

# Security Now! #1078 - 05-12-26

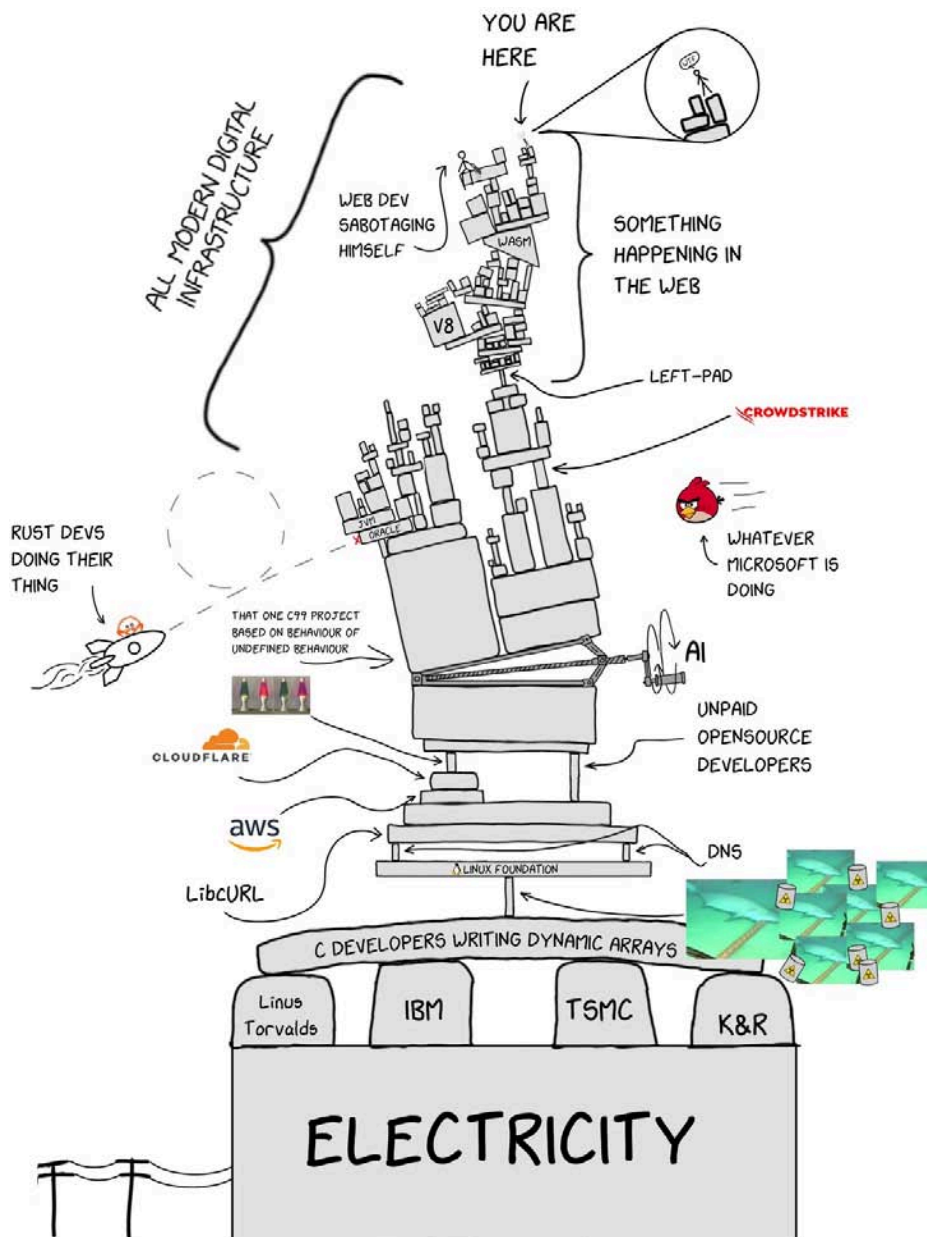
## DigiCert does it right

### This week on Security Now!

- The FCC decides router firmware updates are useful.
- Netgear applies for and gets a full FCC pass.
- AI uncovers a 21-year old critical FreeBSD RCE.
- What was behind that Let's Encrypt outage.
- AI model repositories are overflowing with malware.
- The CISA 2015 info-sharing act is being renewed.
- Edge leaves ALL usernames and passwords in the clear.
- An examination of DigiCert's breach and their response.

