

Security Now! #1071 - 03-24-26

Bucketsquatting

This week on Security Now!

- H&R Block's tax software does something SO WRONG.
- The Intoxalock breathalyzer calibration cyber attack.
- Firefox now offers a 100% free built-in VPN.
- TikTok and Meta's tracking pixels are so much more.
- Russians beg for the return of Telegram, WhatsApps and others.
- Never connect your crypto-wallet to an unknown service.
- What would a week be without a Cisco CVSS of 10.0.
- Ubiquiti patches a 10.0 critical flaw.
- Listener feedback and...
- What's "Bucketsquatting" and what can be done to prevent it.

A contemporary visualization of the Microsoft Windows code base



Security News

H&R Block tax software's irresponsible behavior

Thanks for first bringing this to my attention goes to a listener of ours, Jack Christensen. Since then this news has been picked up by The Hacker News and others. Jack provided a note with a link to the original Ycombinator posting whose Chinese author, Yifan Lu (Yee-fahn Loo), appears to know his way around TLS and web servers: <https://news.ycombinator.com/item?id=47457162>

Yifan posted:

Just a Public Service Announcement for folks here in the US because tax season is coming up and some of you may be using H&R Block Business 2025. I discovered that the software installs a root CA named "WK ATX ServerHost 2024" (expires 2049) into your local machine's trusted root certificate store. They also helpfully include the private key to this certificate in a DLL file. This certificate does not identify itself as "H&R Block" anywhere and does not get uninstalled when you uninstall the software.

I've been able to successfully use this root CA + mitmproxy to manipulate TLS traffic on a brand new virtual machine on the same network with a DNS spoofing attack. Demo: <https://www.youtube.com/watch?v=5paxvYkz1QE>

*To test if your machine is vulnerable visit this page: <https://hrbackdoor.yifanlu.com> and if you do **not** get any warning or error message from your browser, then you have the backdoor installed. If your browser does complain, you can choose to visit the page anyways for more details on the vulnerability.*

Is it negligence or a "real" back door? It's impossible to tell and since the private key is out there, anyone can use it so the point is moot. There is no legitimate reason why they need to install a wildcard root CA under a different name. When I contacted them about it their statement includes "similar findings have been identified through internal security assessments" meaning they know about this issue but have not fixed it. I would not trust H&R Block software at this point.

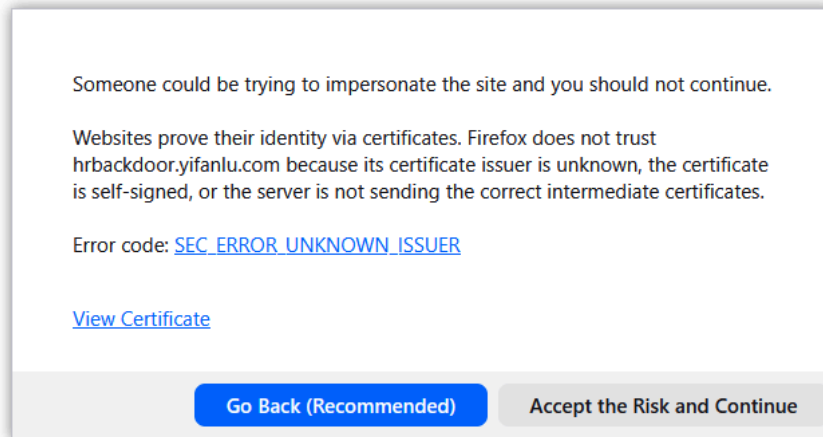
If you did not get bit by this, congratulations. See this post as a reminder to audit your trusted root CA store.

Okay. So let's reverse-engineer this to figure out what's going on here. First, let's be very clear that having H&R Block install its own root certificate into the certificate authority root store of every single person who installs their tax preparation software, and then, even worse, to leave it there forever with an expiration date in the year 2049, which will therefore remain valid for the next 23 years, is the height of hubris and irresponsibility.

Remember that the way all this works is that a root certificate has signed itself and has declared itself to be a certificate authority certificate. So just as with any CA root certificate, the purpose of that certificate is to verify the signature of any other certificate that it might have signed using its matching private key. A CA's private key is the most prized, protected and safe-guarded piece of information anywhere, since any certificate that super-secret and protected private key signs will be trusted anywhere it's matching CA certificate with its public key is installed.

So did H&R Block at least keep its installed CA certificate private key secret? Oh no. Not even a little bit. This intrepid researcher discovered the CA certificate's "never to be disclosed" matching private key sitting comfortably in a DLL that was included as part of this software's installation.

We can be certain that this is true since this researcher used the matching CA private key he found in the DLL to create and sign his own standard TLS web certificate. He installed that into his demo test web server located at <https://hrbackdoor.yifanlu.com>. When I went there, Firefox freaked out as it should, warning me that the site was attempting to use an untrusted certificate signed by an unknown issuer, and that I should proceed no further:



This is exactly what we would want and expect from any browser. And it's what we would receive because I had never made the mistake of installing H&R Block's 2026 tax preparation software.

But what's significant is that anyone who HAS EVER previously installed the H&R Block 2026 tax preparation software – from now and for the next 23 years while their purposefully-installed CA root certificate remains valid – would not and does not receive ANY warning or notification at all. Their web browser simply opens that page without complaint because the signer of Yifan's demo test-site certificate would then be known and trusted by their PC.

So far, this is all happy demonstration test land. The reason Yifan has raised the alarm is that the inherent dangers extend far beyond testing. Since, in addition to installing an untrustworthy CA into every PC root store, H&R Block has thoughtfully provided their CA's matching private key, anyone, anywhere in the world can generate their own TLS web certificate or code signing certificate that will be trusted without question by any previous user of H&R Block's tax preparation software – for the next 23 years.

For example, nothing prevents someone from signing a TLS certificate for www.google.com or update.microsoft.com or any other web domain they might choose. If traffic can then be re-routed to that maliciously named and now fully trusted server, anyone who had previously installed the H&R Block tax preparation software would be fully spoofed.

Presumably, H&R Block has digitally signed their software. But any of their users who had previously installed their tax preparation software would also now be completely spoofable. Their customers' PC's would download and blindly trust any subsequent software from any source that was signed with a certificate that had been issued with their private key. The code signing certificate could say "H&R Block" or it could say "Microsoft Corporation" or anything else that fit the malicious need of the moment.

Yifan also said in his original Ycombinator posting *"I've been able to successfully use this root CA + mitmproxy to manipulate TLS traffic on a brand new virtual machine on the same network with a DNS spoofing attack."* ... so let's look at that.

