



Bucketsquatting

Description: 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, WhatsApp 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?

High quality (64 kbps) mp3 audio file URL: <http://media.GRC.com/sn/SN-1071.mp3>

Quarter size (16 kbps) mp3 audio file URL: <http://media.GRC.com/sn/sn-1071-lq.mp3>

SHOW TEASE: Coming up on Security Now!, Steve Gibson is here, and I am filling in for Leo Laporte. Kick off the show with H&R Block's Tax Software. Well, it's doing something pretty wild, and Steve has a suggested fix for it. We also talk about what happens when breathalyzer firmware needs to be calibrated. Plus Russians want Telegram and WhatsApp to return to Russia. And, very important, we finally learn what "bucketsquatting" means and what can be done to fix it. All of that plus so much more, coming up on Security Now!.

MIKAH SARGENT: This is Security Now! with Steve Gibson, Episode 1071, with Steve Gibson and me, Mikah Sargent, recorded Tuesday, March 24th, 2026: Bucketsquatting.

It's time for Security Now!. And if you're hearing this voice and going, that's not Leo Laporte, well, good for you. You've got a good ear for voices. I am Mikah Sargent. Leo Laporte is not here with us this week. He'll be back, don't you worry. But until then, I am excited to be joined by the ever-knowledgeable Steve Gibson. Hello, Steve.

Steve Gibson: Mikah, great to be with you again. Leo told us last week that the RSA conference is going on in San Francisco, and so he and Lisa are there, shaking hands with past and present and maybe even future advertisers for security-related things.

MIKAH: Absolutely.

Steve: So I'm glad to have you filling in for him.

MIKAH: It's always a pleasure to get to join you.

Steve: Well, yeah. And, you know, once upon a time, when we had Father Robert, he was our backstop for Leo. And now we've got you, so that's great.

MIKAH: Yeah. Good to be here. Now - go ahead, go.

Steve: I was just going to say that this is Security Now! Episode 1071 for March 24th, 2026, two days, as it happens, before my 71st birthday. So...

MIKAH: Wow.

Steve: I will be, yeah, I feel great, so...

MIKAH: Good. Happy early birthday.

Steve: With any luck we'll be doing Security Now! Episode 2000 before very much longer. Today's episode is titled "Bucketsquatting," and this has nothing to do with, like, something you have to do when you're camping. This is about an interesting problem that Amazon has had for years, which it turns out represents a surprisingly serious security vulnerability, which we're going to cover in detail. But, wow, there's a bunch of other really cool things that have happened in the last week. Turns out that H&R Block's tax software, I think they call it the Enterprise 2025 tax stuff, is doing something that is so very wrong.

Also a cyberattack has hit a company called Intoxalock, which provide breathalyzers to enable the ignition systems on automobiles whose drivers need to prove their sobriety before driving. That's an interesting story. We've also got Firefox, now as of today, we should be at Firefox 149, as of today offering a free built-in VPN. Also TikTok and Meta's tracking pixels turned out to be doing much more than we believed. Russian citizens are begging to get their instant messaging back - you know, Telegram, WhatsApp and so forth - which the Russian government have said no, no messaging for you. We've also got the lack of wisdom of connecting your crypto wallet to an unknown service.

Yet another, and what would a Security Now! podcast be if we didn't have a Cisco CVSS of 10.0. Yes, you're just getting them confused at this point because there are so many of them. But Cisco's not alone. Ubiquiti also have a 10.0 CVSS critical flaw that needs to get patched. We've got some interesting listening feedback. And then what is exactly bucketsquatting, and what can be done to prevent it? So, you know, maybe we have some things to talk about this week. I don't know.

MIKAH: Sounds like it. Sounds like there might a few things to talk about. I'm looking forward to learning about bucketsquatting, I'll tell you that. You know, it's a good exercise move, surely. Works on the device.

Steve: That's true. Strengthen those whatever's.

MIKAH: Yeah, exactly, the whatever's. Let us get into the show, the good stuff.

Steve: So last week I showed this picture, same picture that we have here, but with a different caption. I thought this picture was so bizarre that I would just put it out to our listeners to say, give me an idea for a caption. And so that was a caption contest last week. I got flooded with a huge range of very fun and creative bits of feedback. I settled on one which I like because clearly what we're seeing here with this insane communications telephone pole, a power pole, whatever the hell it is...

MIKAH: It's beautiful.

Steve: This demonstrates, like, what, 50 years of accumulation. Clearly this did not happen in a day; right? It used to be initially, when that pole was erected and the first lines were run to it, I'm sure down there in the core, buried deeply, is what was there originally, beautiful, probably made sense, you could take a look at it and see what was going on. Everything, you know, it's like, it was perfect. Then, like, oh, but wait a

minute, we need to add another trunk line. So, okay, tack that onto the side and wire it in. And then who knows how many decades pass, and you end up with what could, you know, affectionately be called a rat's nest of wires. So I gave this...

MIKAH: It's a rat king's nest. You know what a rat king is?

Steve: Yeah. Yeah. And then there's some poor worker guy up there on the top, like, trying to add just one more wire. I just need one more wire to, you know. Anyway, so I gave this thing the caption "A contemporary visualization of the Microsoft Windows Code Base."

MIKAH: This is the caption you added?

Steve: Yes. That's my caption.

MIKAH: That is beautiful, Steve. I read that, and I thought, yes, yes. Oh, very good.

Steve: And I mean, that's what we see with Windows; right? I mean, and in all fairness, it's not just Windows. It's any old code base that has been evolving over time where you can't really throw away the old code because it's working, and things depend upon it being the way it is. So we're just going to add to it. We're going to, you know, and we've got, you know, Windows now has multiple APIs. I hear Paul Thurrott talking about how, you know, oh, nobody codes to that API anymore. Well, of course I do. But not, you know, other normal coders. So anyway, I thought this was a great caption. It's a variation on an idea that I got from one of our listeners, so thank you for that. And thank you, everyone, for sharing your ideas.

Okay. So this first goodie is where we're going to spend some time this morning because, or this podcast, because there's a lot here to unpack. And I really think our listeners are going to find this interesting. The first mention of this goes to a listener I credit for the first mention, Jack Christensen, who first pointed me at this. Since then The Hacker News and other security outlets have picked up on this and shared it. Jack provided a note which included a link to the original Ycombinator posting whose Chinese author, a guy named Yifan Lu, he appears to know his way around TLS and web servers, as we will see.

So here's what Yifan posted to Ycombinator last week. He said: "Just a" - he said PSA, but we know that stands for "Public Service Announcement" - for folks here in the U.S. because tax season is coming up, and some of you may be using - oh, it's business, not enterprise - "H&R Block Business 2025." He said: "I discovered that the software" - get a load of this, folks - "I discovered," he wrote, "that the software installs a root CA named WK ATX ServerHost 2024." So a root certificate authority WK ATX ServerHost 2024 with an expiration in 2049. Yes, 23 years from now. He said: "...into your local machine's trusted root certificate store. They also helpfully include the private key to this certificate in a DLL file."

He says: "This certificate does not identify itself as H&R Block anywhere and does not get uninstalled when you uninstall the software." He said: "I've been able to successfully use this root CA plus mitmproxy" - which is a software package, man-in-the-middle proxy - "to manipulate TLS traffic on a brand new virtual machine on the same network with a DNS spoofing attack." And he gives us then a link in his posting to a YouTube video. I've got the link in the show notes for anyone who's interested.