### When a powerful meme creates fertile ground

(with apologies to Randall Munroe of XKCD)



## Security News

### FCC to allow software updates for routers until 2029

There's news on the residential router front. Apparently someone at the FCC got a clue, or at least they listened to someone who actually knew something about cyber security. Because last Friday they announced a reversal of their previous "no updates for you!" policy. TomsHardware wrote:

*The Federal Communications Commission announced on Friday, May 8, through its Office of Engineering and Technology (OET), that it was extending temporary waivers allowing certain foreign-produced drones, drone components, and consumer routers to continue receiving software and firmware updates in the United States.*

*In late 2025 and early 2026, the FCC added these categories of equipment to its "Covered List," which effectively blocked already-authorized devices from receiving post-approval software and firmware modifications. The agency subsequently issued waivers permitting critical security and functionality updates to continue through March 1, 2027, for consumer routers.*

*Under the [now] updated waiver, manufacturers of affected devices will now be allowed to continue issuing software and firmware updates until at least January 1, 2029, provided the devices had already been authorized for use in the U.S. before being added to the FCC's "Covered List." The extension also broadens the waiver to include certain Class II permissive changes involving software and firmware updates intended to mitigate consumer harm.*

*In its notice, the FCC acknowledged that continued software support remains necessary to protect U.S. consumers. The waiver specifically allows updates that maintain device functionality, patch vulnerabilities, and preserve compatibility with changing operating systems and network environments. The agency argued that the public interest would be better served by allowing these limited updates rather than freezing software support entirely.*

In other words: "**Duh!!**" Tom's adds:

*The waiver does **not** reverse the broader restrictions or remove the devices from the Covered List. It applies only to already-authorized products and to software- and firmware-related changes intended to maintain safe and secure operation. Manufacturers must still comply with other FCC requirements governing permissive changes and equipment certification.*

So that allows for nearly an additional two years of updates to existing routers, which is certainly good news. Of course, the entire thing remains unspeakably ridiculous because control over a router's firmware is all anyone needs to turn that router into an Internet bandwidth weapon. The hardware need not change. So either you trust the foreign manufacturer of a router or you don't. If you do, then there's no problem. If you don't, then limiting updates to the original March 1st, 2027 deadline is of absolutely zero benefit, since you've given them, under the assumption that they have malicious intent, one full year to cook up new sneaky malware with which to infect any routers that may be updated. None of this has ever made any kind of sense.

As we saw last week, CISA has been effectively neutered. And our agencies appear to now be staffed and run by people who will not push back against policies that they know are clearly

wrong. So this sort of nonsense is what results. It is what it is.

### **Netgear gets a pass**

One piece of good news, for those who like and use Netgear routers, is that even before the addition of those additional two years of firmware updates, Netgear had announced that it had received the FCC's conditional approval for its routers. This meant that **none** of those ridiculous FCC-imposed restrictions would affect **any** of Netgear's router products – not those already sold and not any current or future models. Netgear gets a full pass which includes their right to update their firmware with abandon.