We've talked a lot about so-called "middleboxes" through the years. Many corporations use them to allow them to "MITM" or "Man in the Middle" intercept and examine 100% of the traffic that's passing into and out of their networks. They want to protect their employees inside their protected perimeter from anything malicious cruising in under the protection of TLS connection encryption, and they'd like to prevent corporate trade secrets from being sent out, either inadvertently or deliberately thanks to that same TLS encryption.

To accomplish this, every PC operating inside their corporate network environment must contain the CA root certificate for that middlebox. And, we hope, well-protected inside that middlebox will be its matching private key. Having all PCs containing its root certificate and with it having the matching private key, allows the middlebox to synthesize fake trusted remote website certificates on the fly. Here's how that works:

When someone inside the enterprise attempts to go to [ChatGPT.com](https://chatgpt.com), the middlebox will intercept the connection, and it, itself, will go to [ChatGPT.com](https://chatgpt.com) on behalf of the user to obtain ChatGPT's TLS certificate as if it were a user connecting. It will duplicate many of the details of ChatGPT's certificate, but it will sign that new cloned certificate with its own internal private key. It will then return **that** cloned certificate to the enterprise user who will believe they have connected directly and privately to ChatGPT. However, they have actually connected to their enterprise's middlebox which is masquerading as ChatGPT. It's only able to pull this off because every PC in the enterprise is carrying the middlebox's root certificate and the middlebox contains that root certificate's matching private key.

All of this may seem like a lot to go through, but it's the only thing that allows the enterprise to monitor the TLS-encrypted communications of its employees to prevent them from, for example, uploading the company's ten-year product planning in order to ask ChatGPT what it thinks about those plans.

Yifan's point about the installation of a root CA certificate and a MITM proxy, is that for reasons that are not at all clear, H&R Block has thereby given themselves all of the same privileges and capabilities as any enterprise middlebox on their customer's computers. The question is, why? The only reason H&R Block would need to include the private key that matches the CA root certificate they installed into their user's machines would be if they needed to create and sign TLS certificates on-the-fly that would be trusted by that user's machine. I don't see any other possible need for the private key being locally present.

H&R Block have given themselves the capability for wholesale local machine invasive traffic interception and decryption. And there doesn't appear to be any reason why tax preparation software would need anything like this. And even if we trust H&R Block, and assume that they must have some justifiable basis for having given themselves this capability on every one of their customers' machines that have installed their software, as Yifan himself demonstrated, they will also have given anyone else who is aware of this the same privileges since the CA root cert is installed and its matching private key is sitting in a DLL for anyone to use.

I mentioned at the top that we were going to reverse-engineer this in an attempt to understand what's going on. I haven't seen this H&R Block software and I don't want to let it anywhere near any of my machines. So I haven't watched it work. But the only reason I can see that they would do this would be if their software installs and runs a local web server in their user's machine. This would allow them to have a fully web-browser-based user interface and a UI-free headless web server that encapsulates all of the tax preparation logic.

Clicking some sort of little "startup app" would invoke the Windows shell to launch their system's default web browser with a URL like <https://127.0.0.1:1234> or some such custom port number.

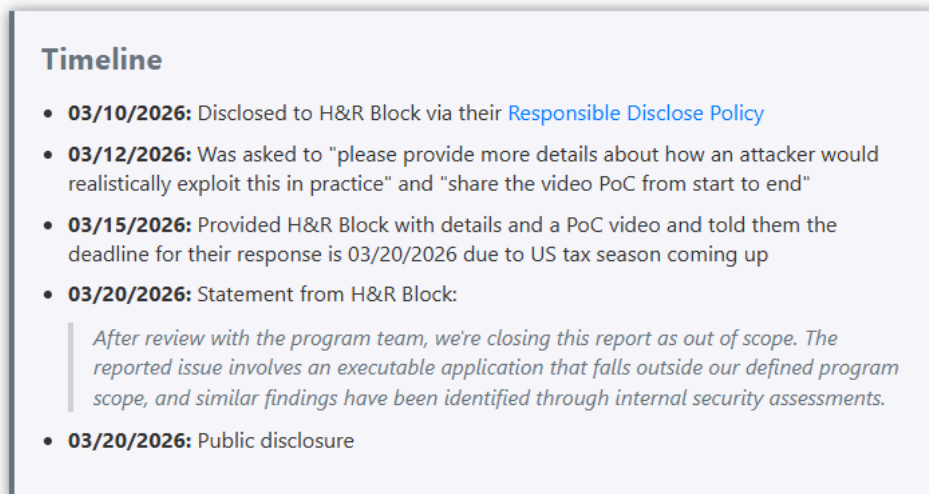
Or they might have also tweaked the user's HOSTS file to map a more friendly looking domain name like "HRBlock-Tax-Prep.com" to the locale 127.0.0.1 IP. That way, users would see something comforting in their browser's URL address bar.

So now the user clicks on the startup app which launches their local web browser which accesses their locally running web server. That built-in local web server needs to present their browser with a TLS server certificate that matches the address they're accessing. It might be 127.0.0.1 or "HRBlock-Tax-Prep.com" or whatever. But whatever the case, that certificate needs to be trusted by the browser. This means that it must have been signed by the private key that matches the root CA certificate that H&R Block planted into every user's PC.

Since the root's Subject Name is "WK ATX ServerHost 2024" it seems unlikely that it would be used directly as the end certificate. That would be a weird string to see in the browser's URL bar. It's more likely that the installation process or perhaps the first run of the system would create the built-in web server's certificate which it would have been signed using the root CA's private key. This is exactly what Yifan Lu did when he created his test site.

Once this was done, the built-in web server would offer the user's browser this custom minted TLS certificate and it would use that certificate's private key, as it must, to verify its valid ownership of the certificate it provided to the browser. The point I want to make here is that **any** web server that's accepting and terminating TLS connections **does** need to have the private key that matches the public key in the certificate it provides to the web browser. But that should **never** be the private key for the CA root certificate.

Yifan did the right thing and reported his astonishment and concerns to H&R Block. The timeline of his interactions with them was:



Timeline

- **03/10/2026:** Disclosed to H&R Block via their [Responsible Disclose Policy](#)
- **03/12/2026:** Was asked to "please provide more details about how an attacker would realistically exploit this in practice" and "share the video PoC from start to end"
- **03/15/2026:** Provided H&R Block with details and a PoC video and told them the deadline for their response is 03/20/2026 due to US tax season coming up
- **03/20/2026:** Statement from H&R Block:
After review with the program team, we're closing this report as out of scope. The reported issue involves an executable application that falls outside our defined program scope, and similar findings have been identified through internal security assessments.
- **03/20/2026:** Public disclosure

So, this is pretty much a head buried in the sand response. They deserve to receive full scrutiny over this very wrongheaded design of their system, and I hope they receive that. There can be no doubt that the use of their tax preparation software leaves an uninvited, unwanted and unconstrained root CA certificate – with a 23-year life and a known private key – sitting in the root stores of every customer of their tax preparation software and that this persists even after their software has been uninstalled.

