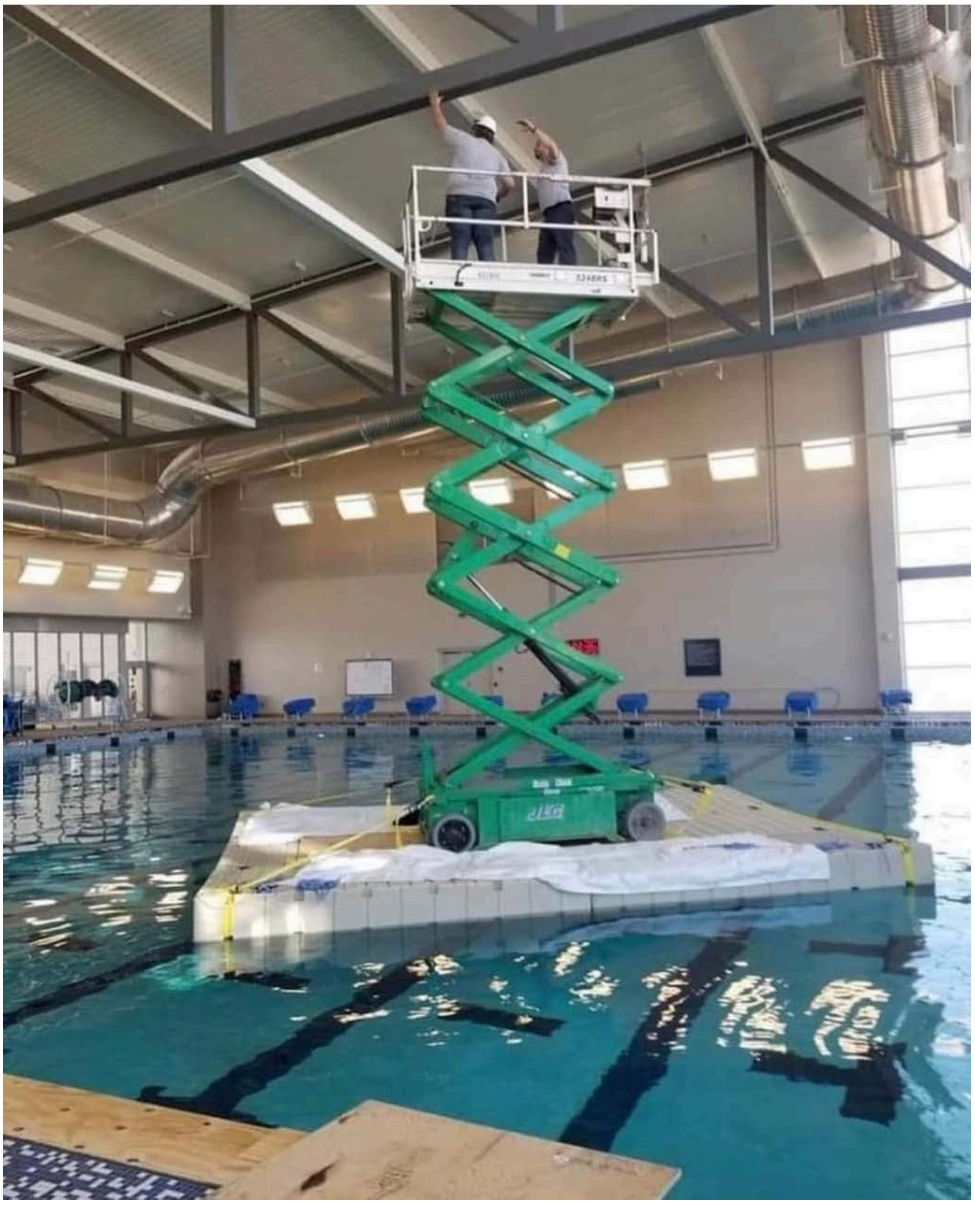
Security Now! #1009 - 01-21-25

Attacking TOTP

This week on Security Now!

What do we learn from January's record breaking 0-day critical Patch Tuesday? Microsoft to "force-install" a new Outlook into all Windows 10 and 11 desktops? GoDaddy is required to get much more serious about its hosting security. More age verification enforcement is coming, including globally. What another instance of a widely exposed management interface teaches us. DJI drone's official firmware update lifts geofencing for unrestricted flight. CISA's efforts pay off with MUCH improved critical infrastructure security. Listener feedback about TOTP, HOTP and age-verification. And we take a deep dive into cracking authenticator keys.

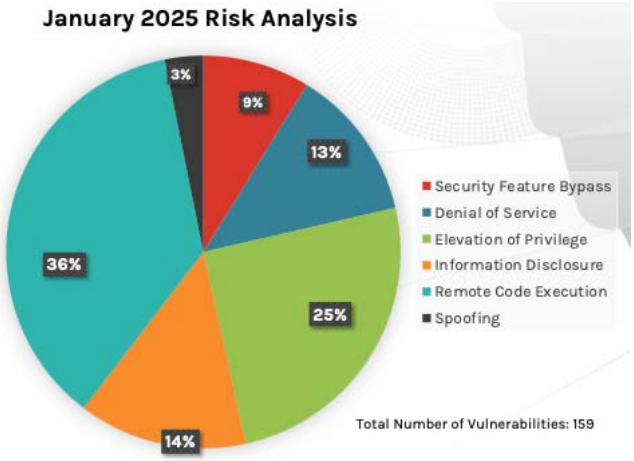
So how exactly do you propose
we get up there to fix that?



Security News

Patch Tuesday

CrowdStrike's blog was titled "January 2025 Patch Tuesday: 10 Critical Vulnerabilities and Eight Zero-Days Among 159 CVEs". The article noted that "This month's leading risk type by exploitation technique is remote code execution (RCE) with 36%, followed by elevation of privilege (25%)."



Unfortunately, of all the vulnerability classes, we know that the two most powerful and desired by the bad guys are remote code execution and elevation of privilege – which were by far the two most prevalent that were found and fixed by last week's updates. At 36% and 25% respectively, together that's 61% that were of the most serious kind. Elevation of privilege allows someone who arranges to get into a system as a regular and somewhat constrained user to bypass the operating system's privilege strictures. And remote code execution can both create that initial entry into a system, and once privilege has been elevated, allow the bad guys to run code of their choice to wreak havoc.

Viewed by product, Windows itself received 132 of the patches, and somewhat chillingly, Microsoft's ESU — as in Extended Security Updates — for previous Windows operating systems that no longer receive free patches and must have these fixes for Microsoft's software flaws purchased received 95. And in distant 3rd place was Microsoft Office with a relatively sedate 19 patches. It's interesting that current Windows received 132 patches, whereas older Windows, which Microsoft has stopped fussing with, was down at 95. Which Windows is objectively safer to use?

It's so easy to become numb to the idea that these vulnerabilities are being actively exploited. This means that there are serious campaigns in the world that are investing heavily in the discovery of these subtle vulnerabilities, then deploying exploits to take advantage of them in the real world.

Windows Hyper-V NT Kernel Integration VSP received three patches all having a severity of "Important" and a CVSS score of 7.8. The three are elevation of privilege (EoP) vulnerabilities allowing an attacker to gain SYSTEM privileges. Microsoft has indicated that the weaknesses are due to heap-based buffer overflow but has not shared details of the vulnerabilities or the source of disclosure.

Microsoft Office Access received patches for another 3, all having the same severity of Important and the same CVSS score of 7.8 ... but all three of these are remote code execution vulnerabilities exploited by opening specially crafted Microsoft Access documents. Microsoft

addressed this attack vector by blocking access to certain types of extensions in addition to patching the vulnerabilities.

There were three critical-rated 9.8 problems:

The first was a Critical RCE vulnerability affecting Windows Reliable Multicast Transport Driver (RMCAST) and has a CVSS score of 9.8. An unauthenticated attacker can exploit this vulnerability by sending specially crafted packets to a Windows Pragmatic General Multicast (PGM) open socket on the server, without any user interaction. However, exploitation is only possible if a program is actively listening on a PGM port. The vulnerability is not exploitable if PGM is installed or enabled but no programs are listening as receivers. Since PGM does not authenticate requests, it's crucial to protect access to any open ports at the network level, such as with a firewall. It is strongly advised to avoid exposing a PGM receiver to the public internet due to these security risks.

I haven't dug in to see how likely it is that a machine might have this port publicly exposed, nor what services might be listening for incoming traffic there. But it's clear from its 9.8 rating, and that it's a remote code execution, if those conditions were met the result would be bad.

