as not only could a NAT use the 16-bit source port field, but by time-sharing the NAT binding, NATs achieved an even greater level of address efficiency. A major reason why we've been able to sustain an Internet with tens of billions of connected devices is the widespread use of NATs.

So that's over on the client side of connections. The solutions that the industry has evolved over on the server side is something we've covered previously. Geoff writes:

Server architectures were evolving as well. With the introduction of Transport Layer Security (TLS) in web servers, a step was added during TLS session establishment where the client informs the server of the service name it intends to connect to. Not only did this allow TLS to validate the authenticity of the service point, but this also allowed a server platform to host an extremely large collection of services from a single platform (and a single platform IP address) and perform individual service selection via this TLS Server Name Indication (SNI).

The result is that server platforms perform service selection by name-based distinguishers (DNS names) in the session handshake, allowing a single server platform to serve large numbers of individual servers. The implications of the widespread use of NATs for clients and the use of server sharing in service platforms have taken the pressure off the entire IPv4 address environment.

A perfect example of this is at GRC. I don't have endless IPs from Level 3, but through the years the range of services I've wanted to offer has grown. Thanks to server name indication, I have (I just checked) 13 different web services sharing a single IP address. DNS points 13 different domains to a single IP and any web browser that wishes to connect indicates the machine it's looking for during that connection handshake.

In the interest of time, I've deliberately skipped over a lot of Geoff's truly interesting discussion. But he eventually gets to examining the question: "How much longer?", writing:

Now that we are somewhere in the middle of this transition, the question becomes: How much longer is this transition going to take?

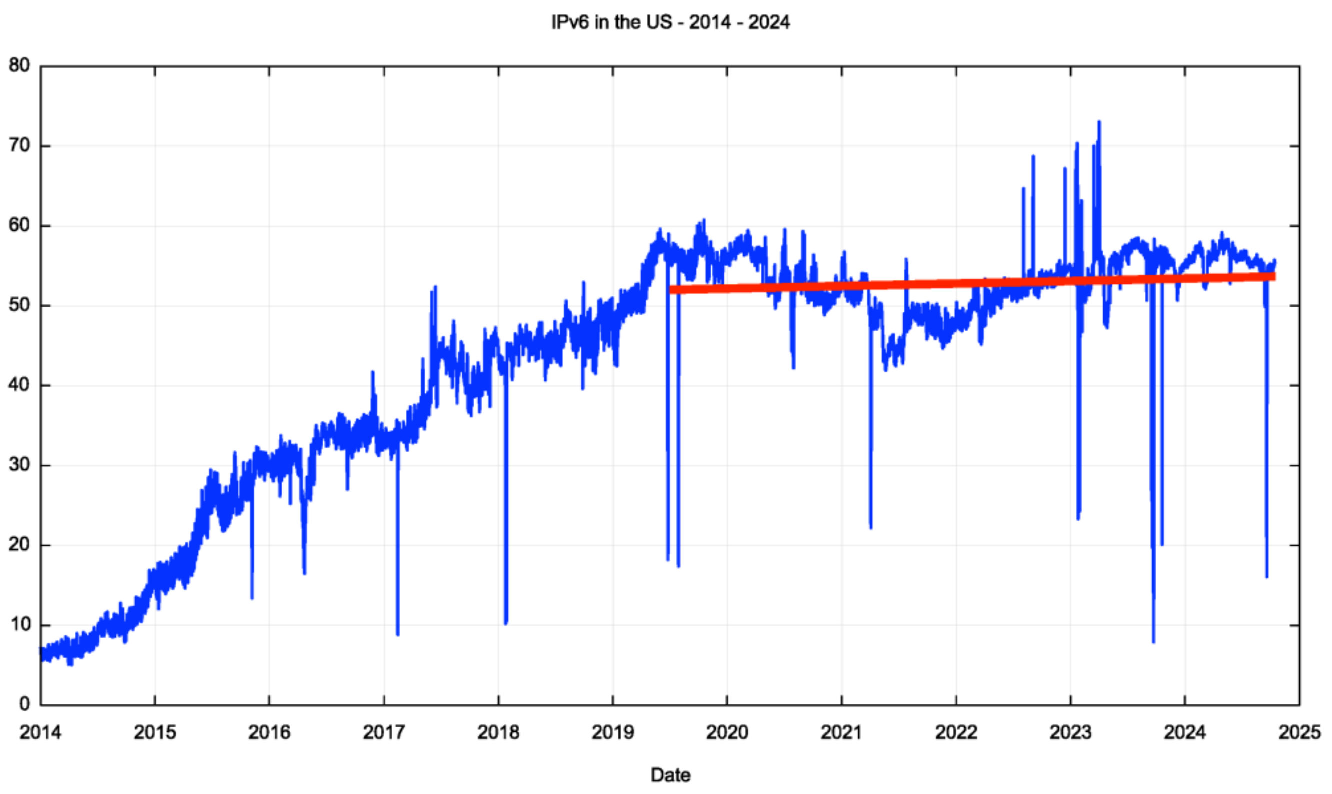
This seems like a simple question, but it does need a little more explanation. What is the 'endpoint' when we can declare this transition to be 'complete'? Is it a time when there is no more IPv4-based traffic on the Internet? Is it a time when there is no requirement for IPv4 in public services on the Internet? Or do we mean the point when IPv6-only services are viable? Or perhaps we should look at the market for IPv4 addresses and define the endpoint of this transition at the time when the price of IPv4 addresses completely collapses?