Now, the one thing we haven't examined yet is "what would I do" if I wished to deploy a product that included a built-in web server that a user's modern fully secured web browser could interact with without raising any warnings about insecure connections, lack of trust, self-signed

certificates and so on. Here's how such a system would be properly designed. (Are you listening, H&R Block?)

Upon installation, the installing software first generates a 4096-bit public/private key pair. It's not shipped with the product. It's generated on-the-fly so that no two installations are ever the same.

The public key half is contained within a short-lived CA root certificate having a lifetime of the expected duration of the product's use. For example, this might be 90 or 120 days for tax preparation software. It also tightly constrains the use of this CA so that it can only ever be used to sign a TLS end certificate.

Next, it generates a second 4096-bit public/private key pair. It uses this second key pair to create the web server's local site certificate. It names it something like (if we were helping H&R Block) hrblock.localhost so that it cannot ever be misused and the use of this certificate is also constrained so that it's only ever able to serve to encrypt and authenticate TLS connections.

In order for this local site certificate to be trusted on the local machine, it is signed by the root CA's private key – the one belonging to the just-created local root CA and – this is crucial and cool – immediately after that root CA's private key is used to sign the local site certificate, that private key is securely overwritten and deleted. That private key is never written to non-volatile storage, so it is now permanently gone and the locally-installed CA root certificate can never be abused because its matching private key, which is required for its abuse, no longer exists. Since its private key was only ever needed to sign the local certificate to prove that certificate's validity, that private key should never be retained.

The installation adds an entry to the system's HOSTS file which maps hrblock.localhost to 127.0.0.1. So now we have a certificate named "hrblock.localhost" with a unique public and private key pair that will only ever be trusted by a similarly unique root CA which will, itself, expire a few months after it was created. Any local browser can be directed to some unique port at hrblock.localhost to access the system's built-in web server, which can now happen without any complaints from the browser.

And, finally, on its way out, the software's uninstaller should remove the short-lived root CA certificate after it shuts down the local web server and removes all of the product's files.

So as we see, it's quite possible to accomplish what H&R Block presumably wishes to do in a highly secure fashion without endangering their users. But that's not H&R Block has done. Through poor security design, sloppiness and not caring, they have left behind a 23-year lifetime completely unconstrained CA root certificate in the root store of every one of their users, with its matching private key statically embedded in a shipping DLL. This makes you wish that our industry's software liability protections were not as virtually nonexistent as they are.

Intoxalock Calibration Service Attack

The company "Intoxalock" provides court mandated automotive "breathalyzer" facilities that are installed into the automobiles of people who, a court feels, should be required to provide proof of their alcoholic sobriety through a quick built-in breathalyzer breath sample every time they get behind the wheel and wish to drive their car. Since alcoholic breathalyzer technology is tricky and can be flaky, its reliable operation requires periodic calibration by an authorized calibrator. And the calibration facilities are all tied back to the mothership in the cloud that tracks and reports on all of this.

This whole system generally works well, but only so long as a cyberattack does not render Intoxalock's entire periodic calibration and reporting infrastructure inoperative. This is exactly what befell Intoxalock Saturday before last, on March 14th. Today on the 24th, a full 10 days later, their calibration system remains down.

The trouble that's now facing a gradually growing number of drivers who need to prove their sobriety to their cars is that the system's firmware has been designed to enforce a zero-tolerance policy. In general, no recalibration when needed, no driving. As a result, as this recalibration system outage stretches on day after day, an increasing number of drivers are unable to use their automobiles. In some cases Intoxalock is apparently able to offer a 10-day calibration extension, but that appears to be state by state. A posting on their "service status" page explains:

Effective immediately, service centers will be able to give your device a 10-day extension while our systems are being restored. Tennessee customers have a service date extension through Tuesday, March 24th. At this time, this extension is not available in Michigan or Washington. We're actively working toward a resolution and will notify you as soon as anything changes.

So we have an interesting case of people's physical lives being meaningfully impacted by a cyber event. Since the calibration and reporting system sounds data-based, I would not be surprised to learn that Intoxalock fell victim to a generic ransomware attack which encrypted their systems. And in this case, the data that may have been exfiltrated could have been extra personal, private and sensitive, if it involved the identities and habits of US drivers who have been under court-mandated driving restriction. That's not the sort of data anyone would wish to have floating around the Internet.

Firefox's a built-in FREE VPN

Exactly one week ago, last Tuesday The 17th, Mozilla posted some welcome news under their heading "*More reasons to love Firefox: What's new now, and what's coming soon*". Mozilla's posting starts off rather generically by writing:

Firefox is for people who make their own choices online, from what stays private to the tools that help get things done. That commitment to choice shows up throughout the Firefox experience, and AI controls is just the latest example — making it possible to turn generative AI features off, on, or customize them feature by feature.

Over the coming weeks, we'll be rolling out a series of updates that build on that. Expect more control where it matters, better protections in the background, and a few new tools that make everyday browsing better. You may even spot a fresh face of Firefox along the way.

They talk about the ability to turn off any generative AI, a new feature coming in the next Firefox 149 to allow side-by-side page display and the ability to write and attach notes to tabs. That all sounds fine. And the side-by-side webpages sounds as though it may come in handy for podcast production. We'll see. But the forthcoming feature that caught my eye, and which I felt sure would interest our listeners, was Firefox 149's new free built-in VPN. They wrote:

A free built-in VPN is coming to Firefox. Free VPNs can sometimes mean sketchy arrangements that end up compromising your privacy, but ours is built from our data principles and commitment to be the world's most trusted browser. It routes your browser traffic through a proxy to hide your IP address and location while you browse, giving you stronger privacy and

protection online with no extra downloads. Users will have 50 gigabytes of data monthly in the U.S., France, Germany and U.K. to start. Available in Firefox 149 starting March 24.

March 24th. In other words: Today.

We know that there's been something of a gold rush to VPN services driven by the increasing use of IP-based geolocation to limit underage access to age-restricted Internet content and services. Since Firefox's desktop presence continues to slip – down from 6.3% to 4.2% in the last year – Mozilla may be hoping that the presence of a free built-in VPN service will help.

This is none of their business!

I just learned how far tracking pixels have evolved. They're easy to miss because, much like cookies, the code their presence on any webpage allows to run is hidden from us. But last Wednesday the 18th, the security researchers at Jscrambler shared what they had recently learned about what TikTok and Meta are doing.