### **A 21-year old critical vulnerability in FreeBSD (found by AI)**

We've heard again from the guys at AISLE Security. They're the commercial group who have been using their own AI to find flaws in software and who were somewhat annoyed by all the hoopla that Anthropic generated with Mythos. The headline of last Thursday's posting was "*AISLE Discovers CVE-2026-42511: a 21-Year-Old FreeBSD Remote Command Execution Vulnerability*"

*FreeBSD is often described as one of the most secure operating systems in the world, with its reputation arising from its high-quality networking stack, deliberate engineering, and a philosophy of security through simplicity. FreeBSD's history and usage are remarkable: it powers Netflix's Open Connect infrastructure, Sony's Playstation OS, part of Nintendo's Switch OS, Yahoo's backend services, NetApp's storage systems, Citrix's Netscaler, has long helped form the software base of major networking platforms (Cisco, Juniper, and so on), WhatsApp's backend services (historically), and is now the focus of a substantial Foundation effort to make it work better on modern laptops, and, for full disclosure, remains the author's personal operating system of choice.*

And I'll just interject to note that it's also my own UNIX OS of choice. It underlies the pfSense personal firewall router system and it's the UNIX that GRC runs for our DNS and Newsgroups.

*AISLE discovered a remote command execution vulnerability in FreeBSD's dhclient, that is trivially weaponizable and wormable by any system on the same local network as the FreeBSD system. The vulnerability first entered FreeBSD in the 2005 release of FreeBSD-6.0 when OpenBSD's dhclient was imported, and lay dormant until discovered by AISLE. The vulnerability also affected OpenBSD until 2012, when that operating system deprecated dhclient-script completely, indirectly fixing the vulnerability.*

*The initial flaw was identified by AISLE's AI-based source code analysis pipeline and then investigated by our triage agents. Joshua Rogers of AISLE's Offensive Security Research Team traced the relevant code paths, established the full security impact, and developed a proof of concept demonstrating a complete local-network-to-root exploit chain.*

*FreeBSD is adding key improvements to laptop support including greater Wi-Fi support, so the attack surface here becomes even more relevant to everyday systems. A malicious wireless access point, or in some cases another attacker on the same Wi-Fi network able to spoof DHCP, can target the exact DHCP path that almost every wireless FreeBSD system will rely on.*

*Imagine you're the author of this post, who runs FreeBSD on their laptop: you're at a coffee shop, airport, or hotel, and as soon as you connect your FreeBSD-equipped laptop to the Wi-Fi, your whole system is hijacked in secret. Imagine you have a PlayStation whose OS is locked down from any unofficial access, only to be hijacked by connecting to a network. In other words, this vulnerability not only affects servers, but any FreeBSD machine that connects to a network using DHCP.*

*The vulnerability was a logic flaw that allowed attacker-controlled protocol data to be persisted into a trusted configuration-like format without proper sanitization, then later reinterpreted in a privileged execution path. That is exactly the kind of bug AISLE's autonomous security platform is built to find. Like our recent findings in OpenSSL, Firefox, libpng, and Amazon's Crypto Stack, this result came from disciplined engineering and end-to-end analysis, **not model mythology.***

Hmmm. "Not model mythology." Seems as though they may still be somewhat annoyed by the "mythology of Mythos." But in any event, AI truly is finding serious flaws, many which have been present for decades. In this case we're talking about FreeBSD's DHCP client which can be fed a maliciously formed reply containing code that it will execute. As Joshua Rogers, who authored this write-up noted, this could have been extremely serious if it had not been found by the good guys.

At some point we may see those who claim that AI-enhanced software vulnerability discovery never turned out to be such a big deal. Objective observers would do well to remember all of the many critical vulnerability discoveries, like this one, that served to clean up our archaeological code base before the bad guys had the chance to exploit it.

## **Let's Encrypt Outage**

There was a bunch of Internet chatter last Friday about a several-hours outage of Let's Encrypt. Our listeners know that with so much of the web now utterly dependent upon certificates issued by Let's Encrypt, and with maximum certificate lifetimes continuing to drop, and especially with Let's Encrypt's optional super-short life 6-day certs now being available, any outage of the system upon which so much now depends is of interest.