He said: "To test if your machine is vulnerable, visit this page." And now we have a URL to a host on his domain, it's "https://hr," as in, you know, H&R Block, "hrbackdoor.yifanlu.com." So this is a TLS, a secure HTTPS secure connection to a web server on his domain that is carrying a certificate he made using this root CA that was installed in everyone's machine who has H&R Block Business 2025, thanks to the fact

that they also provided the private key for this root certificate, which you shouldn't - which should never be there. Anyway, he says: "Go to this URL, and if you do not get any warning or error message from your browser, then you have the backdoor installed." Okay, now, it's not really a backdoor. I understand, I think, that's a popular term. It's frightening and scary, as we'll see. I don't get how this is a backdoor.

MIKAH: Okay.

Steve: But it's not a good door. You know, maybe a side door. He said: "If your browser does complain, you can choose to visit the page anyway for more details on the vulnerability." And I did, and I'll tell you about what I found in a second. So he says: "Is it negligence or a 'real' backdoor? It's impossible to tell. And since the private key is out there, anyone can use it, so the point is moot. There's no legitimate reason" - and, boy, do I agree. And as we'll see, I'm going to demonstrate how we don't need this - "why they need to install a wildcard root CA under a different name. When I contacted them," and I'll be sharing the timeline of that later. "When I contacted them about it, their statement includes 'similar findings have been identified through internal security assessments,'" he said, "meaning they know about this issue but have not fixed it."

He said: "I would not trust H&R Block software at this point. If you did not get bit by this, congratulations." He said: "See this post as a reminder to audit your trusted root CA store." That is, in other words, take this post as a reminder to go take a look and see what cruft things may have installed behind your back for their own purposes because, unfortunately, as this demonstrates, that can happen. And it's not good.

Okay. So let's reverse-engineer this to figure out what's going on here. First, again, I want to 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, I mean, that's bad enough. But then, even worse, to leave it there forever with an expiration date in the year 2049 - and, you know, Mikah, I do think the podcast will probably not last that long. So we're not going to be here to celebrate the expiration of the H&R Block root certificate. So this will remain valid for the next 23 years. Doing that on H&R Block's part is the height of hubris and irresponsibility.

MIKAH: Now, I don't want a spoiler. So if you are going to talk about this, then you can just say that it'll be a spoiler. But I know that when it comes to the practically minded, particularly those in the security world, perhaps the motivation is not as important as the impact. But I would love to know, do you have any thoughts about why, why make this choice of a certificate that doesn't - is it just pure laziness? You talk about hubris and irresponsibility here. That just seems like, well, one would say, "That's a choice." I mean, they made a choice.

Steve: Yeah. Yeah. And in fact I'm going to explain...

MIKAH: Okay, good.

Steve: ...the dangers, and then I'm going to explain the reason for it, and then we're going to show how the same thing could be achieved in a completely secure fashion.

MIKAH: Excellent.

Steve: So I had a lot of fun with this because we never - we've never in, the 21 years of the podcast, haven't seen this and gone into it. So it's a great opportunity...

MIKAH: Beautiful.

Steve: ...to, like, really dig in.

MIKAH: Get ready, folks.

Steve: Yes. Okay. So 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. Certificates have a whole bunch of things they can be labeled, and they can also contain constraints on their own behavior, that is, constraints that they broadcast on what things they can be used for. So this is an unconstrained certificate authority certificate which is the most potent form of certificate you could have. So just as with any root certificate authority certificate, the purpose of that certificate, that self-signed certificate, is to verify the signature of any other certificate that it might have signed using its matching private key.

It contains the public key. So its private key signed something which the root certificate's public key can be used to verify. So it can verify, but it can't itself sign. That's the beauty of this public key, you know, division of labor between private and public keys. So the CA's private key, consequently, like for DigiCert, right, they're a CA. Their private key is the most prized, protected, and safe-guarded piece of information anywhere, since any certificate which that super-secret, protected private key signs will be trusted anywhere that its matching CA certificate containing its matching public key is installed.

Okay. So keeping the private key secret is, like, so important. Did H&R Block at least keep its installed CA certificate private key secret? Well, no. Yifan told us 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's private key, which he found in the DLL, to create and sign his own standard TLS web certificate, just like a CA would, like a certificate authority like DigiCert did when GRC got its most recent certificate. That's what it does.

So he created a TLS web certificate using the private key to be found in this DLL in order to create that backdoor.yifanlu.com website which anybody who installed the H&R Block software can now go to because their machine, their browsers, will all trust this certificate that he made for himself. When I went there, since I didn't install this H&R Block software, Firefox freaked out, as it should, warned me that the site was attempting to use an untrusted certificate signed by an unknown issuer, and that I should proceed no further. I put the dialog box in the show notes.

It says, this is from Firefox: "Someone could be trying to impersonate the site, and you should not continue. Websites prove their identity via certificates," writes Firefox. "Firefox does not trust hrbackdoor.yifanlu.com because its certificate issuer is unknown," meaning to my computer. Firefox said: "The certificate is self-signed, or the server is not sending the correct intermediate certificates." And so then we get the error code SEC_ERROR_UNKNOWN_ISSUER. And you have a link then. You can click on "View Certificate" and see the certificate where we see that everything that Yifanlu has told us is true.

So this is all exactly what we would want and expect from any browser. And it's what we should receive because, as I said, I never made the mistake of installing H&R Block's 2025 tax prep software. But what's significant is that anyone who has ever previously installed H&R Block's Business 2025 tax prep software from now and for the next 23 years, while their purposely 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 complaining, Yifanlu's self-cert-created page, because the signer of Yifan's demo test-site certificate will now be known and trusted by their PC because once upon a time, maybe recently, maybe up to 23 years in the future, they had installed H&R Block's software.

Okay. So, so far this is all just happy demonstration test land; right? The reason Yifan has raised the alarm is that the inherent dangers extend far beyond testing. Since, in addition to installing an untrustworthy certificate authority cert into every PC root store, as I said, and he's proven, H&R Block thoughtfully provided their CA's matching private key. Consequently, anyone, anywhere in the world can generate their own TLS web certificates or code-signing certificates, any kind of certificate because there's no constraint on the use of this that will be trusted without question by any previous user of H&R Block's tax preparation software and 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 could be fully spoofed. Their browser would go to web pages at those URLs. They would see matching trusted certificates and be fully spoofed.

Now, 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' PCs 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, he said: "I've been able to successfully use this root CA and 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 deliberate "MITM," you know, "Man in the Middle" interception in order to 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 into their network under the protection of TLS encryption, and they'd like to prevent corporate trade secrets from being sent out through their network, either inadvertently or deliberately, thanks to that same TLS encryption.

So to accomplish this, every PC operating inside their corporate network environment must contain the root certificate for that middlebox. In other words, a CA root certificate for that middlebox. And we hope that its matching well-protected inside the matching private key is not extractable. So having all PCs in the enterprise containing that middlebox's root certificate, and with it having the matching private key, which it's careful never to let anybody else get, that allows the middlebox to synthesize fake trusted remote website certificates on the fly.

So here's how that works. Say that someone inside the enterprise attempts to go to `ChatGPT.com`, the middlebox will intercept that connection attempt; and it, itself, will go to `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. And it will then return that cloned certificate to the enterprise user who will believe they've connected directly and privately to ChatGPT. Their browser will show `www.OpenAI` or `ChatGPT.com` or whatever the URL is. However, they've actually connected to their enterprise's middlebox which is masquerading as ChatGPT. Which it can do because their PC contains that root CA for the middlebox. Therefore, middlebox created certificates are trusted inherently.