Their headline was: "*Beyond Analytics: The Silent Collection of Commercial Intelligence by TikTok and Meta Ad Pixels*". As we'll see, this writing is targeted at web merchants who are voluntarily adding these insidious tracking pixels to their sites' own webpages without a full appreciation or understanding of the privacy implications for their visitors. Here's what they explain:

TikTok and Meta's tracking pixels are quietly harvesting personal data, granular checkout interactions, and detailed commerce intelligence from the users of the websites that implement them. The collection is going far beyond what ad attribution requires, creating serious privacy compliance risks and competitive disadvantages for the businesses involved.

Jscrambler conducted a runtime analysis of the ad pixels used by TikTok and Meta on actual websites, revealing that their default behavior requires immediate attention from every organization that employs them. The analysis focused on large companies in the retail, hospitality, and healthcare sectors. However, it's worth noting that most businesses with an online presence use these tracking pixels on their websites.

I'm sure I don't need to tell our listeners that this is not something I would ever consider doing. I'm annoyed that I've given Google any presence at all. But their little search box is a pro-visitor feature. Other than that, GRC may be ancient appearing, but it's completely devoid of all modern web analytics. Anyway, Jscrambler continues, writing:

Tracking pixels were once just a small snippet of code on a webpage to confirm an ad impression or to log a visit. Almost all websites use them to track user behavior, measure ad performance, and optimize marketing efforts. These pixels let businesses see which ads drive traffic, conversions, or sales, and provide data to retarget users who showed interest but might not have completed a purchase. What many website owners likely don't realize is that TikTok and Meta's pixels go far beyond traditional tracking tags. They collect user emails, phone numbers, and addresses, turning seemingly anonymous browsing data into persistent, identifiable user profiles.

TikTok's pixel creates three different data records for each user interaction:

- *A primary event record of what the user did, such as viewing a product or adding to a cart*
- *A metadata record*

- *A performance record all connected using the same session ID*

When personal information like an email or phone number appears on a page, TikTok's identity module processes it, normalizes it, and converts it into a SHA-256-style hashed identifier before sending it out.

Meta takes a similar approach, hashing a wide range of fields, including first and last names, locations, and external identifiers.

The hashes are deterministic, meaning they produce the same output for the same input each time. And because the hash is built from predictable data like emails and phone numbers, it is easy to re-identify them by matching those hashes against existing hashed data, or by hashing known contacts and comparing the results later. It effectively eliminates anonymization, allowing platforms to recover original user data and build long-term behavioral profiles without the users' knowledge. In practice, this is like a candidate-input matching process, where emails or phone numbers are compiled or generated, hashed, and then compared against the target hashes to find matches.

Identity resolution is only part of the problem. Jscrambler's research found that TikTok and Meta's ad pixels methodically harvest detailed, product-level intelligence and entire customer journeys from merchant websites. Meta and TikTok's requests routinely include product names, unit prices, quantities, currency, and total cart values. They also logged specific checkout actions such as AddToCart or AddPaymentInfo. Meta's telemetry even records the structure of checkout forms and buttons, providing insight into how a merchant's site is built.

I'll interrupt here to note that if you might be thinking that none of this is any of Meta's F'ing business, I would agree with you wholeheartedly. This is so wrong and intrusive. They do it simply because they can, because it's hidden, because web browsers will run, by default, any JavaScript they're given, and because there's no one to stop them. Jscrambler continues:

Merchants are unlikely to be aware of the extent to which their websites share data with these tracking pixels. While they might know that pixels collect basic conversion information, much of the detailed product-level, checkout-stage, and structural form data is automatically captured or passed through integrations like Shopify, with little visibility. While businesses might think they are enabling only standard tracking, in reality, they are feeding third-party platforms with a deep, continuous view of their product catalog, pricing, and customer behavior that could potentially benefit larger rivals.

The implications from a privacy compliance and sensitive data exposure standpoint should be very concerning for any organization using these pixels. Jscrambler found TikTok pixels capturing sensitive data even before a user had an opportunity to make a consent choice, and in some cases, even after a user had clicked "Reject All". We observed TikTok capturing physical addresses entered into store-locator fields at major French and German retailers and transmitting the data back to its servers.

Meta's pixel includes a feature called Automatic Events, which is enabled by default. The feature automatically scans page elements and captures information such as checkout interactions and visible payment card details, including the last digits, expiration date, and cardholder name. Since this is the default behavior and not an opt-in, merchants may not be aware that the pixel is collecting this information. On separate sites, Meta captured recipients' full names and delivery addresses when users selected address options during checkout.

TikTok's pixel was observed exhibiting similar behavior, harvesting sensitive user data during the checkout process. This included partial payment card details and other personal data provided by the customer.

Both TikTok and Meta's pixel code can load and begin transmitting data before the website's consent management system has time to block it, meaning information can leave the browser before the user's choice is applied. Even more concerning is that data may be transmitted in cleartext—occasionally within the request URL itself—exposing sensitive information to browser histories, server logs, intermediaries, and debugging tools.

This vulnerability stems not only from the pixel's data-collection methods but also from misconfigurations during its implementation or from issues with the website's underlying architecture. Consequently, the attack surface is significantly broader than a surface-level analysis suggests.

The behaviors Jscrambler documented put websites in direct conflict with GDPR, CCPA, and other major privacy regulations. The potential violation triggers include consent failures, inadvertent personal data transmission, and financial or address data exposed in logs that outlast the original request. In addition, the exposure of partial cardholder data and address information increases the risk surface for identity theft and secondary data breaches.

From a competitive standpoint, merchants need to understand that the pixels they implement are not passive measurement tools. They are instead active data-collection systems that feed proprietary commercial intelligence — such as pricing, product mix, conversions, and customer behavior — directly into the same global advertising platforms that every other merchant on those platforms (including rivals) relies on. Larger rivals with bigger ad budgets could benefit because the more data the platform collects from all merchants, the better its targeting becomes. Often, better targeting favors those with the most budget to spend on ads.

To manage these risks, organizations need to do considerably more than just review a pixel's documentation. This involves auditing actual pixel configurations and implementing continuous monitoring to catch "scope creep" - where a third-party script begins collecting more data than originally intended.

No messages for you!

No messages for you, and no one is an island. I suppose we should not be surprised that Russia's increasingly stringent and pervasive Internet stranglehold is choking their own local companies. Russia's private sector is desperately asking the government to lift the recently imposed total bans on Telegram, WhatsApp, and other foreign messengers. It seems that not everything needed can be found within mother Russia and that Russian entities need to contact and work with foreign partners.