Perhaps we should take a more pragmatic approach and, instead of defining completion as the total elimination of IPv4, we could consider it complete when IPv4 is no longer necessary. This would imply that when a service provider can operate a viable Internet service using only IPv6 and having no supported IPv4 access mechanisms at all, then we would've completed this transition.

What does this imply? Certainly, the ISP needs to provide IPv6. But, as well, all the connected edge networks and the hosts in these networks also need to support IPv6. After all, the ISP has no IPv4 services at this point of completion of the transition. It also implies that all the services used by the clients of this ISP must be accessible over IPv6. Yes, this includes all the popular cloud services and cloud platforms, all the content streamers and all the content distribution platforms. It also includes specialized platforms such as Slack, Xero, Atlassian, and similar.

The data published on Internet Society's Pulse, reports that only 47% of the top 1,000 websites are reachable over IPv6, and clearly a lot of service platforms have work to do, and this will take more time.

When we look at the IPv6 adoption data for the US there is another curious anomaly:



This is so interesting! I know that we've previously observed that much of the IPv6 growth has been elsewhere in the world. Developing nations, for example, which are just obtaining Internet access are naturally acquiring IPv6 access since they have no inertia and IPv6 is certainly available.

But where we previously observed a surprisingly straight upward moving line of total global adoption, the chart showing only US-based adoption is an entirely different animal. For the past six years, since around the start of 2019 and through 2024, the United States IPv6 has been flat, showing no growth! None. Anomaly, indeed.

Geoff draws the really interesting conclusion that the services and the service model of the Internet are changing and that, in a very real sense, DNS has evolved into our routing protocol. He explains:

It's domain names that operate as service identifiers, and its domain names that underpin the user tests of authenticity of the online service. It's the DNS that increasingly is used to steer users to the 'best' service delivery point for content or service. From this perspective addresses, IPv4 or IPv6, are not the critical resource for a service and its users. The 'currency' of this form of CDN network is names.

So where are we in 2024? Today's public Internet is largely a service delivery network using CDNs to push content and service as close to the user as possible. The multiplexing of multiple services onto underlying service platforms is an application-level function tied largely to TLS and service selection using the SNI field of the TLS handshake. We use DNS for 'closest match' service platform selection, aiming for CDNs to connect directly to the access networks ***where*** *users are located. This results in a CDN's routing table with an average path length designed to converge to just 1! From this aspect the DNS has supplanted the role of routing! While we don't route 'names' on today's Internet, it functions in a way that is largely equivalent to a named data network. – In other words, no longer addresses but names.*

There are a few additional implications of this architectural change for the Internet. TLS, like it or not (and there is much to criticize about the robustness of TLS), is the sole underpinning of authenticity in the Internet. DNSSEC has not gathered much momentum to date. DNSSEC is too complex, too fragile and just too slow to use for most services and their users. Some value its benefits highly enough that they are prepared to live with its shortcomings, but that's not the case for most name holders and most users, and no amount of passionate exhortations about DNSSEC will change this! It supports the view that it's not the mapping of a name to an IP address that's critical. What is critical is that the named service can demonstrate that it is operated by the owner of the name. Secondly, the Routing PKI, the framework for securing information being passed in the BGP routing protocol, is really not all that useful in a network where there is no routing!

The implication of these observations is that the transition to IPv6 is progressing very slowly not because this industry is chronically short-sighted. There is something else going on here. IPv6 alone is not critical to a large set of end-user service delivery environments. We've been able to take a 1980s address-based architecture and scale it more than a billion-fold by altering the core reliance on distinguisher tokens from addresses to names. There was no real lasting benefit in trying to leap across to just another 1980s address-based architecture – meaning IPv6 – with only a few annoying stupid differences, apart from longer addresses!

So to give this something of a summary, what's happened is that the Internet has become the WebNet. It is mostly all about the World Wide Web. And even where it isn't, most endpoints are still being secured by the web's TLS.

What's happened is that both ends of the web have independently solved their IPv4 resource depletion problem. Over on the client end we have NAT routing which, as I noted earlier, effectively borrows excess bits from the 16-bits of port addressing to allow many client side devices to share a single public 32-bit IPv4 address. And over on the server side we have Server Name Indication – SNI – which allows GRC, for example, to host 13 differently **named** services from a single IP address. The key, and this is the point that I think Geoff brilliantly observes, we are now using names rather than addresses to access the services we need.

And to see that, today, fewer than half of the top 1000 websites are reachable over IPv6 suggests that the majority also feel very little pressure to invest in something that will literally make no difference in the services they deliver.

And finally, even before seeing that the U.S. adoption of IPv6 has been completely flat and static for the past five years we know that no straight line continues straight out to the end. That line was a percentage of IPv6 adoption, so that rate of adoption is absolutely going to slow down, and probably not long from now. Nothing gets to 100%. So my guess is that it will begin flattening out and will asymptotically approach 90% over a great many more decades. And that's fine too since I think it's clear that IPv4 will never die.