Now, 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 10-year product planning in order to

ask ChatGPT what it thinks about those plans. Those should not leave the enterprise. So somebody needs to guard those communications.

Yifan's point about the installation of a root certificate and man-in-the-middle proxy is that, for reasons that are not at all clear, H&R Block has thereby, by doing what they've done, they've given themselves all of the same privileges and capabilities as any enterprise middlebox on their customers' computers. The question is why.

MIKAH: Too much power.

Steve: Yes. They could intercept all of their customers' communications to any other website.

MIKAH: Oh, my god. Why would you want that level...

Steve: Yes. Why would you want the responsibility?

MIKAH: Yeah, that's too much. Yeah, no thank you.

Steve: The only reason H&R Block would need to include the private key that matches the CA root certificate they installed into their users' 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 as it is. So again, why? Why is it?

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 - anyone else, like him, this researcher - 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 extract and 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 have not 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 their tax preparation logic. In other words, we go to Google Docs in order to use Google's document application. It would be possible to instead go to H&R Block software server with our browser, running on our machine, to use their tax prep application delivered to us by a server running locally.

So clicking, for example, some sort of little startup app would invoke the Windows shell to launch the user's default web browser with a URL-like `https://127.0.0.1:` and then, like, some port number, 1234. Or some custom port number, whatever they wanted to use. Or they might have also tweaked the user's hosts file to map a more friendly-looking domain name like `hrblock-tax-prep.com`, mapping that to 127.0.0.1 IP. That way users would see something comforting in their web browser's URL address bar. But none of that actually explains why they couldn't then just install a root CA and a certificate for that. There's no need to make it up on the fly. I'm, you know, kind of like I'm giving them the benefit of every possible excuse here.

MIKAH: Right.

Steve: Okay. So now the user clicks on the startup app, which launches their web browser, which accesses their locally running web server. That built-in local web server needs to present their browser with a TLS certificate that matches the address they've accessed. Again, 127.0.0.1, or hrblock-tax-prep.com or whatever. 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, that root certificate subject name as Yifan told us, is "WK ATX ServerHost 2024," okay, it seems unlikely that it would be used directly as the server's end certificate. That would be a weird string for users to see in their browser's URL bar. It's more likely that the installation process or perhaps maybe the first time you run the system, that it would create the built-in web server's certificate which it would then have signed using the root CA's private key, which we know is sitting in a DLL. And this is exactly, as I said, what Yifan Lu did when he created his test site.

Once all 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, which it must, the web server would have that key, 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 which it provides to the web browser. But that should never be the private key for the CA root certificate. No one does that; right? The only thing the root signs is another certificate, which is then presented to the web browser. It itself is never used. Thus you never need its private key to be around.

Okay. So when all this was, like, revealed, when Yifan found all this, he did the right thing and reported his astonishment and concerns to H&R Block. The timeline of his interactions with them was, on March 10th - so today is, what, the 24th? So two weeks ago today, on the 10th, he disclosed to H&R through their Responsible Disclosure Policy. Two days later, on March 12th, he was asked to "please provide more details about how an attacker would realistically exploit this in practice" and "share the video proof of concept from start to end."

Three days later, on the 15th, he did that. He wrote: "Provided H&R Block with details and a proof of concept video and told them the deadline for their response is March 20th, due to U.S. tax season coming up." On the 20th, he received the following statement back 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." Okay, what? So, like, yeah, we know. Like, uh, really? Somebody actually explained this to you, and you're like, eh? That's how it's supposed to work. Okay.

Okay. So this obviously 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 certificate with a 23-year lifetime remaining and its known private key, sitting in the root store of every customer of their tax preparation software, and it persists even after their software has been uninstalled.

Okay, now, the one thing we've not examined is what I would do, I, Steve Gibson, if I were told, you know, or if I were contracted with to do this in a secure fashion. So Mikah, let's take a break, and then we're going to look at that, how to do this right.

MIKAH: All right. I am looking forward to hearing how we can do it right after seeing how you can do it so, so wrong. We are joined once again by Steve Gibson, as we continue on with understanding what we would do if tasked with fixing things for H&R Block.

Steve: Okay. So first of all, let me reiterate that I'm sort of, like, trying to give these guys every benefit of the doubt. I don't understand why they've done this. The only real reason that I can imagine is actually traffic interception. They would need this to act like, you know, in order to act like a middlebox, to intercept encrypted communications. Maybe, and again I've not seen the software, maybe they set themselves up, and then their app says, okay, now go to IRS.com. Go to your bank's website. I mean, like, maybe there are remote cloud-based sources of information which it asks you to look at, and while intercepting your communications, so that it can suck that stuff in and incorporate it into your tax preparation.

I don't know. I mean, the only thing I can imagine. Now, still horrible. I mean, you don't want a proxy that's able to intercept all of your TLS connections installed on your machine, let alone left behind after you've uninstalled it. But it's the only thing I can imagine. So, and also, if all they wanted was to be able to run a local web server that your browser trusts, then they would need to install their own root cert, and the web browser would then have a certificate that that root cert has signed. But there would never be the root cert's private key because it wouldn't be necessary. And the only thing that certificate would do would be to run a local web browser. You know, web app developers do this all the time. They run a local web server on their machine. It's not a big deal.

So H&R Block could have done that. That's all they would have needed to do, and it could have been safe. But say that for some reason they did need to deploy this on the fly. Even that could be done securely. And here's how. 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, it's possible. So H&R Block, are you listening?

Upon installation, the installing software would first generate a state-of-the-art standard 4096-bit public/private key pair. They would not ship that with the product. In other words, this thing that H&R Block did was they provided a public key in a root CA, and the private key bundled into, bound into the DLL. So they ship those statically, which means anyone who gets a copy of one has what they need for all H&R Block users, which is where the problem is. No. Instead, generate it on the fly. Generate your own 4096-bit public key pair on the fly so that no two installations ever share the same keys.

The public key half would be contained within, to your point, a short-lived root certificate, you know, having a lifetime of the expected duration of the product's use. Maybe 90 or 120 days for tax preparation software. Mikah, as you pointed out, there's no reason for it to live any longer.

MIKAH: Yeah. Because people are going to reinstall it again the next year, if they uninstall or something. The theory is they're going to install it again and use it again next year.

Steve: And, since it's able to make a certificate, if it did expire, it could just make a new one.

MIKAH: Yeah.

Steve: Even it itself could just bump this along forward, you know, always installing a few months' worth of root CA that will gracefully, properly expire after a few months. You'd like to have the uninstaller remove it because you don't want to just have all this

crap accumulating in your users' root stores. But okay. In addition, it should tightly constrain the use of the certificate so that it can only ever be used to sign a TLS end certificate. Those are constraints that could be added to the CA. They didn't do that, either.

Okay. Next, it generates a second 4K-bit public/private key pair. It uses this second key pair to create the web server's local site certificate. So that's going to be - that's the certificate for the server that the server will send to the browser, to the user's local browser. And it would name it something like, you know, `hrblock.localhost`. That would be a safe name. `localhost` is reserved for the localhost, as sort of a localhost pseudo domain. So `hrblock.localhost`, which would mean that certificate could never be misused in any useful fashion. It would only ever be able to serve and encrypt and authenticate TLS connections.