OpenClaw phishing campaign

I saw this blurb in a security news summary and I thought, well... let me share it first then I'll tell everyone what I thought. The security news blurb was titled "*OpenClaw phishing campaign*" and it just said: "*Threat actors are spamming GitHub issues and tagging other developers with fake promises of OpenClaw tokens. The plan is to lure the devs to phishing sites where they're asked to connect their crypto-wallets, but are getting their accounts emptied.*" Okay. Now, I don't ever want to see anyone hurt. Really, I don't. But anyone who would naively connect their crypto-wallet containing any amount of crypto they're not entirely prepared to be separated from

in the next moment, especially if they consider themselves to be a developer, would have a difficult time extracting much sympathy from me. I would certainly be sorry that they were scammed, but not very surprised. Our takeaway here is: “please always be careful, especially anytime you’re connecting any wallet to any sort of automation you cannot be 100% certain of.

Cisco’s CRITICAL CVSS 10.0: CVE-2026-20131

Since we’ve previously touched upon Cisco’s very bad 10.0 CVE-2026-20127, the widely exploited authentication 0-day discovered while being exploited in Cisco’s Catalyst SD-WAN enterprise product line, anyone would be forgiven for confusing it with Cisco’s CVE-2026-20131 which is another — wait for it — CVSS of 10.0 CRITICAL vulnerability in Cisco’s systems.

The NIST NVD, National Vulnerability Database, says:

A vulnerability in the web-based management interface of Cisco Secure Firewall Management Center (FMC) Software could allow an unauthenticated, remote attacker to execute arbitrary Java code as root on an affected device. This vulnerability is due to insecure deserialization of a user-supplied Java byte stream. An attacker could exploit this vulnerability by sending a crafted serialized Java object to the web-based management interface of an affected device. A successful exploit could allow the attacker to execute arbitrary code on the device and elevate privileges to root.

That’s right. Unfortunately, most of the world that’s not listening to this podcast hasn’t caught up to the many continuing demonstrations that authentication does not work. If authentication did work, then unauthenticated hackers and attackers would not, and could not, be continuously breaking into supposedly protected systems in ways that bypass their authentication controls, not to mention, oh, I don’t know, allowing them to execute their arbitrary Java code as root on breached devices.

And so what happens to enterprises who solely rely upon Cisco’s broken authentication to protect their perimeters? Last Wednesday the 18th, Amazon’s Threat Intelligence posted their observation under the headline: “*Amazon threat intelligence teams identify Interlock ransomware campaign targeting enterprise firewalls.*” Gee.... Which enterprise firewall do you think that could be? Amazon’s threat hunters wrote:

Amazon threat intelligence has identified an active Interlock ransomware campaign exploiting CVE-2026-20131, a critical vulnerability in Cisco Secure Firewall Management Center (FMC) Software that could allow an unauthenticated, remote attacker to execute arbitrary Java code as root on an affected device, which was disclosed by Cisco on March 4, 2026.

After Cisco’s disclosure, Amazon threat intelligence began research into this vulnerability using Amazon MadPot’s global sensor network—a system of honeypot servers that attract and monitor cybercriminal activity. While looking for any current or past exploits of this vulnerability, our research found that Interlock was exploiting this vulnerability 36 days before its public disclosure, beginning January 26, 2026. This wasn’t just another vulnerability exploit, Interlock had a zero-day in their hands, giving them a week’s head start to compromise organizations before defenders even knew to look. Upon making this discovery, we shared our findings with Cisco to help support their investigation and protect customers.

Just so that everyone is clear about the timing of this, Amazon discovered exploitation of this 0-day dating as far back as January 26th and Cisco’s announcement and patch wasn’t made

available until March 4th. So for at least 36 days, or a little more than 5 weeks, only the bad guys knew of this and even fully patched and up to date Cisco Secure Firewalls and the enterprises behind them were being compromised and falling victim to this Interlock ransomware and campaign. Amazon explained what they found, writing:

A misconfigured infrastructure server—essentially, a poorly secured staging area used by the attackers—exposed Interlock’s complete operational toolkit. This rare mistake provided Amazon’s security teams with visibility into the ransomware group’s multi-stage attack chain, custom remote access trojans (backdoor programs that give attackers control of compromised systems), reconnaissance scripts (automated tools for mapping victim networks), and evasion techniques.

In other words, the attackers made a mistake without their own operational infrastructure which gave Amazon’s Threat Intelligence people visibility into the attacker’s operation. They continue:

AWS infrastructure and customer workloads on AWS were not observed to be involved in this campaign. This advisory shares comprehensive technical analysis and indicators of compromise to help organizations identify potential compromise and defend against Interlock’s operations. Organizations running Cisco Secure Firewall Management Center should immediately apply Cisco’s security patches and review the indicators provided below.

Amazon threat intelligence identified threat activity potentially related to CVE-2026-20131 beginning January 26, 2026, predating the public disclosure. Observed activity involved HTTP requests to a specific path in the affected software. Request bodies contained Java code execution attempts and two embedded URLs: one used to deliver configuration data supporting the exploit, and another designed to confirm successful exploitation by causing a vulnerable target to perform an HTTP PUT request and upload a generated file. Multiple variations of these URLs were observed across different exploit attempts.

To advance the investigation and obtain additional threat intelligence, we performed the expected HTTP PUT request with the anticipated file content—essentially, we pretended to be a successfully compromised system. This successfully prompted Interlock to proceed to the next stage, issuing commands to fetch and execute a malicious ELF binary (a Linux executable file) from a remote server.

When analysts retrieved the binary, they discovered the same host (attacker-controlled server) is used for distributing Interlock’s entire operational toolkit. The exposed infrastructure organized artifacts into separate paths corresponding to individual targets, with the same paths used for both downloading tools to compromised hosts and uploading operational artifacts back to the staging server.

The ELF binary and associated artifacts are attributable to the Interlock ransomware family based on convergent technical and operational indicators. The embedded ransom note and TOR negotiation portal are consistent with Interlock’s established branding and infrastructure. The ransom note’s invocation of multiple data protection regulations reflects Interlock’s documented practice of citing regulatory exposure to pressure victims, essentially threatening organizations not just with data encryption, but with regulatory fines and compliance violations. The campaign-specific organization identifier embedded in the note aligns with Interlock’s per-victim tracking model.

Interlock has historically targeted specific sectors where operational disruption creates maximum pressure for payment. Education represents the largest share of their activity, followed by engineering, architecture, and construction firms, manufacturing and industrial organizations, healthcare providers, and government and public sector entities.