The second of three critical-rated 9.8 RCE's seems much more worrisome, since it affects Windows old OLE (Object Linking and Embedding) technology which allows embedding and linking to other documents and objects. In an email attack scenario, an attacker could exploit this vulnerability by sending a specially crafted email to the victim. Exploitation of this vulnerability might involve either a victim opening the specially crafted email with an affected version of Microsoft Outlook software, or a victim's Outlook application displaying a preview of the specially crafted email. This could result in the attacker executing remote code on the victim's machine. Yikes.

Given OLE's age, my guess was that this would have been one of those vulnerabilities that Microsoft would have required payment for fixing on their older, yet still vulnerable machines. And, indeed, they list Windows Server 2008 and 2012 among the vulnerable systems. Since Server 2008 and 2012 are the equivalent of desktop Windows 7 and Windows 8, I'd bet that those desktops are vulnerable to this as well.

Microsoft's workaround advice is to only view your email as plain text so that Outlook's HTML viewer will not have the chance to invoke OLE for the display of content which, due to this very old bug in Windows OLE, would allow a PC to be taken over just by previewing email. This is why it seems wrong to me that Microsoft wants to sell the patch for this bug. How is that okay? What they want us to do instead, is to force the move to a newer operating system that has arbitrarily decided that it may not support the hardware we have, and which just had significantly more newly introduced vulnerabilities patched, compared to their older operating systems that are being allowed to settle down because they have stopped "making them better."

The third critical 9.8 vulnerability is a trivial to exploit elevation of privilege in good old NT Lan Manager v1. It's remotely exploitable across the internet and its low attack complexity means that attackers need minimal system knowledge and can consistently succeed with their payload against the vulnerable component. To eliminate the danger entirely, do not expose any Lan Manager network ports to the Internet. I've been saying for many years that there is no safe way to expose any of Microsoft's networking services, other than their web server, to the Internet. They have all been found vulnerable over and over and over. If this "simply don't do it" admonition is not useful to you because your applications or needs leave you with no choice, Microsoft says that the danger can be mitigated by setting Window's "LmCompatibilityLevel" to its maximum value of FIVE on all machines. This forcibly disables both LanMan and NTLMv1,

allowing only the use of NTLMv2. We've talked about how this, too, can be a problem in heterogeneous environments where Windows machines must communicate with older legacy equipment that, for whatever reason, cannot be updated. Not surprisingly, many such situations exist in the real world.

The simplest possible solution is to use IP address filtering where only the IP packets of specific remote machines, filtered by their IP address, are allowed to see the older and less secure Windows protocols. This does make the resulting network slightly more brittle, since firewall rules would need updating in the event of IP addresses changing. But it is such a simple and bulletproof solution.

Before I leave last week's Patch Tuesday topic, I should mention a pair of remaining critical remote code execution vulnerabilities which received CVSS scores of 8.1. Despite being remotely exploitable across the Internet, they were spared the same "hair on fire" 9.8 rating because their attack complexity was high. But the bad news is, they both exist in Windows Remote Desktop Gateway where we've seen problems before, and which many enterprises believe they have no choice other than to expose to the public Internet. I would argue that there are always ways around that. But first one needs to care enough to do so.

To exploit these vulnerabilities, an attacker needs to win a race condition by precisely timing their actions. That may be difficult, but most such Remote Desktop Gateways sit unattended and unmonitored. So attackers can try and retry without limit until they succeed. The attack involves connecting to a system running the Remote Desktop Gateway role, then triggering the race condition to create a use-after-free scenario. If successful, the attacker can leverage this to execute arbitrary code on the target system. Given the patches available, it appears that this problem was introduced in the Server 2012 time frame since Server 2008 is not affected.

I certainly understand that, once bitten, large enterprises will be understandably wary of Windows Updates bringing down any of their important applications and infrastructure. It's a devil's bargain. So the best enterprises can do is give each 2nd Tuesday's updates immediate attention and get the updates deployed as quickly as practical.

But, that said, the smarter thing to do, rather than always being reactive, is to really spend some time arranging to not be vulnerable to most of these problems in the first place by placing some other form of additional access control and/or authentication in front of anything having offering secured public access and exposure. Web and email servers are meant to receive anonymous connections from the public Internet. Pretty much nothing else is. What we keep seeing is that in-built authentication for any other private services is not trustworthy and can not and should not be trusted. Once something other than Windows itself is protecting Windows services, none of this stream of 0-day actively-being-exploited-in-the-wild vulnerabilities will be a source of concern. That's where you want to be. It's really worth spending some time thinking about how to get to that place.

New Outlook to be "force installed" on Windows 10 and 11

Before we leave Microsoft, I wanted to give a heads-up to our listeners about the forthcoming so-called "New Outlook" for Windows. The first I saw of this was a piece of news that said *"Microsoft will force-install a new Outlook email client on both Windows 10 and Windows 11 on February 11 and January 28, respectively."* That news blurb then posted a quote which read: *"Currently, there is no way to block the new Outlook from being installed - if you prefer not to have new Outlook show up on your organization's devices, you can remove it after it's installed as part of the update."*

A bit of poking around revealed that the sharp folks at BleepingComputer were on top of this. Under their similar headline *"Microsoft to force install new Outlook on Windows 10 PCs in February"*, they wrote:

Microsoft will force install the new Outlook email client on Windows 10 systems starting with next month's security update. The announcement was made in a new message added to the company's Microsoft 365 Admin Center, tagged MC976059, and it applies to Microsoft 365 apps users. As Redmond explains, the new Outlook app will be installed on Windows 10 devices for users who deploy the optional January 28 update and force installed for all who install the February 11 security update. The new Outlook client will run alongside the classic Outlook app and will not modify configurations or user defaults. Microsoft added that there's no way to block it from being installed on Windows 10 devices; however, those who don't want it can remove it afterward.

Microsoft wrote: "New Outlook exists as an installed app on the device. For instance, it can be found in the Apps section of the Start Menu. It does not replace existing (classic) Outlook or change any configurations / user defaults. Both (classic) Outlook and New Outlook for Windows can run side by side. Currently, there is no way to block the new Outlook from being installed - if you prefer not to have new Outlook show up on your organization's devices, you can remove it after it's installed as part of the update," the company added in a support document updated on Thursday.

To remove the new Outlook app package after it's force installed on your Windows device, you can use the Remove-AppxProvisionedPackage cmdlet with the PackageName parameter value Microsoft.OutlookForWindows. This can be done by running the following command from a Windows PowerShell prompt and adding a new reg value:

PowerShell: Remove-AppxProvisionedPackage -AllUsers -Online -PackageName (Get-AppxPackage Microsoft.OutlookForWindows).PackageFullName

REG VALUE: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsUpdate\Orchestrator\UScheduler_Oobe\OutlookUpdate

Next, add a REG_SZ registry setting named BlockedOobeUpdaters with a value of ["MS_Outlook"]. After removing the Outlook package, Windows Updates will not reinstall the new Outlook client.

The first preview version of the new Outlook for Windows was introduced in May 2022. The app was generally available for personal accounts in September 2023 (via the September 26 Windows fall update and the Microsoft Store on Windows 11) and for commercial customers in August 2024.

This doesn't seem like the end of the world, but some people may object to having Microsoft force-installing a new and presumably unwanted Outlook client onto their machines. This new client is apparently based upon the web version. It's essentially a port of the web client to a native Windows app. As such, it does not support Outlook's traditional (and problematic) PST file format and it also does not support any COM (component object model) integration with Outlook. I also noticed that Microsoft says that unlike traditional Outlook for Windows, the New Outlook offers "limited" support for 3rd-party email services such as Gmail, Yahoo! And so forth.

I'll take this opportunity to mention that I recently switched away from Mozilla's Thunderbird as my email client.

For years and years, before being driven to Thunderbird, my original true blue email client had been Qualcomm's Eudora – and life was good. I didn't care when Qualcomm's support for Eudora ended, because Eudora worked perfectly. But over time, as other email clients' behavior changed, cracks began forming. Email started coming in with high-ASCII or Unicode capital "A's" with umlauts added to spaces. For a year or so I manually edited them out of every reply that I was quoting until several years ago I finally decided to switch to Thunderbird.

I then used Thunderbird for several years and, truth be told, I've never really been happy with it. I'm finicky about the appearance of my outbound mail – as I am about pretty much everything I produce – and Thunderbird's handling of fonts and formatting, indentation of email threads and the signatures it appends to email never made sense to me. It was trying to handle formatting details, but it made things mysterious and deliberately uneditable. I wasn't allowed to fix these things when they didn't look the way I wanted them to because their formatting was not only erroneous but automatic. Thunderbird's editor apparently believed that it knew better than I how things should be.