In order for this local site certificate to be trusted on that local machine, it needs to be signed by the root certificate's private key, that first one that was made, you know, the one belonging to the just-created local root CA. And then here's what's crucial and cool. Immediately after that root CA's private key was used to sign the local site certificate, that private key is securely overwritten and deleted. The point is that private key is never written to non-volatile storage, so it is now permanently gone, and the locally installed root certificate can never be abused because its matching private key, which is required for its abuse, no longer exists. It was just - it was ephemeral. It was transient. It's gone. Since its private key was only ever needed once to sign the local certificate to prove that certificate's validity, it should never be retained.

So the installation then in my system would - in my solution would add 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, I guess 80 if it's free; but you might want to do, you know, `8888` just for the sake of being out there somewhere. So, you know, `hrblock.localhost:8888`, and now you're able to access the built-in web server in order to view the H&R Block UI in a server-client relationship. And lastly, 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 deletes all of the product's files.

So as we've just seen in this example, it is, if you really needed to do this for some reason, I don't see even why you have to because you could just, you know, give somebody a set of static certs that would only be useful for `hrblock.localhost`. But even if you needed, if you wanted every instance to be unique, and I can see a benefit there, it can be done in a way that does not open any security holes. But that's not what H&R Block has done. Through very poor security design, sloppiness and, as you said, apparently not caring, they've left behind a 23-year lifetime completely unconstrained CA root certificate, meaning it can sign TLS web certificates, it can sign code, it can be used for any malicious purpose you can imagine, left behind in the root store of every one of their users, with its matching key statically embedded in a shipping DLL so that it's known to the world.

This makes you wish that our industry's software liability protections were not as virtually nonexistent as they currently are. I mean, I'm sure somewhere in the license agreement that all users click on, not reading because you can't, you know, it's 25 pages of legalese. It says, you know, by using this software you agree to hold us harmless of any damage that may arise, you know, even if we had been informed beforehand that such damage could occur. The license agreements all say that. And so it's like, eh, we can do anything we want to your computer.

MIKAH: That's so frustrating to me. Because I was going to say, you know, I'm always curious. He reached, this person reached out to H&R Block. H&R Block had to know that if this was coming from truly a security researcher, that this was going to be made public, as opposed to it just being some, you know, I would imagine if you just heard from some random person that did not, you know, put themselves forth as a security researcher at all, they might go, oh, we can bury this. This won't matter. But to say, basically, we don't care to someone who is going to disclose?

That's the stuff where I'm like, I want to meet every person involved in every part of these decisions and just hear what their thinking is because what is their thinking? And why would they feel that this is not a big deal when you're showing us how it can quickly become a big deal, and that the responsibility there is vast. But as you pointed out, with the right agreements in place, they don't need to care, do they.

Steve: No.

MIKAH: It's frustrating.

Steve: Speaking of frustrating, the company known as Intoxalock - got to love the name - provides court-mandated automotive "breathalyzer" facilities that are installed into the automobiles of people who, a court has come to the determination, should be required to provide proof of their alcoholic sobriety through a quick built-in breathalyzer breath sample every single time they get behind the wheel and wish to drive their car. Now, since alcoholic breathalyzer technology is tricky and can be a bit flaky, it requires periodic calibration for reliable operation by an authorized calibration facility. And the calibration facilities, it turns out, are all tied back to the mothership in the cloud that tracks and reports on all of this.

Now, that whole system generally works pretty well, but it turns out 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. So today is the 24th. A full 10 days later, their calibration system remains offline. It's down.

MIKAH: Wow.

Steve: 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, if you don't get a recalibration when it's needed, you don't get to drive. And you cannot now recalibrate. As a result, as the 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 - and this is on a state-by-state basis - apparently able to offer a 10-day calibration extension, but that appears, as I said, to be limited by state.

A posting on their "service status" page explains. It writes: "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." That's today. "At this time, this extension is not available in Michigan or Washington."

MIKAH: Wow.

Steve: "We're actively working toward a resolution and will notify you as soon as anything changes." So here we have an interesting case of people's physical lives being

meaningfully impacted by a cyber event. Since the calibration and reporting system sounds like it's very database based...

MIKAH: Database based.

Steve: Yeah, I would not be surprised to learn that Intoxalock, although this hasn't been publicly disclosed, had fallen victim to a generic ransomware attack which encrypted all of their systems. And in this case, consider the fact the data that may also have been exfiltrated...

MIKAH: Oh, wow.

Steve: Yes.

MIKAH: A list of all the people, oh, my goodness.

Steve: Yes. All of the people, extra personal, extra private, and extra sensitive, you know, involving the identities and the habits, the drinking habits and the drinking and driving habits of U.S. drivers who have been under court-mandated driving restrictions. That's not the sort of data anyone would wish to have floating around the Internet. I would argue, you know, it makes a Social Security number look tame by comparison.

MIKAH: Yeah. This is a huge blackmail target.

Steve: Yeah.

MIKAH: Honestly, I was going to make some silly joke about I wonder how many executives didn't get to work on time that day, and then you said what you said. And I thought, actually it gave me goose bumps. This is horrible. This is awful. This is terrible.

Steve: Yeah. It's not good. And they apparently are the go-to company for this service. I was looking to see...

MIKAH: Of course.

Steve: ...what their status page says today because this was support.

MIKAH: It also made me wonder, while you're looking for that, it also made me wonder if the calibration narrative is pushed by that company so that they can continue to make money even after those breathalyzers are installed. And of course that's just, you know, supposition. But it is a very interesting thing to be one of the main companies providing what ends up being sort of a government-mandated device needing to have these regular check-ins.

Steve: Yeah. And the good news is they have got their systems back up. I went to [Intoxalock.com/status](https://intoxalock.com/status), and I'm getting a green bar, systems restored, services and installations resuming. So, and it's got lots of instructions for people who are inconvenienced, and there's a mobile app and so forth. So note if you received a service data extension in Tennessee during the temporary pause, please return to the service center today, on March 24th. Failure to do so may result in an extension or full restart of the interlock program, whatever that means. So anyway, the good news is looks like they've paid ransom or they restored from backups. We don't really know. But they are back up. But if nothing else, this demonstrates that our cyber world is increasingly interacting with the physical world. And, you know, here cause people a lot of inconvenience.

MIKAH: Especially when you've got all of your eggs in one basket, or very few baskets.

Steve: And as you said, even with the system restored, if they did, if this was a cyberattack, and they exfiltrated all their database data, as you said, now there's serious extortion opportunities for anybody important who, you know, cares about keeping their past problems with drinking and driving private.

MIKAH: Yeah, that's mortifying.

Steve: Okay. So exactly one week ago, last Tuesday the 17th, Mozilla posted some welcome news under their heading "More reasons to love Firefox." I don't need any more reasons. I love Firefox. But okay. They wrote: "What's new now, and what's coming soon." They start their post off rather generically by writing: "Firefox is for people who make their own choices online." Apparently they're saying, you know, Chrome people are sheep. I don't know.

"From what stays private," they wrote, "to the tools that help get things done. That commitment to choice shows up throughout the Firefox experience. The AI controls is just the latest example" - meaning you can turn them off - "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."

Okay. Then they talk about the ability to turn off any generative AI, a new feature coming in the next Firefox 149 which will allow side-by-side page display and the ability to write and attach notes to tabs. Okay.