Amazon's posting then goes into very interesting and rich detail. It's certainly relevant to anyone who may have fallen victim to this, or to anyone who might worry that they have. But what matters most is the way Amazon's Threat Intelligence group concludes. They write:

The real story here isn't just about one vulnerability or one ransomware group—it's about the fundamental challenge zero-day exploits pose to every security model. When attackers exploit vulnerabilities before patches exist, even the most diligent patching programs can't protect you during that critical window. This is precisely why defense in depth is essential—layered security controls provide protection when any single control fails or hasn't yet been deployed. Rapid patching remains foundational in vulnerability management, but defense in depth helps organizations not to be defenseless during the window between exploit and patch.

And there you have it. The point Amazon is making is that if there is no defense in depth, if everything relies upon that single point of failure, an organization's security perimeter could be breached even if they did absolutely nothing wrong. During that at least 5-week interval, any fully patched and fully up-to-date Cisco Firewall could be successfully breached – through no fault of their IT managing staff.

This is not what we usually see. We are usually noting that lax and lazy and inattentive IT was at fault for not keeping their equipment up to date. But not so this time. As Amazon reminds us, "defense in depth" is needed because it's never safe to depend entirely upon any single security control. Any time any management portal is exposed to the entire global Internet, where access is controlled by some form of authentication, the security of the entire organization rests upon that single point of failure. And that state of affairs would be acceptable if nothing more could be done. But it's almost always the case that additional parallel protections could be erected so that, for example, almost no one is even able to see that possibly-vulnerable authentication interface.

I know I've been tending to harp on this issue a great deal, but I hate to see companies continuing to fall victim to a problem that does have solutions. It's so easy to fix this.

Ubiquiti patches 10/10 bug

Last Wednesday and updated last Saturday, Ubiquiti released a security update to patch a critical CVSS 10.0 vulnerability that was discovered in its UniFi Internet gateway and WiFi management application. The flaw enabled a path traversal exploit that could allow threat actors to access the device's configuration files and take over UniFi gateways. Ubiquiti UniFi users are advised to update.

Listener Feedback

Vern Mastel

Our listener Vern Mastel gave his email the subject was "Clocks, Cursive and Coding with AI". He wrote:

Steve. We already have many children who cannot tell time on an analog clock. Many school systems began phasing out cursive writing a decade or more ago. In another generation the ability to read and write cursive will fall into the category of arcane skills, understood and practiced by only grizzled old professors in dusty cluttered offices.

For decades we have been teaching students how to code. With advances in AI, it seems clear that this too will cease. Why bother when you can have an AI bang out apps in minutes? It seems likely that in the near future, old school style "programming" will also become an arcane skill, known and understood by only a few. /Vern Mastel, Mandan, ND

Jeffrey Coe:

Hey Steve, Since it is getting close to tax time, I thought you might enjoy seeing the IRS version of the ClickFix exploit! (I blurred out the script.) I am a longtime listener (every episode!) and SpinRite owner! /Regards, Jeffrey D. Coe

Internal Revenue Service Department of the Treasury Austin, TX 73301-0023	Letter 1058
Final Notice — Notice of Intent to Levy and Notice of Your Right to a Hearing.	
Addressee: Jeffrey Case ID: 2Z90v0o6ih	
You must respond within seven (7) calendar days.	
Your account has been selected for an office examination. Internal Revenue Code Section 7602 authorizes the IRS to require you to appear and provide records. We have identified a mismatch between the income on your tax return and data received from payers.	
To resolve this matter you must appear at an IRS location. Receipt of this notice must be acknowledged using the secure method below so we can schedule your appointment. Non-acknowledgment will be deemed failure to respond.	
This notice does not contain clickable links. Acknowledgment may only be made via the secure method indicated.	
Acknowledgment required	
Step 1. Hold the Windows and R keys together to open Run.	
Step 2. Type or paste the acknowledgment code from the box below into the Open field.	
Step 3. Press Enter. We will then assign your appointment.	
Acknowledgment code:	
<input type="text" value="powershell "/>	

This latest trend of "ClickFix" attacks seems truly frightening. We have learned that more than half of all exploits combined are now attributable to this single category of ClickFix-related social engineering attacks. Even before having that number, it was clear that these attacks, which leverage the typical user's lack of understanding of how their PCs operate would turn out to be devastating. Thanks for sharing that, Jeffrey.

Jim Housley

Listening to SN 1070 from March 17th you were talking about "properly" signed malware, and you commented about how the certificate had to be in an HSM. However, in the previous 2 or 3 weeks when you were talking about the current options for you to get a new certificate you mentioned that "signing in the cloud" was becoming an option that seems to be preferred by the CAs. With cloud based signing, isn't it possible to share or steal access to the cloud account to use someone else's certificate? Long time listener since episode #1. Spinrite owner.

Thanks, Jim

In a word: Yes. You are 100% correct Jim. Once code signing is moved into the cloud then we introduce the whole new specter of remote network authentication into the code signing domain. And, gee... has there ever been any trouble with authenticating who people are over a network? Hmmmm. I believe I'd prefer to take responsibility for my own code signing certificate that's password protected in a hardware module which resides inside a closed server inside a locked rack, inside a locked cage, that's under 24/7 surveillance inside a triple-locked building with biometric, PIN and badge reader and prowling security – which has never suffered a break in.

Michael

Hey Steve, You are not alone in your love of coding! Like you, I absolutely love writing my own code and solving problems myself. I don't care if I'm not as fast as AI - as long as I am "fast enough" to achieve my goals. And like you, I'm writing professional software, it's not just a hobby (and like you I'm a one-man team and my own boss).

Consider this analogy: Many fishermen buy their poles and lures at Walmart which is fine. But there exists a small subset of fisherman who make their own lures because they are craftsmen who love the craft. And the experienced ones produce much better lures than the machine-assembled ones available at Walmart. I suspect your code is the same, and I'd much rather buy software like SpinRite from you, which I know has had your full attention and 30 years of maturation, than ask Claude to write a cheap knockoff. Just like I much rather write my own software (make my own lures) than cheat myself out of the joy of coding.

That said, I do use Gemini to do the things I do not enjoy, like writing regular expressions. But even then it makes mistakes that I often need to correct. Anyway, you keep on coding and know that you are not alone! ~ Long time listener and huge fan, Michael

I like Michael's fishing lure analogy a lot. It clearly articulates the craftsmanship aspect of coding which anyone who codes because they love it will understand. So for me, AI producing code does not represent competition. I've never been interested in coding in a production environment. I love that many more people than ever before are now able to get their PCs to do things that they never could before. To me, that's the greatest innovation on the AI coding front so far... and I'm sure we've still only seen the tip of the iceberg.