In this case, it was a deliberate and temporary administrative suspension of new certificate issuance following reports of a missing extension from a class of certificates. Let's Encrypt's post-incident report said:

*Let's Encrypt's Gen Y (YE and YR) Cross-Certified Subordinate CAs were issued in violation of CCADB policy, which requires that the serverAuth EKU extension MUST be present in cross-signed intermediate certificates issued since June 15th 2025. Root YE and YR were issued September 3rd 2025 and are subject to the requirements.*

The certificate extension in question, a "serverAuth EKU", where EKU is the abbreviation for Extended Key Usage, specifies the limits on the application of any certificates which that CA validates. These are things like may be used for "Server Authentication", "Client Authentication",

“Code Signing”, “Email Protection” or “Time Stamping”. That extension was missing. It should have been there. It mostly doesn’t matter that it wasn’t. But, as we know, any CA must take their responsibilities absolutely seriously – and Let’s Encrypt did. They immediately stopped issuing. They fixed the problem. They resumed issuing. In their words: *“We temporarily disabled certificate issuance, deployed a configuration change to prevent future issuance from the cross-signed Gen Y hierarchy, and then re-enabled issuance.”*

### **Deliberately malicious open source AI models**

Last Friday, TheNextWeb posted the news of an analysis of the Large Language Models being hosted and offered at HuggingFace and ClawHub. The news is not good. Here’s what they said:

*The two most important software supply chains in artificial intelligence have been systematically compromised. Hugging Face, the repository that hosts more than a million machine learning models used by virtually every AI company on the planet, has been found to contain hundreds of malicious models capable of executing arbitrary code on the machines of anyone who downloads them. ClawHub, the public registry for OpenClaw’s AI agent skills, has been infiltrated by a coordinated campaign that planted 341 malicious skills designed to steal credentials, open reverse shells, and hijack AI agents for cryptocurrency mining.*

*The attacks are different in technique but identical in logic. Both exploit the implicit trust that developers place in shared repositories. Both use the infrastructure that the AI industry built to accelerate development as the vector for compromising it.*

*Hugging Face has been aware of malicious models on its platform since at least 2024, when security firms JFrog and ReversingLabs independently identified models containing hidden backdoors. The problem has not been contained. It has scaled.*

*Protect AI, which partnered with Hugging Face to scan the platform’s model library, has examined more than four million models and identified approximately **352,000 unsafe or suspicious issues across 51,700 models**. JFrog found more than 100 models capable of arbitrary code execution. The attack technique, known as “nullifAI,” exploits Python’s pickle serialisation format, the standard method for packaging machine learning models. Attackers embed malicious Python code at the start of the pickle byte stream and compress the file using 7z rather than the default ZIP format, which breaks Hugging Face’s PickleScan detection tool.*

*The payloads are not subtle. Security researchers have documented models that establish reverse shells connecting to hardcoded IP addresses, giving attackers direct access to the machine of anyone who loads the model. Others execute credential theft, exfiltrate environment variables, or download secondary malware. A data scientist who downloads what appears to be a legitimate model for a research project or production pipeline is, in some cases, handing control of their machine to an attacker.*

*Hugging Face has responded by partnering with JFrog and Wiz to improve scanning capabilities. JFrog’s integration has eliminated 96 per cent of false positives in malicious model detection. But the platform’s open architecture, which is the source of its value to the AI community, is also the source of its vulnerability. Anyone can upload a model. The scanning catches known patterns. The attackers who designed nullifAI built their technique specifically to evade the scanning.*

*ClawHub, the registry for OpenClaw's AI agent ecosystem, faces a different but related problem. OpenClaw has grown to 3.2 million users and attracted partnerships with OpenAI, but its skill registry has become a target for attackers who understand that an AI agent executing a malicious skill has access to whatever the agent has access to, which in enterprise environments can mean databases, APIs, internal networks, and cloud credentials.*