MIKAH: Yeah, okay.

Steve: Yeah. I'm not sure about the whole AI thing. The side-by-side web page thing sounds as though it might come in handy for me for podcast production since I'm often flipping back and forth between Google Docs, where I author the page, and the source information. So that would be cool. But the forthcoming new feature that caught my eye, and which I felt sure would interest our listeners, was Firefox 149, which drops today, on March 24th, its 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." And I've often said, you know, don't trust a free VPN from some sketchy provider. Certainly Mozilla is not that. They said: "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 24th." In other words, as I said, today.

Now, 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, unfortunately, its desktop presence continues to slip in terms of market share. It's down from 6.3% last year, now just down to 4.2% desktop share this year. So Mozilla may be hoping that the presence of a built-in free VPN service will help. It is worth noting also that it has not escaped, unfortunately, the awareness of legislators that people are rushing to use VPNs in order to get around state-based location age restriction, and there actually has been some conversation about legislation to prohibit VPN use. Which, you know, good luck with that. I mean, it's

like, hey, you know, those TCP connections, they're pesky. I think we'd better outlaw TCP.

MIKAH: Oh, lord. Can you imagine?

Steve: You know, come on, guys, where does it stop? Okay. So I just learned how far "tracking pixels" - and we now need to put that in air quotes because they are so much more. They're easy to miss because, much like cookies, the code which their presence on any web page allows to run is completely hidden from us. I mean, it's not that you can't get it, but it's not easy to see. Last Wednesday the 18th, the security researchers at Jscrambler shared what they had recently learned about what TikTok and Meta are both now doing.

Their headline was: "Beyond Analytics: The Silent Collection of Commercial Intelligence by TikTok and Meta Ad Pixels." Okay, now, as we're going to see, this writing, Jscrambler's writing is targeted at web merchants who are voluntarily putting these insidious tracking pixels onto their sites because, you know, this is not something that happens without the site provider's knowledge. This is something that they said, oh, yeah, we want the analytics, or we want the whatever we're getting in return, so every page that we serve will have a reference to some JavaScript back at Meta or at TikTok or wherever, basically causing the user's web browser to pull in that resource and invoke whatever the script is.

So here's what they explain. They wrote: "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, as well."

Okay. Now, I'm sure I don't need to tell our listeners that this is not something I, with GRC, would ever consider doing. I'm annoyed that I've given Google any presence at all. But their little search box is - that's a pro-visitor feature. Other than that, GRC may be ancient appearing, but it's also completely devoid of all modern web analytics because I just - I would rather protect my users' privacy. What's so insidious about this is that, when a company says, okay, yes, we'll make a query to your ad server to pull in your ad pixel, they have no control over what it does once it gets there. That's the key point. It's very much like software that goes out and downloads some other software to enhance itself. If the behavior of that augmented software changes, then the overall product changes after the fact.

MIKAH: Yeah.

Steve: Anyway, Jscrambler continues, writing: "Tracking pixels were once just a small snippet of code on a webpage to confirm an ad impression or 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 now go far beyond those traditional tracking tags. They collect user emails, phone numbers, and addresses, turning seemingly anonymous browsing data into persistent, identifiable user profiles.

"TikTok's pixel," they write, "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; and 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's easy to re-identify them by matching those hashes against existing hashed data." Meaning that if Meta has your email address and hashes it, they get the same hash. If they have your phone number and hash it, they get the same hash. So when those hashes later pop up on the web, they know it's you. There's no mystery there. You are not anonymized.

And they write: "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," they wrote, "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 log specific checkout actions such as AddToCart or AddPaymentInfo." In other words, stuff that is none of their business. "Meta's telemetry even records the structure of checkout forms and buttons, providing insight into how a merchant's site is built." Wow.

MIKAH: Are they making this data available to the people whose sites they're on? That's wild. This is not a trade.

Steve: No. It's for their own - it's total intrusive for Meta's benefit. And maybe they're selling it.

MIKAH: Yeah.

Steve: I mean, it's data that they are aggregating. So I'll just interrupt to note that, you know, you might be thinking, if you might be thinking that none of this is any of Meta's effing business, I would agree with you wholeheartedly. It is so wrong and intrusive. They do it simply because they can.

MIKAH: They can, yeah.

Steve: Because it's hidden. Because web browsers will run, by default, any JavaScript they're given. And because there's no one looking. There's no one to stop them. Jscrambler continues, well, and I'll note also, Meta can say, oh, but we hash the data. We anonymize it. No, you don't. You're not throwing in a random token with the hash because, if you did, it wouldn't be useful to you. It would then be purely random noise.

So they said: "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're 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 the 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 their 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."

MIKAH: Why? I mean, I know why. But I just - you should...

Steve: Full, full spying.

MIKAH: Yeah. It's absolutely full spying. And to do it without any regard of just, like, they're not even having to explain themselves because, as you pointed out, no one is paying attention to this.

Steve: Yup. They write: "Since this is the default behavior and not opt-in, merchants may not be aware that the pixel is collecting this information." The pixel. I love calling - it's like calling it a pixel makes us say, oh, look, it's a little bitty pixel.

MIKAH: Just a tiny little - exactly.

Steve: That's right. "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" - hear it again. "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" [crosstalk]. We asked the user, they said no, but oops, it was too late.

MIKAH: Yeah.

Steve: "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." So it wasn't even well done, like a privacy-first approach. They said: "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 would suggest.

"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 for identity theft and secondary data

breaches. From a competitive standpoint, merchants need to understand that the pixels they" - pixels. I mean, calling it a pixel is so wrong.

MIKAH: Yup, I agree.

Steve: "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." Because there's more ads available for choosing.

"To manage these risks," they write, "organizations need to do considerably more than just review a pixel's documentation. This involves auditing actual pixel configurations," meaning work, "and implementing continuous monitoring to catch 'scope creep.'" Meaning the pixel used to be a cute little thing that only targeted ads, but that was 10 years ago. Now you're downloading a whole suite of spying intelligence ware in that cute little pixel's JavaScript. They finish, "...where a third-party script begins collecting more data than originally intended." Exactly so. Wow.

MIKAH: Geez Louise.

Steve: Yeah.

MIKAH: That's frustrating.

Steve: So again, it's commerce, you know, getting away with everything it can. And of course why wouldn't they? You know, we're talking Meta. They're an aggressive commercial organization, and they've convinced the whole world to put their cute little tracking pixels everywhere. Yikes.

Okay. As we know, no one is an island. Unfortunately, that's a problem if you don't get any messages. 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 their government to lift the recently imposed total bans on Telegram, WhatsApp, and other foreign messaging platforms. It seems that not everything needed to conduct business in Russia can be found within Mother Russia, and that Russian entities need to contact and work with foreign partners.

And unfortunately, Russia is saying, no, we don't want you to be using, you know, they've got their own - I can't remember now what it's called. They did a native Russian messaging app, which of course no one wants to use because we know that Russia is spying on it. So nobody outside Russia wants to have anything to do with some Russia spyware. Who knows what it does once it gets into their machines?

In another bit of news, I saw this blurb in a security news summary, and I thought, okay, well, that's interesting. 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. So that's all we know. I don't ever want to see anyone hurt, of course. Really, I don't, ever. But anyone who would naively connect their crypto-wallet containing any amount of crypto which they're not entirely prepared to be separated from in the next moment, especially

if they consider themselves to be savvy enough to be a developer, they would have a difficult time extracting much sympathy from me.