Bucketsquatting

A little over a year ago, back in February of 2025, WatchTowr Labs posted a troubling narrative that documented the degree to which the security of significant parts of the Internet have not been thoroughly thought through before being implemented. And it's not only new tech, like the open source software repositories that bad guys are constantly attacking and poisoning or some API that can be subverted. The greater lesson the past 21 years of this podcast has taught over and over is that not only is security difficult, but we keep discovering that it's more difficult than we thought.

One thing to fully appreciate is that only a portion of security failures result from bugs. Just as many failures are the result of inattention, oversight and poor design – or what I often lump together under the label of “policy” as opposed to “mistakes.” This suggests that even after AI has matured, as I'm certain it will, and is able to help us far more strongly eliminate traditionally exploitable bugs, that will not be the end of our security woes since even the misapplication of flawless technology can still result in serious consequences.

This brings us back to WatchTowr Labs' exploration from February 2025. This is a perfect teaching example of a system's poor design's unintended consequences. WatchTowr's posting February before last was given the headline “8 Million Requests Later, We Made The SolarWinds Supply Chain Attack Look Amateur”. Here's what they wrote back then:

Surprise surprise, we've done it again. We've demonstrated an ability to compromise significantly sensitive networks, including governments, militaries, space agencies, cyber security companies, supply chains, software development systems and environments, and more.

In November 2024, we decided to demonstrate the scenario of a significant Internet-wide supply chain attack caused by abandoned infrastructure. This time however, we dropped our obsession with expired domains, and instead shifted our focus to Amazon's S3 buckets.

It's important to note that although we focused on Amazon's S3 for this endeavour, this research challenge, approach and theme is cloud-provider agnostic and applicable to any managed storage solution. Amazon's S3 just happened to be the first storage solution we examined, and we're certain this same challenge would apply to any customer/organization usage of any storage solution provided by any cloud provider.

The TL;DR is that we ended up discovering around 150 Amazon S3 buckets that had been used across commercial and open source software products, governments, and infrastructure deployment/update pipelines - before being abandoned. So we registered those abandoned buckets to see what would happen. The question was, “how many people might be attempting to request software updates from S3 buckets that appear to have been abandoned months or even years before?” At the start of this we had no idea how this would turn out.

The research panned out progressively, with S3 buckets registered as they were discovered. It went rather quickly from “Ah ha!, we could put our logo on this website” to “Uhhh, .mil, we should probably speak to someone”. After spending around \$400 on S3, CloudTrail, and CloudWatch logs querying, we had some results worth talking about.

When creating these S3 buckets, we enabled logging - allowing us to track who requested files from each S3 bucket (via the source IP address) and what they requested - filename, path, and the name of the S3 bucket itself.

Collectively, these S3 buckets received more than 8 million HTTP requests over a 2 month period for all sorts of things. Those making the queries were requesting all sorts of things: Software updates, Pre-compiled (unsigned!) Windows, Linux and macOS binaries, Virtual machine images (?!), JavaScript files, CloudFormation templates, SSLVPN server configurations, and more.

Had we been maliciously inclined, we could have responded to each of these requests with something malicious like a nefarious software update, a CloudFormation template that gave us access to an AWS environment, virtual machine images backdoored with 'remote access tooling', binaries that deployed 'remote access tooling', scary ransomware, or such, etc. to give us access to the requesting system or network that the requesting system was within.

These many millions of incoming 'give me this file' requests came from the networks of organizations (based on DNS/WHOIS lookups) that included: Government networks in the USA (including NASA, numerous laboratories, state governments, etc), the UK, Poland, Australia, South Korea, Turkey, Taiwan, Chile, and more. Then there were military networks and the networks of Fortune 500s, Fortune 100s, a major payment card network, a major industrial product company, global and regional banks and financial services organizations, Universities around the world, instant messenger software companies, cyber security technology companies, casinos, and more.

We want to take this opportunity to give our sincere thanks to the entities who engaged with us when we realized what we'd stumbled into, including:

- *The UK's NCSC who helped with introductions to the correct teams for us to speak to.*
- *AWS (who took those around 150 S3 buckets off our hands to sinkhole.*
- *A major unnamed SSLVPN Appliance Vendor who worked with us very quickly and directly to take relevant S3 buckets off our hands, and*
- *CISA, who very quickly remediated an example that affected cisa.gov.*

AWS's agreement to sinkhole the identified S3 buckets means that the release of this research does not increase the risk posed to any party—the same issues discussed in this research could not be recreated against the same specific S3 buckets, thanks to the sinkholing performed by the AWS team. We believe that in the wrong hands, the research we performed could have led to supply chain attacks that out-scaled and out-impacted anything we as an industry have seen so far.

As an industry, we spend a lot of time trying to solve issues like "securing the supply chain" in as many complex ways as possible while still completely failing to cover something as simple as 'make sure you don't take candy from strangers'.

WatchTower's posting then delves into the specific details about each of these many extremely embarrassing and potentially explosive exposures.

To best understand how the industry got into this mess we need to talk a bit about Amazon's somewhat astonishing AWS S3 bucket naming. First of all, a so-called "bucket" is nothing special; that's just Amazon's name for a cloud-based directory that can hold files.

The name S3 itself, as in “three S’s” stands for “Simple Storage Service” and simple is exactly what it is. The simplicity of Amazon’s Simple Storage Service likely accounts for much of its early success and popularity. But it may also have contributed to the service’s very spotty security record.

What’s perhaps most shocking about Amazon’s S3 bucket naming is that access to any S3 storage bucket is via an HTTP URL that ends with the standard web domain s3.amazonaws.com. And, surprisingly, that ending can be prefixed with anything that looks like a valid world wide web domain name having from 3 to 63 characters, because that’s exactly what it is.

For example, I have an Amazon bucket named [grc](https://grc.s3.amazonaws.com). That’s right. The bucket’s name is “grc”, which means that the bucket’s full name is: grc.s3.amazonaws.com and that bucket can be accessed by anyone anywhere in the world at any time. And if that seems like a terrifying thing, you’d be correct to think that. The only thing that even begins to make this system safe is access controls. So, of course, I have extremely strong access control security policies set on that bucket so that only I am able to work with it. But we have seen many examples where someone mistakenly – though presumably for some purpose – allowed global read or read/write access to one of their S3 buckets and disaster soon followed.

So I have a bunch of S3 buckets with many wonderful, simple and fun names since I got there early and grabbed them. Assignment of S3 bucket names could not be any simpler. It’s as simple as first come first served. If you attempt to create a bucket, which is always by name, that effort will succeed if that name is not currently assigned to anyone else’s bucket. So I have “grc” and that name is exclusively mine until I delete it. And as long as I have it no one else can have it.