*Koi Security audited all 2,857 skills on ClawHub and found 341 malicious entries. Of those, 335 were traced to a single coordinated operation called "ClawHavoc." Separately, Snyk's ToxicSkills research examined the broader ecosystem and found that 36 per cent of all AI agent skills contain security flaws, with approximately 900 skills, roughly 20 per cent of the total, classified as malicious. Thirty skills from a single author were silently co-opting AI agents for cryptocurrency mining.*

*The ClawHub attacks are particularly dangerous because of the nature of AI agent architectures. The rise of model context protocol and similar standards in the agentic era has created a new category of software supply chain in which AI systems autonomously select and execute tools from external registries. A compromised skill does not require a human to click a link or open a file. It requires an AI agent to select the skill as part of a workflow, at which point the malicious code executes with the agent's permissions.*

*The Hugging Face and ClawHub compromises are the AI-specific manifestation of a supply chain attack pattern that has been accelerating across the software industry. In March 2026, the LiteLLM package on PyPI was compromised, potentially exposing 500,000 credentials including API keys for Meta, OpenAI, and Anthropic. Meta froze its AI data work after the breach put training secrets at risk. In April, a Bitwarden CLI package on npm was hijacked for 90 minutes with a payload specifically designed to harvest credentials from AI coding tools including Claude Code, Cursor, Codex CLI, and Aider. Days later, the PyTorch Lightning package was compromised for 42 minutes with a credential-stealing payload from the "Mini Shai-Hulud" campaign.*

*The European Commission itself was breached after attackers poisoned Trivy, an open-source security scanning tool, demonstrating that even the tools designed to detect supply chain attacks can become vectors for them. The United States Department of Defence published formal guidance on AI and ML supply chain risks in March 2026, acknowledging at an institutional level that the AI software ecosystem has become a national security concern.*

*The common thread is speed. The PyTorch Lightning compromise lasted 42 minutes. The Bitwarden CLI hijack lasted 90 minutes. The LiteLLM attack window is estimated at hours. These are not persistent campaigns that defenders have weeks to detect. They are brief, targeted insertions that exploit the automated dependency resolution systems that modern software development relies on. A developer who runs a package install at the wrong moment downloads the compromised version. The window closes. The damage is done.*

*The AI industry has invested hundreds of billions of dollars in model training, inference infrastructure, and application development. The investment in securing the repositories through which that software is distributed has been a fraction of the total. Hugging Face has partnered with security firms. ClawHub has implemented basic moderation. Package registries have added two-factor authentication requirements. None of these measures has prevented the attacks documented above.*

*State actors can already produce AI-powered malware that evades conventional detection, and the supply chain attacks on AI repositories represent a natural evolution of that capability. The models and skills hosted on Hugging Face and ClawHub are consumed by systems that make automated decisions, process sensitive data, and operate with elevated permissions. A compromised model in a production AI pipeline is not equivalent to a virus on a personal computer. It is a backdoor into an automated decision-making system that the organisation trusts precisely because it appears to be a legitimate component of its AI stack.*

*The fundamental problem is architectural. The AI industry built its development infrastructure on the same open-registry model that has defined software development for the past two decades: centralised repositories where anyone can publish, automated tools that download and execute code from those repositories, and a culture of trust that treats popular packages and models as implicitly safe. The difference is that AI models are not just code. They are serialised objects that execute during deserialisation, a property that makes pickle-based models inherently more dangerous than traditional software packages, because the malicious code runs the moment the model is loaded, before any human has a chance to inspect it.*

*The AI supply chain is now the most attractive target in software security. The repositories are trusted. The consumers are automated. The payloads execute on load. And the industry that built these systems is spending its security budget on model alignment and prompt injection while the infrastructure through which the models are distributed remains, in the assessment of every major security firm that has examined it, comprehensively compromised.*

So Leo, I would say that the caution and trepidation you felt when you were first considering turning OpenClaw loose in your world was likely warranted. I wanted to share this with our listeners because I am certain that we have many who are enjoying playing with and experimenting with and perhaps even deploying systems or solutions using these openly available AI models.