Finally, two weeks ago, something drove me to seek another email client. I already had an older copy of "The Bat!" installed, which I briefly attempted to resurrect but it wasn't much improvement. So I went cruising around various of the "Top 10 best email clients" lineups until I stumbled upon one I'd never heard of before named "eM Client" and life is good once more.

It's difficult for me to explain exactly why this is so, but I can say that after setting it up as an IMAP client and allowing it to synchronize with GRC's email server I almost immediately felt that I had a handle on my email. It found back and forth email from before and knitted them into threads. It allows me to mark things in various names and colored tags and to then view all of those as if they were all in their own dynamic folder. I can also see all of the inboxes of my various accounts consolidated into a single view, and – thank goodness! – it doesn't do **ANY** mysterious, unwanted and wrong things with the nesting of replies. Since my needs are not necessarily aligned with everyone else's I'll share a broader overview from Wikipedia:

eM Client has a range of features for handling email, including advanced rules management, mass mail, delayed send, or a built-in translator for incoming and outgoing messages. It supports signatures, Quick Text, and tagging and categorization for easy searching. Watch for Replies and Snooze Email functions are available, as well as direct cloud attachments from cloud services like Dropbox, Google Drive, OneDrive, ownCloud or Nextcloud.

eM Client also provides a lookup service for GnuPG public keys (eM Keybook) in order to more easily send encrypted communication via email, and generally simplify PGP encryption in email communication.

eM Client supports all major email platforms including Exchange, Gmail, Google Workspace, Office365, iCloud and any POP3, SMTP, IMAP or CalDAV server. Automatic setup works for Gmail, Exchange, Office 365, Outlook, iCloud, or other major email services. Following the shut down of IncrediMail, an auto-import option was added to transfer data from this platform to eM Client. Since version v8.2, eM Client supports online meetings via Zoom, Microsoft Teams, and Google Meet). eM Client allows extensive appearance customization. eM Client 10, released in 2024, also provides AI features for composing messages and replies, Inbox categories and Quick Actions which allow users to create their own macros.

My only complaint is that the free version will only handle a single email account and I need at least four. And that would be okay if I could purchase a paid version once. But it's "rental ware" only available for $40 per year. I rent no other software of any kind, and that's something I actively fight against. So this is the first time I've ever capitulated. But at $3.33 per month, allowing installation on three machines, the experience of using this client continues to impress me. And if paying something is what's required to keep this stunning creation alive and maintained, then I'd rather do that than not have access to it. I didn't realize how unhappy I had been with Thunderbird until I began using eM Client. The mobile editions are available at no charge and I can't vouch for anything other than their Windows edition which is all I've used, but it's available for macOS, iOS and Android and they claim to be in use in over 100,000 businesses and by 2.5 million users.

For anyone who might be seeking a similar improvement to a major aspect of their lives, eM Client is available for download as a feature-complete 30-day trial. I've been tweaking it here and there, removing displayed columns and such and I could not be happier. It's also possible to export all of those tweaks and preference settings to an XML file and then import them into another instance of eM Client. So as I've been moving back and forth among machines I've been able to keep all of the instances looking and operating the same.

I wanted to pass this along in case any of our listeners might be wishing for something better. This might be it. You can find it at www.emclient.com. This is not intended to be any sort of comprehensive review, and my sense is that everyone's needs and tastes are so different that no one else's option would or should ever form more than a pointer. I need very few of the other features it offers. I really just need a multi-account IMAP client for GRC. But the various display options, the threading of conversations, and the utterly perfect experience of composing and replying to messages has me hooked.

GoDaddy must get more serious about security

We've previously covered the various security troubles with GoDaddy's web hosting services. The sense I've had is that adding web hosting was an afterthought behind their domain name services and that's what got them in trouble.

The news is that the U.S. Federal Trade Commission has decided to require GoDaddy to clean up its act. Last Wednesday the FTC announced that GoDaddy will be required to bolster its cybersecurity program to address years-long deficiencies. The FTC stated that GoDaddy's failure to use industry standard security measures led to what the FTC called "several major security breaches" between 2019 and 2022. The agency also alleges that GoDaddy deceived its customers about how adequately it safeguards its web hosting product. The agency said that consumers were sent to malicious websites and otherwise harmed after hackers broke into GoDaddy customers' websites and accessed their data.

The extensive information security measures which the FTC is requiring GoDaddy adopt are similar to the reforms the agency ordered Marriott to implement after the hotel chain failed to improve its cybersecurity posture despite being breached three times between 2014 and 2020.

In a statement explaining why the FTC acted, Samuel Levine, Director of the FTC's Bureau of Consumer Protection, said "Millions of companies, particularly small businesses, rely on web hosting providers like GoDaddy to secure the websites that they and their customers rely on."

GoDaddy, which has about five million web hosting clients, failed to track and manage software updates, analyze threats to its shared hosting services, properly log and continuously assess cybersecurity incidents, and silo its shared hosting from more insecure platforms.

GoDaddy also falsely advertised that it prioritized a strong security program and complied with international frameworks requiring companies take "reasonable" measures to protect personal data. Consequently, the proposed settlement order bars GoDaddy from exaggerating its security practices; orders it to design a "comprehensive" information-security program; and directs it to retain an outside company to assess its enhanced cybersecurity program when it launches and every two years thereafter.

It's interesting that the reporting about this referred to the infamous Marriott Hotels / Starwood breach incident. What we recall from that is that Marriott acquired the independent Starwood group whose network security was a lackluster afterthought. But Marriott never took the time to thoroughly vet what they were purchasing and that lack of oversight over their purchase came back to bite them.

GoDaddy's past is similar inasmuch as it has grown into the behemoth it is today through a long series of mergers and acquisitions – buying up and consolidating independent Internet registrars. And I recall that their web hosting business was the result of one or more similar acquisitions. So, much like Marriott, they purchased something that needed work, and was then bitten when their name became tied to that new acquisition's poor security.

I'm sure there's a lesson here for any large organization that purchases another high-tech entity. The purchase negotiation should include a very thorough and deep independent 3rd-party review of that soon-to-be-acquired company's security practices. For one thing, the enforcement of true security can be expensive. Which means that an entity's true bottom line profit may be inflated due to a lack of sufficient security. Since any missing security practices would need to be added afterward, a better purchase price might be negotiated once its lack had become apparent, and in any event, the buyer will have a better idea about the potential liability that might come along as part of the package.

More impending age verification

I saw a news item that indicated that the U.S. Supreme Court appeared to be poised to support the enforcement of age verification for adult-content websites. The determination being made was whether more than 1/3rd of the site's content contained adult-oriented material. And, if so, any such websites would be forced to affirmatively verify any visitor's age before they would be able to view that site's content. Since we had just discussed this issue last week, I decided not to bring it up again. But then I ran across some news from across the pond about what's about to transpire in the United Kingdom. And since the verification of age is a sticky wicket, I decided to share the news from the UK. The publication "The Record" reported the following last Thursday:

The United Kingdom's communications regulator Ofcom announced on Thursday that online pornography sites must, by July, verify that all of their users are adults or potentially face being blocked by the country's internet service providers.

James Baker of the Open Rights civil liberties group expressed concerns that "the roll-out of age-verification is likely to create new cybersecurity risks in the form of additional scam porn sites that will trick users into handing over personal data to 'verify their age'."

Ofcom has set out a range of methods that it considers highly effective for checking users' ages, including photo ID matching and checks on credit cards — which you must be 18 to own in Britain. Other age-checking methods could be acceptable, said Ofcom, but they "must be technically accurate, robust, reliable and fair in order to be considered highly effective."

*Specifically, the regulator has stated that the self-declaration of age and online payments using a debit card — which do not require a person to be 18 — would **not** be considered effective, and could leave those sites open to enforcement action.*

James Baker said: "Some of the verification methods that Ofcom has defined as highly effective could put people at risk of new cybercrimes" citing research published with the Electronic Frontier Foundation.

The age verification measures are part of Britain's controversial Online Safety Act, which passed in 2023 and aims to force technology companies to address a range of online harms. Businesses that fail to comply could face a range of enforcement actions, from being fined up to £18 million ($22.3 million) or 10% of their global revenue, having their websites blocked by British internet service providers, or even facing criminal prosecution.

For their part, Ofcom's chief executive, Melanie Dawes, said: "For too long, many online services which allow porn and other harmful material have ignored the fact that children are accessing their services. Either they don't ask or, when they do, the checks are minimal and easy to avoid. [Yeah… Like clicking the "Yes, I'm 18" button.] *That means companies have effectively been treating all users as if they're adults, leaving children potentially exposed to pornography and other types of harmful content.*

Melanie Dawes, said: "As age checks start to roll out in the coming months, adults will start to notice a difference in how they access certain online services. Services which host their own pornography must start to introduce age checks immediately, while other user-to-user services – including social media - which allow pornography and certain other types of content harmful to children will have to follow suit by July at the latest."