MIKAH: Yeah.

Steve: Even though I would never want them to be hurt. I mean, really. You know...

MIKAH: You sometimes need an object lesson. And someone who's making those mistakes needs perhaps an object lesson.

Steve: And I just hope it's not too expensive a lesson. I would certainly be sorry if anyone were scammed. But I would not be very surprised. So our takeaway here is "Please, please always be careful, especially anytime you're connecting any wallet to any sort of automated system which you cannot be 100% certain of." Really. Like set up a secondary wallet. Transfer a little bit of working money there. And then, you know, use that if you must connect it to something. But, you know, not your wallet where you actually store any useful amount of money.

Since we've previously touched upon Cisco's very bad 10.0 CVE-2026-20127, which was that widely exploited authentication zero-day discovered while being exploited in Cisco's Catalyst SD-WAN enterprise product line, really, anyone could be forgiven for confusing that one with Cisco's CVE-2026-20131, so not 27, no, 31, which is another - wait for it - CVSS 10.0 critical vulnerability in Cisco's systems. As I said at the top of the show, what would the Security Now! podcast be without a brand new shiny Cisco CVSS critical 10.0?

The NIST NVD, the National Vulnerability Database, says of the new one, 31, they write: "A vulnerability in the web-based management interface" - who would have guessed? - "of Cisco's Secure Firewall Management Center" - apparently not that secure - "Software could allow an unauthenticated, remote attacker to execute arbitrary Java code as root on an affected device." In other words, there you go, Cisco, 10.0. Woohoo. They wrote: "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 has not 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. Right? I mean, one plus one equals two. 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 promises 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 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 4th, 2026." Now, that's an important date, March 4th, 2026. "Disclosure" means patch available on March 4th, 2026.

Amazon wrote: "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," they write, "This wasn't just another vulnerability exploit. Interlock had a zero-day in their hands, giving them a five-week 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."

Okay. Just so that everyone is clear about the timing of this again, Amazon discovered exploitation of this zero-day dating back as far 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 five 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 through no fault of theirs. They were fully patched and updated.

Amazon explained what they found, writing: "A misconfigured infrastructure server" - essentially a poorly secured staging area used by the attackers. They actually found a misconfigured infrastructure server of 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 (meaning automated tools for mapping victim networks), and evasion techniques." In other words, Amazon has honeypots. The bad guys infected a honeypot. Amazon was able to use the infection to track backwards up into the attackers' infrastructure, which they found improperly set up so they were able to get in.

MIKAH: Now, hold on. What if - maybe I'm writing a movie now. But what if that was just a reverse honeypot? And it pointed Amazon down the wrong route when they found all this data?

Steve: That could be possible. Although what they worked from was Cisco's fresh disclosure.

MIKAH: Ah.

Steve: So they had evidence that that system was attacking their honeypot back as many as 36 days earlier using information that was not public. So nobody knew how it could be used to get into their customers' machines.

Okay. So just to finish on Amazon's Threat Intelligence, they wrote: "AWS infrastructure and customer workloads on AWS were not observed to be involved in this campaign." Meaning Cisco customers, not Amazon customers. They said: "This advisory shares comprehensive technical analysis and indicators of compromise to help organizations identify potential compromise and defend against Interlock's operations." Right? I mean, this was going on for 36 days. Anybody who the bad guys could find who had this firewall may well have been compromised. So a true problem.

They said: "Amazon threat intelligence identified threat activity potentially related to this CVE 20131 beginning January 26th 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." So that was the compromised

system sending stuff back up to the bad guys' infrastructure. They said: "Multiple variations of these URLs were observed across different exploit attempts.

"To advance the investigation and obtain additional threat intelligence, we performed the expected" - so they were pretending to be infected. "We performed" - "we" Amazon - "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." So that suggests that the Cisco firewall is Linux-based, and so this was downloading Linux malware into and to be run by the Cisco firewall.

They said: "When analysts retrieved the binary, they discovered the same host (hacker-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." And if they were able to look around in there, they may have seen all the uploads from all the infected systems. No one's talking about how many systems were infected, but my guess is Amazon knows, and Cisco probably knows and hopefully is not happy about it.

They said: "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 the TOR negotiation portal are consistent with Interlock's established branding and infrastructure. The ransom note's invocation of multiple data protection regulations" - oh, you've got to love that. We just attacked you and infected you, and now we've exfiltrated your data, which puts you in violation of various data protection regulations.

MIKAH: Congratulations.

Steve: Oh, by the way, yes, congratulations. So they said: "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 and exfiltration, but with regulatory fines and compliance violations."

MIKAH: Wow. It's clever. It's clever.

Steve: They're bold. "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 anyone who might worry that they may 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 cannot 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." Right?

So 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 which could fail, and we see keeps failing, an organization's security perimeter could be breached even if they did absolutely nothing wrong. During that at least five-week interval, any fully patched and fully up-to-date Cisco Firewall could have been successfully breached through no fault of their IT managing staff.

Unfortunately, this is not what we normally see. We are usually noting that lax and lazy and inattentive IT was at fault, at least at fault, for not keeping their equipment up to date. Right? Like if a patch was available months ago, and they still hadn't got around to updating. 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 now rests upon that single point which could fail.

And that state of affairs would be acceptable if nothing more could be done. Right? If, like, if we've done everything, and we still have a single point of failure. If nothing more could be done, then okay. I guess that's the best we can do. But it is 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 no one in China, no one in North Korea, no one in Iran, no one in Russia can even get to because there's port filtering that blocks their access to that authentication interface. Unless you really need everybody in the world to be able to guess your password, don't give them the opportunity.

MIKAH: So then arguably you said in this case IT did everything right. But it seems like you are also arguing that they didn't because doing everything right would also mean that you have security in depth.

Steve: Right. Yeah. So that's a very good point. What I meant was IT was at least keeping everything patched.

MIKAH: Got it.

Steve: But this demonstrates that even keeping everything patched is really no longer enough. You need other layers. And if somebody had other layers, although you wouldn't want to be lax on patching, at least being lax on patching, like being a few weeks or months late, well, you wouldn't get infected because bad guys couldn't even attempt to infect your system. So really, you know, we would - the jargon that I've been using most recently, that I used at Zero Trust World and before, was really bringing stronger meaning to least privilege. You want least privilege to apply.

It is a privilege if someone in China has the opportunity to guess your password. Why? Why have you given Chinese attackers the privilege of doing that? They shouldn't even be able to see that you have imagined the interface. You know, they should be blocked by simple IP-based filtering. And it's so easy to do. But everyone says, oh, look, Cisco, you know, they have a, you know, oh, we've got security. No, your security needs security.

MIKAH: I like that. That's a shirt for Security Now!. "Your security needs security."

Steve: Your security needs security. Okay. So finally, before we talk about listener feedback, I just did want to mention that Cisco's not alone. Last Wednesday, and then

updated on Saturday, Ubiquiti released a security update to patch a critical, also one of the rare - you know, it used to be that CVSS 10.0 were rare; right? Nobody had them. Like the worst you would see was a 9.8. And we would joke about, oh, you've got to try real hard to get up to a 10.0. Well, unfortunately Cisco has put the lie to that because they're getting them all the time now.

In this case, so did Ubiquiti. A CVSS 10.0 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 - that's never good - and take over UniFi gateways that way. So Ubiquiti UniFi users are advised to update. I know we've got a bunch of Ubiquiti users and UniFi users. Leo is a big proponent.