Please please please be careful.

### **Reauthorization of the Cybersecurity Information Sharing Act of 2015**

There's some good news on the horizon. The word is that the original decade-long CISA 2015 Cybersecurity Information Sharing Act which expired last year before being temporarily extended until this coming September is in the process of receiving a much-needed long-term reauthorization. Recall that this is what allows private sector enterprises to share cyber intelligence with the government without fear of legal blowback.

### **Edge statically decrypts all stored passwords in RAM**

A number of our listeners pointed me at the news that Microsoft's Edge browser is doing little to protect its users' passwords. A posting from the SANS Institute (which I've enhanced a bit) says:

*Yup, it's for real. This started with a post in X which highlighted research by @L1v1ng0ffTh3L4N that found exactly this issue. Edge stores all of your browser passwords in clear text, even if you haven't used them in this session, y'know, just in case.*

*I figured, it couldn't be that easy, right? But like so many things, yes, yes it was.  
To reproduce this:*

- *Open Edge. Don't browse anywhere, just open it*
- *Flip out to Task Manager, search for Edge, then expand that task*
- *Highlight the "browser" sub-task, right click, and choose "Create Memory Dump"*

*Navigate to where the DMP file is stored. If you haven't used strings before, you're in for a treat. Strings is of course just part of most Linux distros, but you can easily get a copy for Windows as part of MS Sysinternals. Now let's look for passwords! You could use strings and look for known credentials, just search for a known password and you will certainly find it. Or you can take advantage of the format of the saved data:*

*<url of the site><protocol>< ><userid>< >password>*

*So, searching for "<tld><protocol>", which in most cases is "comhttps" (no spaces) will find them, and they'll all be in one nicely formatted group no less. The command for that will be:*

*strings -n 8 msedge.DMP | find "comhttps"*

*It really is that easy.*

*And the ironic thing? To view these same credentials in the browser, there's a whole security theatre process where Edge wants your biometrics as proof before disclosing even the userid and site names - you know, "for security". All the while, the whole shot is in clear text, free for the looking. Also as noted in the X post, Microsoft classifies this as "intended behaviour." I'm not sure what manager or lawyer decided that, hopefully it wasn't anyone in their security team.*

*Any logged in Windows Edge user can dump all of their stored Edge credentials with no additional rights. Which means that any malware that user executes also has access to all of those credentials for the asking. But not to worry, right? It's intended behaviour.*

*If what's intended is also to get me to use Firefox or Chrome, it's working!!*

# Listener Feedback

Todd Whittaker

*Steve, I thought this might be of interest for Security Now: The Rival Security group has a thoughtful follow-up on the Claude Mythos FreeBSD exploit story, arguing that Mythos may not have been quite as "creative" as the initial coverage suggested.*

*<https://rival.security/posts/mythos-discovered-a-cve-already-in-its-training-data---and-thats-s-till-worrying>*

*Their claim, as I understand it, is not that the result is unimportant. It is that the vulnerability, the prior fix pattern, and perhaps even the exploit-relevant structure may already have existed in the model's training data. The FreeBSD issue appears closely related to CVE-2007-3999 in MIT Kerberos: same general RPCSEC\_GSS validation logic, same stack-buffer overflow pattern, and a strikingly similar bounds-check fix. So Mythos may have "discovered" something genuinely dangerous, but perhaps by recombining known historical material rather than reasoning from first principles in the way many of us initially imagined.*

*That still seems worrying, just in a different way. If advanced models can rediscover old vulnerability patterns embedded in the fossil record of open-source code, then attackers may not need models to be brilliant. They only need them to be tireless, well-tooled, and good at recognizing dangerous old ideas in new places.*

I'll interrupt Todd to note that I agree completely with that. As our understanding of computer science has evolved, we've come to notice patterns in the solutions to problems. They're like a short and small abstraction away from the concrete solution where we see that many different concrete solutions can be grouped together by their sharing of a common underlying pattern.