Baker of the Open Rights Group said: "There needs to be a specific and enforceable guarantee that age verification systems will be private, safe and secure. The new plans miss this vital step, so place people at risk of data leaks, and having their sexual interests exposed to blackmailers and scammers."

So I'd say it's very safe to conclude that the handwriting is on the wall here. Like it or not, both in the U.S. and the U.K., true age verification – more than pressing a button – are coming soon to the Internet. And it's worth noting that whereas it's very difficult for any regulator to ascertain the effective network security of any given organization, it could hardly be any easier for regulators to determine whether a website is verifying the ages of its visitors. Just go there from any anonymous IP and see what happens.

"Console Chaos"

Reinforcing the point I made about never relying upon any single manufacturer's public-facing remote access authentication, the security of the FortiNet security appliance has once again been found wanting. In a posting on the Arctic Wolf security firm's website, titled *"Console Chaos: A Campaign Targeting Publicly Exposed Management Interfaces on Fortinet FortiGate Firewalls"* they listed four key takeaways:

- *Arctic Wolf observed a recent campaign affecting Fortinet FortiGate firewall devices with management interfaces exposed on the public internet.*
  [Everyone heard that, right? "with management interfaces exposed to the public Internet".]

- *The campaign involved unauthorized administrative logins on management interfaces of firewalls, creation of new accounts, SSL VPN authentication through those accounts, and various other configuration changes.*

- *While the initial access vector is not definitively confirmed, a zero-day vulnerability is highly probable.* [That has since been confirmed.]

- *Organizations should urgently disable firewall management access on public interfaces as soon as possible.*

That final point: "*Organizations should urgently disable firewall management access on public interfaces as soon as possible.*" They meant that to apply to this instance. But it ought to be general practice. The phrase "public-facing management interfaces" should never occur.

I forgot to mention that this is so serious that CISA and multiple cybersecurity firms warned of a zero-day vulnerability in FortiGate firewalls that hackers are actively exploiting. CISA ordered all federal civilian agencies to patch the vulnerability by today, January 21, one of the shortest deadlines ever issued. And Fortinet said in an advisory that the bug **is** being exploited in the wild but did not say how many customers had been impacted. The company said threat actors attacking organizations with the vulnerability are creating administrative accounts on targeted devices and changing settings related to firewall policies.

Patching as soon as possible is the responsibility of the owner of the device. But this was being exploited before any problem was known and before any patches were available. Secure remote access to a device such as this is entirely possible but it should never rely solely upon the manufacturer's account logon protections; always add your own independent layer of authentication.

What's up with DJI lifting firmware-enforced drone geofencing?
I posed the introduction of this next surprising bit of news as a question; "What's up with DJI lifting firmware-enforced drone geofencing?" – and I'll follow that with "and is it, really?"

I was put onto this but a short one-liner in the Risky Business news, which said, simply: *"DJI gives the middle finger to US: Facing an impending ban in the US, Chinese drone maker DJI has removed firmware restrictions preventing its drones from entering no-fly zones."* So I thought, "Whoa! If true, I didn't see that coming and that's no way to smoke the peace pipe with authorities in the U.S." The Risky Business news then provided a screenshot of posting by Matthew Stoller on BlueSky Social, which read:

Matthew was kind enough to link to the source of his news, a posting on the Viewpoints blog at none other than DJI.COM: https://viewpoints.dji.com/blog/geo-system-update. Viewpoints bills itself as the official DJI blog.

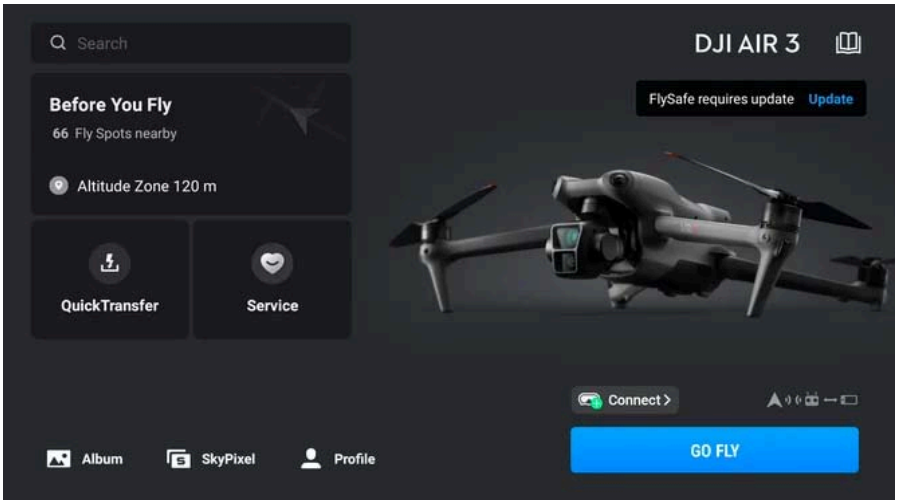
Last week's DJI blog posting is titled: "DJI Updates GEO System in U.S. Consumer & Enterprise Drones" and the posting says:

The update follows changes in Europe in 2024 and aligns with FAA Remote ID objectives.

DJI has announced updates to its geofencing system (GEO) which applies to most of its consumer and enterprise drone products in the United States (U.S.). These changes will take effect starting from January 13 on both the DJI Fly and DJI Pilot flight apps. This update follows similar changes implemented in the European Union (EU) last year.

With this update, DJI's Fly and Pilot flight app operators will see prior DJI geofencing datasets replaced to display official Federal Aviation Administration (FAA) data. Areas previously defined as Restricted Zones (also known as No-Fly Zones) will be displayed as Enhanced Warning Zones, aligning with the FAA's designated areas. In these zones, in-app alerts will notify operators flying near FAA designated controlled airspace, placing control back in the hands of the drone operators, in line with regulatory principles of the operator bearing final responsibility.

To update, operators need to connect their flight app to the internet and click 'Update' on the FlySafe pop-up notification (see image below).



When DJI first introduced the GEO system in 2013, consumer drones were still a relatively novel technology, and formal drone flight rules and regulations were sparse. The geofencing system was created as a voluntary built-in safety feature to help foster responsible flight practices and prevent DJI drone operators from unintentionally flying in restricted airspace, such as around government buildings, airports, or prisons.

For many years, DJI has led the drone industry in safety, making several unprecedented commitments to integrating advanced safety systems into its drones, including:

- *First to install altitude limits & GPS-based geofencing to guide drone pilots away from unsafe locations.*
- *First to deploy autonomous return-to-home technology if drones lose connection to their controllers or have critically low batteries.*
- *First to integrate sensors for nearby obstacles and approaching aircraft.*
- *First to operate Remote Identification technology to help authorities identify and monitor airborne drones..*

Since then, global regulations and user awareness have evolved significantly, with a greater focus on geo-awareness and Remote ID solutions which makes detection and enforcement much easier. National aviation authorities, including the European Aviation Safety Authority (EASA) in the EU, the UK Civil Aviation Authority (CAA), and the FAA in the U.S., have established comprehensive geographical zones for unmanned aircraft systems (UAS) and enforce drone regulations.

This GEO update has been active in the UK and several EU countries since January 2024, starting with European countries that have implemented geographical maps compliant with existing technical standards, such as Belgium, Germany, and France. In June, it expanded to Estonia, Finland, and Luxembourg. The remaining EU countries under EASA jurisdiction will also receive the update this month.

DJI reminds pilots to always ensure flights are conducted safely and in accordance with all local laws and regulations. For flights conducted in Enhanced Warning Zones, drone operators must obtain airspace authorization directly from the FAA and consult the FAA's No Drone Zone resource for further information.

While this posting from the early last week is far less inflammatory than the "middle finger" reference I first encountered, it does say exactly the same thing. Specifically here, where it says:

"With this update, DJI's Fly and Pilot flight app operators will see prior DJI geofencing datasets replaced to display official Federal Aviation Administration (FAA) data. Areas previously defined as Restricted Zones (also known as No-Fly Zones) will be displayed as Enhanced Warning Zones, aligning with the FAA's designated areas. In these zones, in-app alerts will notify operators flying near FAA designated controlled airspace, placing control back in the hands of the drone operators, in line with regulatory principles of the operator bearing final responsibility."

So, in other words, operators will be notified but the updated firmware will no longer prevent a DJI drone from flying right into and across what was previously designated as a No-Fly Zone.