MIKAH: One of them.

Steve: So everybody should update. Okay. Some listener feedback. Vern Mastel, his email had the subject "Clocks, Cursive and Coding with AI." And 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."

He said: "For decades we've 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." I have to agree. I think we're very clearly seeing that actually writing code will change into managing its authorship by AI devices. That clearly seems to be where we're headed. And I remind everyone what we have today is not what we're going to have tomorrow. It's going to be getting way, way better.

Jeffrey Coe wrote: "Hey, Steve. Since it's getting close to tax time, I thought you might enjoy seeing the IRS version of the ClickFix exploit." Then he said, "I blurred out the script." He said: "I'm a longtime listener, every episode, and SpinRite owner. Regards, Jeffrey D. Coe."

So, and he attached in his email a picture of an Internal Revenue Service Department of the Treasury sent from Austin, Texas with their zip code. It's Letter No. 1058. And it says: "Final Notice - Notice of Intent to Levy and Notice of Your Right to a Hearing. Addressee: Jeffrey." And then we have a Case ID, and it says: "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." Whoa, 1099s that you didn't report. That's never good.

"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 a failure to respond. This notice does not contain clickable links. Acknowledgment may only be made via the secure method indicated."

And now we have the increasingly familiar and unfortunately increasingly successful ClickFix variant. It says, in a separate box callout, it says: "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." And then it provides down below "Acknowledgment code." Which starts out reading "powershell" space quote, and then we have something that Jeffrey blurred out, then we have "/irs" and then a "space | iex."

So again, as we know, this succeeds because so few people actually understand how Windows works. They're users who basically follow scripts of various forms. And this latest trend of "ClickFix" to me is truly frightening. We've learned and previously reported that more than half, I think it was 52%, of all exploits combined are now attributable to this single category of ClickFix-related social engineering attacks. More than half of all successful attacks. And even before we had this number, it was clear just looking at them that these attacks, which leverage the users' lack of true understanding of how their PCs operate would turn out to be devastating. So Jeffrey, thanks for sharing that.

Jim Housley wrote: "Listening to SN-1070 from March 17th," so that's last week. "You were talking about 'properly' signed malware, and you commented about how the certificate had to be in an HSM," you know, Hardware Security Module. He said: "However, in the previous two or three 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 seemed 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, huh. Has there ever been any trouble with authenticating who people are over a network? Hmm. Yeah, uh-huh. 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 ever-prowling security. That's where mine is. And it has never suffered a break in. So I think old-school physical security is a little better than allowing foreign attackers from states we don't trust free roam and access, trying to impersonate us and get our code signed. I think Jim is exactly right.

And finally, Michael wrote: "Hey, Steve. You're 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 own 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" - and I bet you would, Mikah, because you're a craftsman.

MIKAH: I was going to say, ooh, this sounds fun.

Steve: Uh-huh, "who make their own lures because they're 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'd much rather write my own software, make my own lures, essentially, 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're not alone. Long-time listener and huge fan, Michael." And I like Michael's fishing lure analogy a lot.

MIKAH: Yeah.

Steve: I think it clearly articulates the craftsmanship aspect of coding which anyone who codes because they love it just for its own sake would understand. So for me, AI

producing code does not represent a worry or competition. I've never been interested in coding in a production environment. I love that many more people than ever before are now finding themselves able to get their PCs to do things they never could before because AI is able to create code for them which does what they want. You know, to me, that's the greatest innovation so far on the AI coding front. And I'm sure that, as I've said, we've only seen the tip of the iceberg. We've got lots more to come in upcoming years.

MIKAH: All right. Back from the break. I've got my bucket. My legs are ready. Tell us all about it.

Steve: Okay. A little over a year ago, back in February 2025, so 13 months ago, 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 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 even more difficult than we thought. Or as we just recently noted, security needs to be more secure.

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 into the label of "policy" as opposed to "mistakes." So this suggests that even after AI being used to improve our security has matured, as I'm sure it will, and it's able to help us far more strongly eliminate traditional 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. So I don't see security as an issue being resolved by AI.

So this brings us back to watchTowr Labs' exploration from February 2025. It's 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." So here's what they wrote back then. They said: "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," they wrote, "to demonstrate the scenario of a significant Internet-wide supply chain attack caused by abandoned infrastructure." I'll just pause to note that we've looked at problems with abandoned infrastructure before, where something has all been set up and is going. It's for whatever reason, you know, been left behind. Pieces of it maybe have been taken away. But some pieces remain, and those end up being targets of abuse.

So they said: "This time, however, we dropped our obsession with expired domains" - which of course is an example we've looked at before - "and instead shifted our focus to Amazon's S3 buckets." Thus bucketsquatting. They said: "It's important to note that, although we focused on Amazon's S3 for this endeavor, 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 in any storage solution provided by a 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 and update pipelines before they were abandoned." 150 Amazon S3 buckets that were used before being abandoned. They said: "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?"

MIKAH: Wow.

Steve: Uh-huh. "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 'Aha, we could put our logo on this website' to 'Uhhh, .mil? We should probably speak to someone about that.'"

MIKAH: Oh, my god.

Steve: They said: "After spending around \$400," as in only \$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 - the filename, the path, and the name of the S3 bucket itself. Collectively, these S3 buckets received more than eight million..."

MIKAH: Are you kidding me?

Steve: "Eight million HTTP requests over a two-month period for all sorts of things. Those making the queries for whatever it was that used to be there were requesting all sorts of things: software updates, pre-compiled unsigned Windows, Linux, macOS binaries, virtual machine images, JavaScript files, CloudFormation templates, SSLVPN server configurations and credentials, and more. Had we been maliciously inclined, we could have responded to each of these eight million 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 and so forth, to give us access to the requesting system or network that the requesting system was within." And in some cases .mil.

They wrote: "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, et cetera), 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 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," they wrote, "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."

MIKAH: Wow.

Steve: Yeah. "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.'"

Okay. So their 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. It's just Amazon's name for a cloud-based directory that can hold files. The name S3 itself" - that's three esses; right? - "stands for 'Simple Storage Service.'" That's what S3 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." I mean, there's been lots of problems with exposed AWS S3 buckets in the past.

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 between 3 and 63 characters, because that's exactly what it is.

For example, I have an Amazon bucket named grc. 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've seen many examples where someone mistakenly, though presumably at some point for some purpose, deliberately allowed global read or read/write access to one of their S3 buckets, and disaster soon followed.

MIKAH: It shouldn't even be an option; right? Like why would anyone ever want...

Steve: Well, read/write I would agree, although you might want people to be able to put information up, like to submit things to you. And certainly, if you were, for example, a certain SSLVPN appliance vendor, you might want to have your appliance querying that S3 bucket by name, whatever name you've given it, to pull down updates, or security configuration improvements or something.

MIKAH: That makes sense.

Steve: So you're basically using it like a globally accessible CDN, a content delivery network. So actually that's how many people use it.

So okay. As a consequence, I have a bunch of S3 buckets with many wonderful, simple, and fun names since I got there early, and I grabbed them. Assignment of S3 bucket names could not be any simpler. It's as simple as first come, first served, like Twitter handles were back in the day. If you attempted 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, this is known as 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, simply by appending `.s3.amazonaws.com` to the end of 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 through z, or numeric digits from 0 through 9. You can also have dots and hyphens. So just like domain names. They also must begin and end with a letter or a number, meaning they cannot begin and end with dashes, and they cannot contain a pair of adjacent dots. There are also a bunch of specially reserved prefixes and suffixes that Amazon has, but like for things like Punycode and things, you know, in order to use a larger spelling alphabet. But overall, anything that anyone wishes to use will be valid within those guidelines.