So, what Rival Security observed was Mythos Preview finding what we might describe as a flaw design pattern — a common type of mistake that coders have been making through the years, which winds up being a "natural" sort of mistake to make due to the underlying architecture of the computer behavior we're programming. We all know that today's large language models excel at pattern discovery. Probably more than anything, that's what they are. So we would expect that if someone, somewhere, made a similar mistake and correction that had been captured in the model's training corpus, it would, indeed, be able to make the connection.

So I'd say that this is an interesting and useful observation about the underlying way Mythos discovered the newer similar-patterned flaw, but I don't see that taking anything away from the fact that, as Yoda might say: "Discover that flaw it did." Todd's note continues with some interesting observations from his own background as a computer science educator. He writes:

*I would also be interested in your broader take on what this means for computer science education and the profession. My current working view is that the most productive human-AI collaboration in software engineering depends on advanced judgment: architecture, design patterns, threat modeling, failure modes, invariants, tradeoff analysis, and knowing when the AI's answer is superficially plausible but structurally wrong.*

*The problem is not that students must learn "the old ways" before they are permitted to use the new tools (that's just reframing the old learn-assembly-before-C argument). The real issue is that AI makes code cheaper while making judgment more valuable.*

*To supervise AI-generated software, a person needs mental models: how state behaves, where abstractions leak, how protocols fail, why concurrency is treacherous, how memory and parsing bugs become security bugs, and why a working MVP may still be architecturally unsound. Those mental models do not emerge from prompting alone. If we let AI collapse the difficult apprenticeship too early, we may produce developers who can ask for software but cannot reliably tell whether the software they received is safe, coherent, or professionally defensible.*

*I'm writing from my personal email, but my day job is in computer science education, so this one lands close to home. I spend a fair amount of time thinking about what we should still teach humans when machines can increasingly produce the code.*

I don't think I've seen any more coherent and clear description of the AI -vs- human coding question. I loved his one line: *"The real issue is that AI makes code cheaper while making judgment more valuable."* I think that captures where we're headed in computer science education and vocation. In the same way that any higher level language lifts its coder away from the grubby details of the specific underlying computer hardware, the use of coding-trained large language models clearly lifts its users from the grubby details of the way computers are applied to problem solving. As many *"never before programmed anything"* users are discovering, they're now able to simply ask for what they want the computer to do and the LLM will almost magically produce a potion that does that.

But, there are clear limits to what can be asked for. We saw a perfect example of such limits last week when those bad guys, who had created that credit card clearing web portal, apparently forgot to ask for authentication to be added. Whoopsie.

What we're seeing rapidly evolve during the rush to use AI for code generation, is that for AI to be applied to the creation of any very large and complex problem, **a solution architect** is still required. No large problem can be dropped, whole, into AI's lap – at least not today, not yet... and I have no idea when. Instead, for now, today, a solution architect who is trained and experienced in the application of the various higher level solution abstractions that have been developed over the years of true computer science, needs to carefully decompose the larger problem into much smaller individual safely codeable pieces. These solution abstractions are the true core of the science of computing. Coding is just their implementation. These are the sorts of things that Donald Knuth and other scholars of the art have spent their lives exploring and documenting.

## **Randy Krum**

*Hi Steve, In episode 1077, you mentioned you think companies that have closed-source software should move quickly to utilize Anthropic's Mythos, Claude Security or similar tools as they emerge. Their closed-source code is only closed-source to the outside world. I think you glossed over the risk that using these online AI tools potentially exposes your closed-source code to the world. They have privacy and security tools in place, but their motivation is financial. They have a setting to disallow using your AI conversations to train their AI models, but you have to trust that they are actually following that setting. For the same reason you trust Apple with your data more than Facebook or Google, the use of online AI tools to review closed-source code is a risk. A security breach (internal or external) could be devastating to a*