Apparently, variations of the "middle finger" reference were widely picked up and circulated. And this prompted DJI to release a second blog posting later in the week, last Thursday. This second blog was titled *"DJI's GEO System Is An Education - Not Enforcement - Tool"*. It attempted to clarify DJI's position and mollify the critics. It said:

Earlier this week, we announced an update to the DJI geofencing system (GEO) in which prior DJI geofencing datasets in most of our consumer and enterprise drone products in the United States (U.S.) will be replaced with official Federal Aviation Administration (FAA) data.

We first introduced the GEO system in 2013, at a time when consumer drones were still a relatively novel technology and formal drone flight rules and regulations were sparse. This voluntary initiative helped foster responsible flight practices and provided valuable user education. Having been in place for over a decade, we understand why there are mixed sentiments surrounding this change.

However, some concerning reactions circulating online are either categorically false or seek to politicize this update given the current geopolitical climate. In the first Get the Facts article of the year, we want to take this opportunity to dispute the misinformation and set the record

straight.
***FACT 1:** Politics does not drive safety decisions at DJI. For over a decade, DJI has led the drone industry in safety, making several unprecedented commitments and investments to integrate advanced safety systems into our drones - often ahead of regulatory requirements and without being prompted by competitors. To suggest that this update is linked to the current political environment in the U.S. is not only false but also dangerous. Politicizing safety serves no one. We encourage discussions and comments to remain focused on technological facts and evidence. To understand the true reasons behind this update, read on.*

***FACT 2:** Aviation regulators around the world - including the FAA - have advanced the principle of operator responsibility. This GEO update aligns with and respects this principle. Similar updates to the GEO system began in the European Union (EU) last year, with no evidence of increased risk. We had planned to roll this update in the U.S. months ago but delayed the implementation to ensure the update would work properly.*

To add, over a decade has passed since DJI introduced the GEO system and regulators have not chosen to mandate geofencing, instead opting for solutions like Remote ID (which requires drones to broadcast the equivalent of a "license plate"), LAANC (automated drone flight approvals in controlled airspace near airports) and community-based training.

***FACT 3:** The GEO system has always been an educational - not an enforcement - tool. The GEO system has also not been removed; warning zones and in-app alerts remain in place to continue educating pilots on safe flight operations. This change gives back control to operators and provides them the information they need to fly safely.*

DJI remains committed to promoting safe and responsible flight practices and will continue its community education efforts, reminding pilots to always ensure their flights are conducted safely and in accordance with all local laws and regulations.

***FACT 4:** In addition to aligning with the FAA's operator responsibility-led principles, the update to "Enhanced Warning Zones" provides two operator benefits:*

- *Reduced operational delays for pilots. The previous "No Fly Zones" (NFZs) often placed an unnecessary burden on operators. While a user could receive instantaneous approval through LAANC to fly, they were still required to submit an application to DJI and wait for manual review and an unlocking license. This process could result in missed opportunities, delayed operations, or unnecessary waiting times. This was especially challenging for commercial operators, drone businesses - and most critically - public safety agencies performing lifesaving work, where delays are simply unacceptable.*

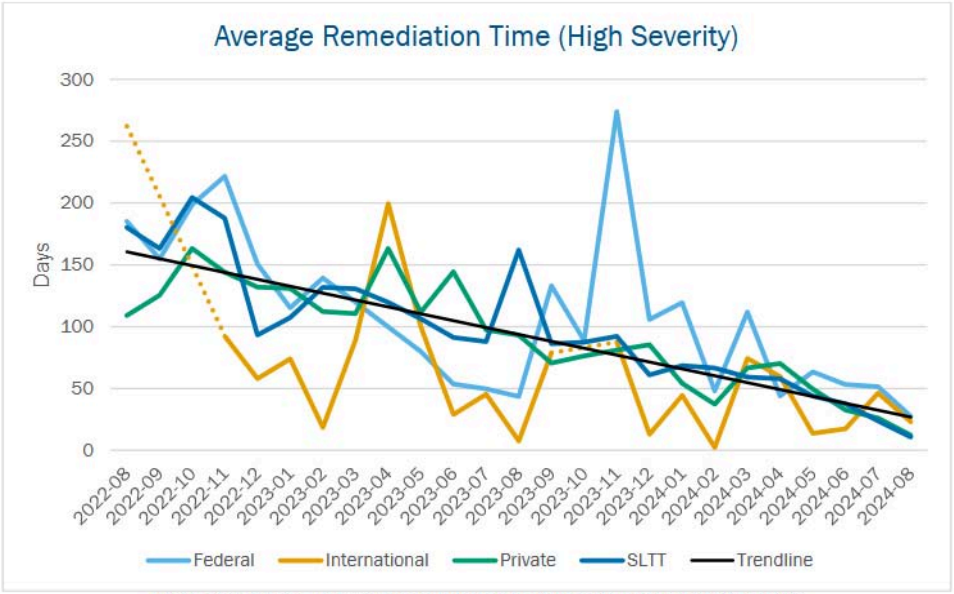
- *Improved consistency with official FAA data. Previously, the global geofencing system relied on ICAO Annex 14 configurations for airspace around airports, which did not always align with official FAA data. This mismatch caused confusion among operators unsure about where it was safe to fly. By displaying official FAA data, this update ensures operators can view airspace as the FAA intends, clearly understanding where they can and cannot fly.*

We hope this explanation clarifies the real reasons behind the update to the GEO system: an opportunity to align with regulatory principles, empower customers with greater control, and provide them with accurate, official information to confidently operate their drones within safe and permitted airspace.

Okay. So what appears clear, when all is said and done, is that whereas a DJI drone may have previously flatly refused to fly somewhere, that will no longer ever happen. Instead, the drone will be broadcasting an ID to identify the drone's owner/operator, and that owner/operator will be given clear notification and warning when they are approaching so-called "Enhanced Warning Zones." But the entire responsibility of whether or not they then choose to do so will be left to them.

Given the concerns and accusations that have been levied at DJI over the possible use of their high-quality camera-equipped drones for unwarranted surveillance, it's not a stretch to imagine the conspiracy theories that this would have triggered. And given the United State's current political climate toward China, which certainly is a thing, I have no idea what's really going on here. If nothing else, this would appear to be an inopportune time for DJI to remove its historically firmware-enforced No Fly system. But I thought this was interesting and I wanted our listeners to know this had happened.

CISA shows a HUGE improvement in vulnerability remediation:
In its recently published "Cybersecurity Performance Goals Adoption Report", CISA said that the number of critical infrastructure organizations enrolled in its vulnerability scanning service nearly doubled over a two-year period, reaching 7,791 organizations at the end of August 2024. CISA added 1200 vulnerabilities to its known exploited vulnerabilities catalog through the same period. During the two-year period of analysis, critical infrastructure organizations enrolled in CISA's vulnerability scanning service reduced average remediation times from 60 days to 30 days.



Note: Dotted lines represent gaps in collection that were interpolated for clarity.
Figure 5: Average Remediation Time of High-Severity KEVs

Followers of this podcast know firsthand that this is not a simple feat to pull off. This is especially true for any sort of large and lumbering bureaucratic organization. But this is truly looking like a significant change in security posture and active vulnerability reduction.

We talk about the work that CISA is doing more and more frequently because they're doing so many things surprisingly right. They really are having a huge effect by raising the awareness of cybersecurity as a crucial consideration for any and every organization, whether in the public or the private sector. We've come a long way during the 20 years of this podcast.

Closing the Loop

Earl Rodd

Other stats on 6 digit numbers that I think feed our psychological tendency to see "patterns" where there are none. Remembering that only 151,200 of the million have all 6 digits unique:

- *157,600 have at least 3 of the same digit (that's more than have 6 unique digits)*
- *395,200 (of the million) have 4 or fewer unique digits.*
- *409,510 have at least 2 consecutive digits the same!*

We've talked about the famous Birthday Paradox in the past. Given randomly distributed birthdays occurring throughout the year, we're surprised by how small a group of people is needed to get a better than 50/50 chance of there being a birthday collision, where any two in the group share the same birthday. When you think about it, the same thing is happening with our 6-digit authenticator codes: Here, we have six digits and only ten possibilities for each one of those six digit places. I think that the same sort of counter-intuitive experience occurs where the likelihood of inter-digit collisions is actually much greater than our intuition would predict. As with the surprising Birthday Paradox, every digit has a collision possibility with every other one.