And notice that I keep saying that any bucket 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 that 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 and returned to the available bucket pool, and become available for use by anyone who wishes to have it. By name.

So now we see what these watchTower 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 - the content within those previously deleted buckets.

So when these researchers discovered and recreated 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 what it was that was being requested, they could readily choose to return whatever malicious content of that form they might wish. Since these requests were being made over TCP connections, the true IP address of the entities making the requests could be determined.

They wrote, remember, 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 - NASA, laboratories, state governments, et cetera - the UK, Poland, Australia, South Korea, Turkey, Taiwan, Chile, and more." You know, on and on, universities, instant messaging software companies, cyber security technology companies, casinos, and more. Credit processing companies, you know, you name it. So this is not the result - and this is the key. This is not the result of a bug. This was the result of a fundamentally poor system design. Amazon should never have allowed bucket names to be recycled and reused. And in fact right now they ought to take any which were ever in use...

MIKAH: Yup, sinkhole them.

Steve: ...and sinkhole them. Just take them offline. And really, when you think about it, bucket names are really just vanity; right? They're like license plates.

MIKAH: Yeah, that's what I was thinking. Yeah. What does it matter?

Steve: Yeah. 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. You know, anybody could guess I might have grc. Well, I do. So, yeah. And I'll admit it. I'd rather have "grc" than 092D7630B5F. You know? Much sexier. But since S3 buckets are almost always accessed by automation, names were really never even necessary. They were just for 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. Once taken, can never be used again.

Okay. I have "grc," and that should be it forever. But it probably won't be. Unless they change their policy - which after all they could at any time - my use of GRC, those three initials, will eventually end because I will eventually end. You know? Whether I do it deliberately or not, I'll cancel my longstanding AWS account, or it'll be canceled posthumously. At that time, the account's data will be deleted, and the "grc" bucket, the bucket named "grc," will be recycled back into the available pool, ready to be used again by someone.

I have only ever used S3 as an off-premises encrypted storage archive. The danger for me has never been present. You know, there's nothing for anybody to ask for. But the guys at watchTower discovered that many S3 users are using once-used, in other words, I meant once used 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's appropriately been called. But Amazon has been slow to come around. Change is always difficult. But the good news is, and the reason we're talking about this today, is that last Thursday, 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. Short, it's just quick. They said: "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."

And 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 any not-yet-created bucket names. The Amazon S3 namespace is flat, as I said, not hierarchical. If I own the domain, as I do, "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." They're 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 has always used.

So, for example, say that an organization had the practice of saving their annual archives into a series of buckets named "Acme-Enterprises-Archive-2024." Then the next year "Acme-Enterprises-Archive-2025" and so on, where the year obviously is incremented successively. If some begrudging ex-IT employee...

MIKAH: Uh-huh.

Steve: You know where we're going - wished to cause their ex-employer "Acme-Enterprises" some grief, nothing prevents them, the ex-employee, from opening an

Amazon S3 account, which costs nothing, 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 name-space shared by every S3 user may have once seemed simple and maybe even fun, but there's a better way. So Amazon's addition of what they're now calling "account regional namespaces" is a long-needed addition.

Their announcement continues to explain this, writing: "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'" - okay, not particularly sexy, but okay. "That would exist in an account regional namespace." They said: "'Ourbucket' is the bucket name prefix specified. Then we add the account regional suffix to the requested bucket name," which is that -123456789012-us-east-1-an. "If another account tries to create buckets using this account suffix" - and lord, why would they? - "their requests will be automatically rejected." So that's the good news. You get protection against somebody trying to create a bucket with your specific account number suffix.

They finish, saying: "Your security teams can use AWS Identity and Access Management 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."

Okay. So I would argue that the implementation of this is something of a kludge. The total length of the bucket name prefix plus the regional account suffix is 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 a bucket which uses that suffix. And I guess a third thing they've done is they've allowed you to set policies, upper management, the IT overlords, can set policies that require all new buckets created to use this account regional namespace suffix. So they finally address this problem.

Notice that it is incumbent on the organization to turn this on. This doesn't do anything retroactively, retroactively. All of that is still a problem, that is, non-regional suffix buckets presumably still exist and work, but only newly ones created. So this is just enforcing a new bucket creation policy, bucket creation naming policy, on future bucket creation. So unless Amazon also decides to prohibit the recycling of previous bucket names, and they ought to just take - they ought to immediately blacklist any that have been abandoned right now. They're still doing nothing to prevent "bucketsquatting" on earlier buckets, which these guys may not have found.

They had to do some sort of brute force scanning to find all of these buckets. There are probably other buckets that still exist that have not been found. So, you know, the good news is, if organizations adopt and enable this enforced account number regional suffix bucket naming, then at least going forward the problem will have been prevented. And that is bucketsquatting.

MIKAH: That is far more terrifying than I thought it was going to be based on the name. I will say that.

Steve: Yeah.

MIKAH: It's pretty prevalent across the web. And the fact that they...

Steve: It's frightening, yeah, how widespread this is.

MIKAH: There's a level of creativity among security researchers that I really find to be quite incredible, when sometimes you'll be talking about these different exploits and ideas, and I think about the thinking required to get to that endpoint; right?

Steve: Yeah.

MIKAH: And it's, you know, despite being on its face a more practical, more sort of scientific approach, there's a lot of creativity that's involved in the work that you all do, and I think that that's something that really stands out to me at these times because I just - yeah, you have to think about the different tools that we use that are out there and then go, as you like to say, what could possibly go wrong? And that list can be hundreds of thousands of things, and you just highlight the one, oh, yeah, let's check this, we'll see if that works. This is just a cool thing to think of. But it's a terrifying outcome wherever it comes out to be true; right?

Steve: Yeah. Yeah. Yeah, we're still just - on one hand I think we're dragging legacy design and policy forward. And it's also the case that, I mean, because AWS was created after the Internet was mature. We already, I mean, you know, as a cloud-based service. Yet these guys didn't stop to think, well, what if somebody uses automated agents to pull code from buckets and, like, firewalls or SSL VPN appliances that are checking for new firmware, and then they go out of business, or they decide we don't like this bucket. We're going to move that in-house. But there's all this equipment out there that is still pulling from a bucket which has been abandoned, but some bad guy could then register and provide their own download for all this equipment to download. I mean, this is a real problem.

MIKAH: Yeah, they actually proved it to be a real problem.

Steve: Yes, yes.

MIKAH: And that's terrifying. I'm glad that it's, you know, again, we're shining a light on it. That's the first step.

Steve: And it's the good guys who are finding it and not the bad guys because it could be exploited. Unfortunately, there appear to be no lack of problems that the bad guys are finding and exploiting, too.

MIKAH: There are always so many things on that list that we talked about earlier. So Steve, I'm glad that you have your eye on the prize, and others out there are paying attention, as well.

Copyright (c) 2014 by Steve Gibson and Leo Laporte. SOME RIGHTS RESERVED

This work is licensed for the good of the Internet Community under the Creative Commons License v2.5. See the following Web page for details:
<http://creativecommons.org/licenses/by-nc-sa/2.5/>