In computing, we call this a “global namespace” — a single shared naming space where every name must be unique — so this means that everyone in the world shares the same naming space. There’s only one Amazon S3 namespace that everyone shares to name any and all of the S3 buckets they may have created. And governed by whatever access controls its owner may have configured, any S3 bucket is accessible, by its name, by appending [.s3.amazonaws.com](https://s3.amazonaws.com) to it.

For the sake of thoroughness, I’ll add that S3 bucket names must be between 3 and 63 characters total, they must always be lowercase alpha from a to z, numeric digits from 0 to 9, periods (dots) and hyphens. They also must begin and end with a letter or number and they cannot contain a pair of adjacent periods. There are also a bunch of specially reserved prefixes and suffixes. But overall, anything that anyone wishes to use will be valid within these guidelines.

Notice that I keep saying that any bucket name that isn’t currently in use by someone. The point here, and this where things get somewhat sticky, is that buckets that are no longer needed by their owner can be deleted. Two things occur when this happens: whatever content they may have contained will be deleted and the bucket’s name, which will then no longer be in use by its original owner, will be released, returned to the available bucket pool and become available for use by anyone who wishes to have it.

So now we can see what the WatchTowr guys did. They created some form of directed brute-force Amazon S3 bucket scanner which they used to search for named buckets that once existed but which were then deleted by their original creators.

The problem was that many widespread automated tools – software update systems, anti-viral templates, Virtual Machine images, even executable program downloaders – continued in their attempts to access those previously deleted buckets.

So when these researchers discovered and re-created one of these previously deleted buckets and began logging the failed file access attempts they were able to quickly learn what resource something somewhere was attempting to obtain from that bucket. The danger of this is obvious and truly horrifying. Given that they could see that files being requested, they could readily choose to return whatever malicious content they might wish.

Since these requests are being made over TCP connections, the true IP addresses of the entities making the requests could be determined. They wrote:

These many millions of incoming 'give me this file' requests came from the networks of organizations that included: Government networks in the USA (including NASA, numerous laboratories, state governments, etc), the UK, Poland, Australia, South Korea, Turkey, Taiwan, Chile, and more. Then there were military networks and the networks of Fortune 500s, Fortune 100s, a major payment card network, a major industrial product company, global and regional banks and financial services organizations, Universities around the world, instant messenger software companies, cyber security technology companies, casinos, and more.

So this is not the result of any bug. This was the result of a fundamentally poor system design. Amazon should have never allowed bucket names to be recycled and reused. And really, bucket names are really just vanity, right? They're like license plates. All you really need is something random and unique that's yours. It doesn't need to be your name, your initials or some cutesie expression. But when users are given a choice, they'll tend to create bucket names that are meaningful to them. And that likely means they could be guessed by someone else.

I'll admit it. I'd rather have "grc" than 0902D7630FB5. But since S3 buckets are almost always accessed by automation, names were never really necessary. They were just fun. But that fun comes at a cost. Since Amazon chose to give us control over our bucket names, they should have appreciated the inherent problem with reuse and made them single use from the start.

I have "grc" and that should be it forever. But it probably won't be. Unless they change their policy – which they could at any time – my use of GRC will eventually end and I'll cancel and close my longstanding AWS account. At that time, the account's data will be deleted and the "grc" bucket will be recycled back into the available pool, ready to live again. I have only ever used S3 as an off-premises encrypted storage archive, the danger for me has never been present. But the guys at WatchTowr discovered that many S3 users are using their S3 buckets as a form of CDN to deliver quite sensitive files.

Both Microsoft Azure and Google Cloud have long provided protections against the inherently dangerous practice of recycling bucket names within a single global namespace which enables this form of "Bucketsquatting", as it has appropriately been called. But Amazon has been slow to come around. Change is always difficult. But Thursday before last, at long last, that finally changed.

Amazon's announcement carried the headline "*Introducing account regional namespaces for Amazon S3 general purpose buckets*" and they wrote the following:

Today, we're announcing a new feature of Amazon Simple Storage Service (Amazon S3) you can use to create general purpose buckets in your own account regional namespace simplifying bucket creation and management as your data storage needs grow in size and scope. You can create general purpose bucket names across multiple AWS Regions with assurance that your desired bucket names will always be available for you to use.

That last phrase *"with assurance that your desired bucket names will always be available for you to use"* reminded me of another aspect of the single global namespace problem, which is that no one owns anything about not-yet-created bucket names.

The Amazon S3 namespace is flat; it's not hierarchical. If I own the domain grc.com then I also own all subdomains and host names of grc.com, www.grc.com and forums.grc.com and noodles.grc.com are all automatically mine. We would say that everything "under" grc.com is mine. But "under" only applies because the domain name system is an inherently hierarchical namespace. There's no "under" in any flat namespace, such as S3 uses.

So, for example, that an organization had the practice of saving annual archives into a series of buckets named "Acme-Enterprises-Archive-2024", then "Acme-Enterprises-Archive-2025" and so on, where the year is incremented successively. If some begrudging es-IT employee wished to cause their ex-employer "Acme-Enterprises" some grief, nothing prevents them from opening an Amazon S3 account and creating the bucket "Acme-Enterprises-Archive-2026". Now, at the end of this year, when Acme Enterprises went to create their succeeding year's archive bucket, that attempt would fail because someone else had beaten them to it. Having a single global namespace shared by every S3 user may have once seemed simple and maybe even fun, but there is a better way. So Amazon's addition of what they are calling *"account regional namespaces"* is a long needed addition.

Their announcement continues:

With this feature, you can predictably name and create general purpose buckets in your own account regional namespace by appending your account's unique suffix in your requested bucket name. For example, the bucket named "ourbucket-123456789012-us-east-1-an" would exist in an account regional namespace. "ourbucket" is the bucket name prefix specified, then we add the account regional suffix to the requested bucket name: -123456789012-us-east-1-an. If another account tries to create buckets using this account suffix, their requests will be automatically rejected.

Your security teams can use AWS Identity and Access Management (AWS IAM) policies and AWS Organizations service control policies to enforce that your employees are only able to create buckets in their account regional namespace. This will help teams adopt the account regional namespace across your organization.

The implementation of this is something of a kludge. The total length of the bucket name prefix and the regional account suffix remains 63 characters. So they've really only done two things: First, any bucket created while this new policy is enabled must end in the proper account number regional suffix. Second, that account number regional suffix is now reserved for employees using that account number. So no one else can create any bucket which uses that suffix.

Unless Amazon also decides to prohibit the recycling of previous bucket names this change will do nothing to prevent "bucketsquatting" on earlier buckets. But if organizations adopt and enable this enforced account number regional suffix bucket naming then at least going forward this problem will be prevented.