I received a great piece of feedback from someone who's in the field trying to do the right thing. This is important because Microsoft, for all practical purposes, owns the enterprise world. This listener's feedback contains a bunch of Microsoft jargon that will mean something to our enterprise listeners. For everyone else, these details are not important because everyone will be able to understand the dilemma that our enterprises face:

Hi Steve, I'd like to remain anonymous. I'm 24 years old and have been a listener since around episode 900. I work as an IT Systems Administrator for a local government in North Carolina. One of my responsibilities is managing security for our city's police department. We are required to comply with the FBI's CJIS (Criminal Justice Information Services) Security Policy, which is updated regularly. I've included a link to the policy below; it's 451 pages long, and all law enforcement agencies must adhere to it and pass periodic audits: https://wilson.gjassets.com/content/uploads/2024/11/CJIS_Security_Policy_v5-9-5_20240709-1.pdf

Okay. So all of that sounds like the right thing so far. This clearly sets a high bar that's onerous to meet, but we know from everything we've seen that unless this level of specification and its enforcement-by-audit are applied, the *"everything appears to be working so let's not break it"* approach will be taken by default. Everyone has too much work to do and no one wants to go looking for trouble. And while first achieving compliance may be a heavy lift, once things have been tightened up to meet the audited requirements, remaining compliant should only require a much more modest effort. Our listener continues:

One requirement in the policy, found on page 97 (requirement #20), is especially challenging: "All secrets must be hashed and salted."

Okay. Now you might wonder why that would be especially challenging? After all, hashing and salting stored secrets, such as any passwords, has been standard operating procedure for a very long time. I didn't find the earliest reference to salting hashes in our transcripts, but, for example, I found a reference from 2012 where Leo and I were talking about it as if it was

something that everyone knew. I'd imagine that we were talking about this from the start. But I was curious for the sake of this discussion how old the idea of salting a hash for storing secrets was. So I asked the o1-mini model of ChatGPT the following question: *"what's the earliest appearance of the recommendation that stored passwords should both be hashed AND salted for secure storage?"* and I received the following reply:

The recommendation to store passwords using both hashing and salting has its roots in the late 1970s, primarily driven by the practical implementations in early operating systems and evolving security best practices.

UNIX Version 7 (1979): One of the earliest and most influential implementations of salted password hashing was introduced with UNIX Version 7 in 1979. This version of UNIX featured the crypt() function, which incorporated a 12-bit salt alongside the hashing process. The primary purpose of the salt was to ensure that identical passwords would result in different hash values, thereby mitigating the risk of attackers using precomputed tables (like rainbow tables) to reverse-engineer passwords from their hashes.

[Skipping over o1's completely correct explanation of the purpose of salting, it then added:]

Evolution in Security Practices: Following the implementation in UNIX, the practice of salting hashed passwords became a cornerstone in password security:

- *Early 1980s: Security literature and guidelines began to formally recommend the use of salts in conjunction with hashing to protect stored passwords.*
- *Subsequent Decades: As computing power increased and new attack vectors emerged, the methods for hashing (e.g., transitioning from DES-based hashing to more secure algorithms like bcrypt, scrypt, and Argon2), salting became more sophisticated, further strengthening password storage mechanisms.*

Key Takeaway: While the precise first recommendation in academic or security policy literature might be harder to pinpoint, the practical implementation of hashing with salting in UNIX Version 7 in 1979 marks the earliest prominent appearance of this security practice. This implementation set a standard that has been built upon and refined in subsequent years to enhance the security of stored passwords.

I couldn't have phrased any of that any better. And now we have a marker. This brings us back to our listener who quoted page 97 of the security requirements his IT systems were required to offer: *"All secrets must be hashed and salted."* which he said was especially challenging. He continued:

Like many small to medium-sized cities, we operate on a tight budget and are often behind on adopting the latest technologies. We still rely on Active Directory, which syncs with Microsoft Entra (formerly Azure AD) via Microsoft Entra Connect for managing Office 365 products and Exchange Online. ***However, Active Directory does not salt user password hashes, and it seems Microsoft has no plans to implement this feature.***

And our listener is correct. Active Directory is still using older LAN Manager or NT LAN Manager user passwords which have never incorporated salt. As we know, both of these technologies are horrifically old and insecure, yet they are still in use. So what are people supposed to do? Our listener continues:

From my research, Microsoft's suggested solution is to migrate entirely to the cloud with Entra ID (Azure AD), eliminating the need for on-premise domain controllers and moving all authentication to the cloud. Here's where we run into two major issues:

Limited Features in GCC: (Government Community Compliance)

We're on the GCC tenant of Microsoft 365, which lacks many features available to regular enterprise customers. I recall you mentioning the federal government's frustration with Microsoft; local governments face similar challenges. Information about feature differences between Enterprise, GCC, and GCC High is not easily accessible, especially from Microsoft.

We tested a full migration to Entra ID with Intune for device management, but Intune in GCC is noticeably less functional than in the enterprise environment. Many settings and options are greyed out, often with messages indicating that our tenant didn't contain the correct license.

[And then there are the] High Costs: Fully migrating to the cloud is expensive, with steep annual fees. It would require us to upgrade every user's license from Office 365 to Microsoft 365. Given the lack of features in GCC, it's hard to justify the additional cost.

So, my question is: For IT environments that still rely on on-premise Active Directory, what solutions are available to salt password hashes in AD? Thanks for your insight, and I appreciate all the work you do!

Unfortunately, this is where the expression "caught between a rock and a hard place" comes in. I'm not an expert on Microsoft's Enterprise offerings – for which I will be eternally grateful. But I poked around and nowhere could I find any solution for specifically adding salt to Active Directory passwords. There are all manner of enhanced security and authentication features, such as the Kerberos, but even there, Kerberos authentication uses the unsalted password stored by Active Directory.

On principled grounds, I so strongly dislike the idea of these blanket security requirements driving organizations into Microsoft's cloud services where they will be even more at Microsoft's mercy than they are today, and have even less recourse when Microsoft raises their rental rates. The only thing I can suggest is that an appeal be made proactively to the auditor to explain the situation, and ask what solutions other government organizations may have found. Has this single requirement driven everyone else into the cloud? Or is there a wink and a nod that allows this singular requirement to be quietly ignored?

Lawrence Fisher

Steve, In your segment re: DJI geofencing - you might want to mention the drone strike that occurred on the wing of one of the super-scooper fire aircraft fighting the Palisades fire last week. Hopefully things are not too windy at your place today!

I wanted to share Lawrence's reminder of this happening last week, since it reminds us that the problems caused by privately owned drones flying into hazardous areas is not fictional. The recovered drone was a DJI model. I don't know one way or the other whether the recent firmware update untethered the drone such that it was then able to fly where it wouldn't have been able to before. Temporary Flight Restrictions (TFRs) were in effect and those apply to all aircraft. If nothing else, any drone operator likely knows that.

Dean Weiten

Hi Steve, I have a suggestion for the Podcast. I'm a long time listener, not quite back to the beginning, but something like 16 years. I am a member of Club TWiT, and I do enjoy the respite from advertising. However, I would like to know which advertisers support the show, and maybe take advantage of special offers. For instance, for a VPN provider. Would Leo considering inserting a short "this podcast is supported by <company> which offers 15% off using promo code <promo code>", or whatever short announcement is appropriate pointing the listener to the show notes which might have full details, in place of each advertisement, instead of just cutting out the advertisement audio? Best regards, Dean, in Maryland.

I've sometimes found myself in a similar situation, so I discovered some time ago that TWiT maintains an easy-to-find "Sponsors" page at https://twit.tv/sponsors. You can also just go to TWiT.tv and it's in the menu at the top of the page. And the entries there include the special discount sponsor codes and URLs. So anyone can, at any time, check that out. And that way you'll also get information about TWiT sponsors other than those that only sponsor this podcast.

Errata

I have a piece of errata to share because my mistake was picked up by several of our listeners, who essentially asked variations of: "What do you mean 'SyncThing hardly ever updates?'" But this feedback from our listener **Brendan Coupe** offered some interesting additional information:

I'm catching up on last week's show and I was surprised to hear you say that Syncthing is rarely updated. I rarely use Windows and love Notepad++ but agree that at times it seems to update just to increase the version number. I think the developer sends political messages with some updates which is their right.

I have been a Syncthing user from way back when BitTorrent sync went from a useful free application to a mess with lots of restrictions. I stumbled onto Syncthing and have never looked back.

I have Syncthing running on more than 25 devices, including various Android phones and tablets. I have a half dozen backup servers running on Odroid HC-2 and HC-4 devices running Linux at various locations. It functions as a live backup system that syncs as files are changing. Most of the time there is a local server that should sync quickly while the offsite servers can catch up even if I shutdown the source device before the remote servers are synced up. I can also turn on my laptop when I use it and before long it matches my desktop computers. Not sure what I would do without Syncthing.

One thing I have not heard you talk about is self hosting the relay and discovery servers. I have been doing that since day one and have it running at 5 or 6 locations. I never rely on the public servers that Syncthing provides. TNO:-)

When I first started using Syncthing, it was very early in the development and it was a little rough around the edges. As I recall it used to update more than monthly and possibly more than weekly at times. A while back they switched to a monthly update cycle and it seems to update at the beginning of the month most months. What made your comment about how they rarely update it [stand out], especially this month, is that they issued 2 updates shortly after the initial monthly update which is unusual. You picked the worst month in the past couple of years to say they rarely update the software since this is the first time in 2+ years

they have done it more than twice in one month. I have attached the update log I have on one of my backup servers. Luckily it updates automatically and all of my Linux devices send me an email with my update log when they update.

This month's updates included updates to the relay and discovery servers, which doesn't happen often. I had to update them 3 times this month instead of the normal zero times.

2025.01.12: syncthing v1.29.2
2025.01.10: syncthing v1.29.1
2025.01.06: syncthing v1.29.0
2024.12.03: syncthing v1.28.1
2024.10.16: syncthing v1.28.0
2024.09.06: syncthing v1.27.12
2024.09.03: syncthing v1.27.11
2024.08.06: syncthing v1.27.10
2024.07.02: syncthing v1.27.9
2024.06.04: syncthing v1.27.8
2024.05.08: syncthing v1.27.7
2024.04.09: syncthing v1.27.6
2024.04.02: syncthing v1.27.5
2024.03.05: syncthing v1.27.4
2024.02.06: syncthing v1.27.3
2024.01.02: syncthing v1.27.2
2023.12.11: syncthing v1.27.1
2023.12.05: syncthing v1.27.0
2023.11.15: syncthing v1.26.1
2023.11.06: syncthing v1.26.0
2023.10.03: syncthing v1.25.0
2023.09.05: syncthing v1.24.0
2023.08.09: syncthing v1.23.7
2023.07.04: syncthing v1.23.6
2023.06.07: syncthing v1.23.5
2023.04.05: syncthing v1.23.4
2023.04.04: syncthing v1.23.3
2023.03.07: syncthing v1.23.2
2023.02.07: syncthing v1.23.1
2023.01.02: syncthing v1.23.0
2022.12.06: syncthing v1.22.2
2022.11.02: syncthing v1.22.1
2022.10.31: syncthing v1.22.0

So, I certainly stand corrected. I'm obviously not seeing those update notices for whatever reason. And perhaps I did happen to see one specifically because there were so many of them last month.

In any event, I'm happy to have that corrected. And it's interesting to hear about Brendan's success running his own relay and discovery servers. I've considered doing that, but my particular application allows me to create direct point-to-point links between remote SyncThing instances. I took the trouble to do that, which I've been very happy with, after seeing that the use of communal relaying was dramatically slowing down the resynching process.

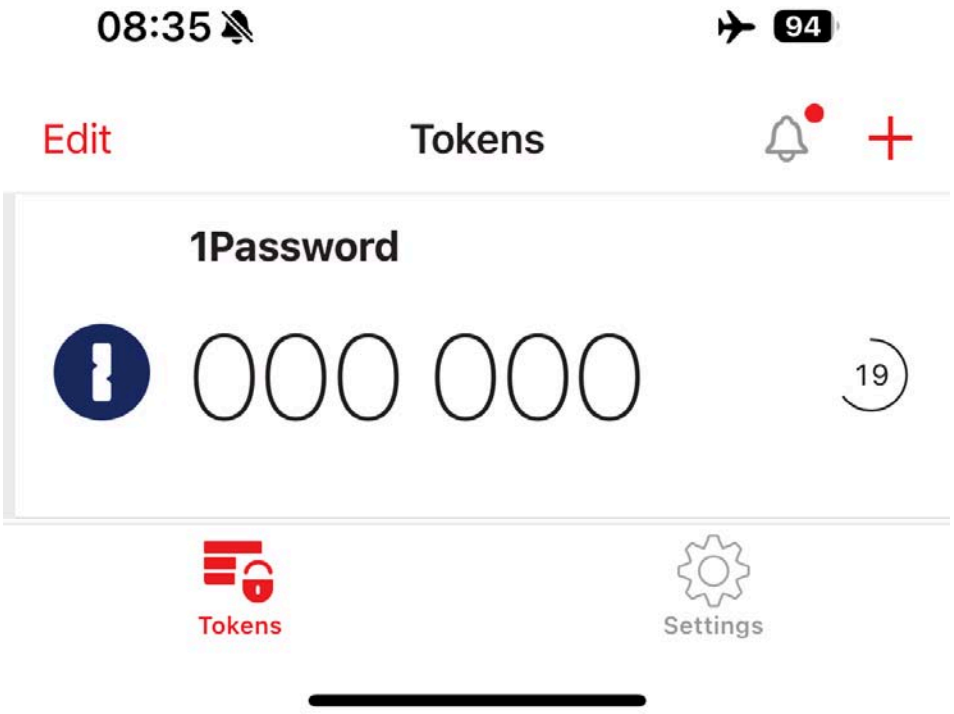
One thing everyone appears to agree about is that SyncThing rocks!

Attacking TOTP

This week we have another example of an instance where a piece of listener feedback I started replying to kept expanding until it had acquired a life of its own and I realized that our listeners would probably enjoy another journey and thought experiment in a direction this podcast has never taken us. Following from last week's podcast topic of HOTP and TOTP, this week we're going to take a detailed look at the task of attacking and cracking a key for the authenticators we all use. We're going to answer the question of whether the 80-bit keys that most sites give authenticators to use are long enough to contain sufficient entropy? And if by any chance you tend to skip podcasts from time to time so that you missed last week's main HOTP and TOTP topic, I would strongly suggest that you pause here to first listen to that one since I need to assume that everyone here now is aware of what happened last week.

This all started with an interesting piece of feedback from our listener, **Lachlan Hunt** (lock-lan):

Hi Steve, I enjoyed your review of HOTP and TOTP algorithms in episode 1008, and wanted to share some of my own observations. I agree that the algorithms are designed to be very easy. I had previously implemented it as a hobby project, and the whole HOTP algorithm can be done in around 10 lines of code. It's a fun coding challenge, and I used it to brute force the next year's worth of codes and see when interesting numbers will appear. See the screenshot showing my 1Password 2FA token equalling 000000:



The widespread use of QR codes for setting up TOTP is not actually defined by either RFC, and instead seems to have originated with Google Authenticator and copied by all other implementers.
https://github.com/google/google-authenticator/wiki/Key-Uri-Format

The QR codes encode the secrets as base 32 strings, where each character represents 5 bits. I had a look at the secrets for some of my own accounts to see how long the secrets were. Many sites had secrets with 16 characters, which is only 80 bits. On the other hand, the longest

secret I saw was a full 256 bits, which seems a bit extreme.

However, The HOTP RFC actually requires that the secret key be a minimum of 128 bits, with a recommendation to use 160 bits. The ones below 128 bits are technically not compliant.

Finally, I thought it was a nice coincidence that there are a million possible 6 digit codes, and there are a little bit over a million 30-second intervals in a year.

The HOTP recommendation of a 160-bit secret key input to the SHA-1 HMAC makes sense since, as we saw last week, SHA-1 produces a 160-bit hash, so that's the output size of HOTP's HMAC. So there's some symmetry there.

But Lachlan observed that many sites were using secrets having 16 characters, which expanded to "only" 80 bits. How should we feel about that? Using a key having only 80 bits for this application provides (and I'll read the number) 1,208,925,819,614,629,174,706,176 unique keys. That's 1.2 million million million million possible keys. Which brings us to the question of whether that is a sufficient number. To address that question we need to remember that when judging relative security, everything is about the application in which the various security components will be used.

So what's the security model of an HOTP-based TOTP authenticator?

The purpose of time-based authentication is the generation of a completely unpredictable code generated within any 30-second window. Using an authenticator whose specific key is hidden among more than 1.2 million million million million possible wrong keys would appear to meet that requirement. But one of the key concepts in security is that of a security margin. So how much security margin do 80-bit time-based authentication keys provide? To answer that question we need to examine the system and design an optimal attack to determine a key.

Given the proven high quality of SHA-1 for pseudo-random bit generation, which is then wrapped by the HMAC algorithm, the only known attack on authentication would be brute force guessing of different input keys which would then be used to generate a specific 6-digit authentication code output at a specific time.

So let's say that we knew our targeted authenticator's output at any given time. So we know the time and the 6-digit code produced at that time. Given the solid design of the authentication algorithm, which is essentially an extremely well-designed cryptographically strong hash function with some ad hoc post-hash processing, the only strategy available to us is simple brute force guessing.

So we start testing all 1.2 million million million million possible keys one at a time, starting at 0.

Each key we feed into the algorithm is combined with the timestamp for the one authenticator output we know. That's processed by the HOTP's HMAC-SHA-1 algorithm, each use of which requires two uses of SHA-1 with some XOR'ing and bit manipulation. Then, as we saw last week, we perform the extraction of the four bytes from the 20, followed by the modulus one million division to extract the remainder and to arrive at our first candidate 6-digit code. Being a high-quality pseudo-random 6-digit code, this first candidate will have one chance in a million of matching the 6-digit code we're seeking.

The probability of things happening is something that often trips people up. If the probability of something random happening is one in a million, we might tend to assume that giving that one-

in-a-million thing one million opportunities to occur – or in our case one million key guesses – we would probably get a collision of 6-digit values. And that's true, but it's not guaranteed. Probability theory tells us that even given one million guesses of a one-in-a-million event, there's a 36.79% chance of **never** hitting upon the value we're seeking. That does mean that given one million guesses there's a 63.21% chance – so better than 50/50 – of hitting upon the number we're looking for. But it's not certain that we would. For random events it's all about probabilities and 693,147 guesses – so nearly 700,000 – would be required to hit the 50/50 point – for an even chance of any one-in-a-million guess being correct.

So at this point, all we can do is keep guessing key values. I should make clear that assuming that the key was generated in a purely pseudo-random fashion, there's absolutely no benefit to generating trial key values at random. No key-generating algorithm could be any better than any other, and being fancy about it would just take more time and waste resources. So to generate successive guesses we treat the key like a large 80-bit binary number that we simply increment. Starting at zero, we'll eventually test them all. The problem, of course, is that "80" is a lot of bits. We've already seen that there are 1.2 million million million million possible combinations of those 80 bits.

So let's proceed to see what happens. We keep incrementing our key and keep producing 6-digit codes until we hit upon the one that the target authenticator produced for the same timestamp.

So, yay! We found an 80-bit authenticator key that gives the proper 6-digit output at the proper time. But that's no use to an attacker since it's never going to be that time again. And, besides, they already know the proper 6-digit code for **that** time. The goal is to be able to generate the proper code for **any** time in the future. So for that they need THE one key that will do that.

The problem is that there are 1.2 million million million million possible 80-bit keys and the only thing we've just accomplished is to find the first key counting upward that produces this one correct 6-digit code. Since we know that these codes are randomly distributed throughout the entire key space, that means that there will be, on average, 1.2 million million million total keys that will also produce this same 6-digit code for this same timestamp. In other words, the discovery of that first matching code is very unlikely to be useful. We still need to eliminate many millions of millions of other keys.

To do that, we need some more sample outputs from the target authenticator. So we've just clearly proven one thing: There is absolutely no possible way for an attacker who obtains a user's single 6-digit code at one point in time, to reverse engineer a user's authentication key regardless of how much time and processing power they may have. And note that this is all symmetric crypto which has always been safe from any threat from quantum computing. So holding out for a quantum computer to arrive also won't help.

As I said, to usefully narrow things down, we need some more sample outputs from the target authenticator. So, let's make that a given. Let's agree that our attacker is able to observe the target authenticator being used with the same key at multiple points in time. So how many points in time do we need and what will that allow us to achieve?

As we've seen, each point in time gives us one code in a million. And in its first use, out of the total 1.2 million million million million possible keys, this one in a million matching would allow us to select one candidate key out of every million possible keys. So it effectively reduced the candidate key space by a factor of one million. In other words, we're able to use a 6-digit code generated by the targeted authenticator to weed out a factor of a million possible keys. Or phrased differently: Each application of a different 6-digit code can be used to reduce the remaining candidate key space by a factor of one million.

So, suddenly that doesn't seem so bad.

An 80-bit key space gives us a total of 1.2 million million million million keys. That's four millions. And we've seen that each use of one 6-digit code for a given point in time will, on average, eliminate a factor of one million wrong keys that do **not** produce a matching 6-digit output. So that would suggest that the use of four 6-digit code output samples, each reducing the total key space by a factor of one million, would bring the key space down to one or two remaining candidate keys.

Okay. So let's go back now to that first test where we were incrementing the 80-bit key and generating a test 6-digit code to look for a match against the authenticator's known output. We know that we will eventually find a match and that the probability of that happening is 50% during the first 693,147 tries, rising to 63.21% by the time we've tried the first million keys. Regardless, we know it's going to happen sooner or later.

So having found the first candidate key that gave us the first proper 6-digit output, we know that this only reduced the possible key space by a factor of one million. So next we try this same candidate key against the **second** point in time to see whether we obtain the proper second 6-digit code. This will still be highly unlikely since that first test left 1.2 million million million candidate keys, only one of which is the one we're seeking.

But nevertheless, we check the key against the second point in time and almost certainly fail. That means that the first test found a key that produced the proper 6-digit result at this point in time but not at the second reference point. So we need to keep searching. We move forward until we again find a match for the first point in time, then again check that against the second point in time. As before, there are still so many candidate keys that will pass the first test but fail the second that it's likely to take quite a bit more searching until we find a candidate key that passes both the first and second tests.

But we're still a long way from home. Since each of these first two tests reduces the candidate key space by a factor of one million, together they reduce it by a million million. But since we started with an 80-bit key that gave us a key space of 1.2 million million million million, that means that even after finally finding a candidate key that passes the first two tests, that new key that was found is still only one among the 1.2 million million that will pass that test so it's still exceedingly unlikely that the one we found is the proper key.

To test this, we, of course, check this latest candidate against our 3rd authenticator sample. And as we know, there's only one chance in around 1.2 million million that this first key that passed the first two tests will also pass the third. And even if it did by some miracle pass the third test, it would still be one of 1.2 million keys that would do so. So we would then need to test against a fourth authentication sample output to see whether that key, which somehow managed to pass the first, second and third tests, was the one out of 1.2 million that can also pass the fourth sample test. And since there were "**1.2** x one-million^4" possible keys, even **this** might not be the one we're looking for. And we need to remember that **when** we succeed in this search boils down to statistics.

That 69.3% number we encountered earlier comes back here, since we're essentially performing four, unrelated one-in-a-million tests against random events where we need all four of them to succeed. So we would need to test on the order of 6.93×10^23, 80-bit keys before we would reach the point of having a **50% chance** of finding a first key that passes all four of our one-in-a-million 6-digit matching tests. 6.93×10^23 is 57.3% of the total 80-bit key space to search, only to achieve a 50% chance of success.

One question to ask is whether there might be any shorter route for brute forcing a solution. After giving this some thought I cannot see one. The shortest solution will be the algorithm we just examined: to check successive keys against a first test and to then apply successive tests until we either rule out a key with a failure or, by some miracle, find a key that works against every test we apply.

And we also know that we will need to test 57.3% of the total 80-bit candidate key space, which is 6.93 x 10^23 keys, in order to have just a 50% chance of success with no guarantee even then. And each test with a candidate key will require two uses of SHA-1 for the HMAC algorithm and the application of the ad hoc HTOP 6-digit extraction.

It's easy to say 6.93 x 10^23, just as it's easy to be glib about 80 bits. But 6.93 x 10^23 is 693,000,000 million billion. So, if an attacker were able to perform a million billion of these complete TOTP/HOTP candidate key tests per second, we would still be left with 693,000,000 seconds, which is 22 years full time around the clock without pausing and to even then obtain a 50% chance of cracking a single key of a time-based one-time password when having a handful of that authenticator's outputs and knowing exactly when they were generated.

Modern hardware has become very fast. But it's generally fast at performing simpler algorithms for which it's been designed; like straight SHA-256 hashing for cryptocurrencies. Ad hoc algorithms, especially something as whacky as HOTP which selects the bits to be divided based upon some other bits in a nibble, would be much more difficult to accelerate. So it might be that even a million billion complete tests per second would be difficult to achieve in practice.

But that said, given the current performance of crypto mining, and a million billion tests per second taking "only" 22 years for a 50% chance of success, that's not the sort of security margin that would or should make anyone feel completely comfortable. It's better when realistic estimates come in at 22 million years than just 22 years. This really boils down to how fast the individual tests can be performed.

I'm not at all worried about sites being protected by 80-bit keys. But given what we've just learned from this exploration, I would feel more comfortable if the keying material had 128 bits. That's a difference of 48 bits, and that makes a HUGE difference. Adding 48 bits scales the entire problem up by a factor of nearly 281,475 million times. So NOW we're talking many many millions of years and we have the sort of security margin that means we never need to think about the problem.

Given that the key length being offered is entirely transparent to any authenticator user, there's just no reason not to use 128 or more bits for the key. 80 is okay, but more would be better and 80 should definitely be considered a minimum.