*software company that has their code exposed.*

*I've been experimenting with LM Studio, to run local, off-line, open-source LLM models for use with proprietary data. (NOTE: I work with client data, not code) The hypothesis is that a local, off-line LLM can be safely used with confidential, internal, proprietary data or code. These LLM models are usually not as current as the online tools, but they catch up quickly. Also, they're not as flashy, news-worthy or marketing-hyped as strongly as the major online tools.*

*What are your thoughts on the risk of exposing proprietary code (or data) by using the major online tools? Listener since Episode 1. Thanks for everything you and Leo do!*

Randy's right. I did gloss-over those risks, so I'm glad he brought it up. And it's not **at all** that I meant to down-play them. Given everything we know about cloud breaches, network data interception and decryption and so on, even if the external LLM provider did nothing wrong and made no mistakes, shipping highly valuable source outside of a company's perimeter creates risk.

However, that said, how many firms are already doing just that by using Github? I think that's insane, yet it has become common practice to use Github for highly proprietary source code management. I'm not doing that, so perhaps my view is skewed. But it does mean that the company's crown jewels are already exposed.

Randy correctly notes that sending the code up into the cloud for an LLM to rummage around in poses another level of danger, and I agree in principle. So the use of local models, which I have no doubt we're going to be seeing much more of in the future, makes a great deal of sense once they become as capable as what's available in the cloud.

# DigiCert does it right

The first I learned of some sort of trouble was from someone posting to GRC's Security Now! newsgroup with firsthand experience. Peabody wrote:

*This morning Windows Defender told me it had discovered a "severe" rootkit on my Windows 10 laptop called "Win32 Cardigent A!dha", which it has quarantined. Searching online tells me this is happening on both Windows 10 and 11 computers worldwide, and one hash involved is that of a legitimate DigiCert certificate. This is all above my pay grade, but I'm going to leave things alone for a while and see what happens. Apparently lots of people are reinstalling Windows because of this, but I think that's super premature at this point. My guess is that this is a gift from Microsoft, which they will admit to shortly. And if you reformatted your drive, they will apologize for the inconvenience.*

Later that Sunday, BleepingComputer's Lawrence Abrams was all over this and was providing answers. Lawrence's headlined his news: "*Microsoft Defender wrongly flags DigiCert certs as Trojan:Win32/Cerdigent.A!dha*", following which Lawrence writes:

*Microsoft Defender is detecting legitimate DigiCert root certificates as Trojan:Win32/Cerdigent.A!dha, resulting in widespread false-positive alerts, and in some cases, removing certificates from Windows. According to cybersecurity expert Florian Roth, the issue first appeared after Microsoft added the detections to a Defender signature update on April 30th. Today, administrators worldwide began reporting that DigiCert root certificate entries were flagged as malware and, on affected systems, removed from the Windows trust store.*

Now, to be clear what a disaster this was, as we know, root certificates anchor the chain of trust for everything that chains down to them. With them removed from a system, nothing that chains down to them will be trusted despite having been trusted just moments before. That's the way the system works and no one has come up with a better idea for validating signed code. Those two certificates **are** DigiCert's code signing roots and, for example, all of GRC's signed apps are anchored by one of them. So the mistaken removal of those two code signing roots from the Windows trusted root store automatically and instantly renders **every app** that was ever signed by a certificate from DigiCert, the industry's number one certificate authority, invalid and untrusted by Windows. BleepingComputer continues:

*These false positives have led to concern among Windows users, with some thinking their devices were infected and reinstalling the operating system to be safe. Microsoft has reportedly fixed the detections in Security Intelligence update version 1.449.430.0, and the most recent update is now 1.449.431.0. Reports on Reddit indicate that the fix also restores previously removed certificates on affected systems.*

Well, yeah, thank goodness for that. And it's not as if Microsoft had any choice. It would have been a true disaster if there weren't some immediate means for reverting the specious removal of DigiCert's perfectly valid root certificates. As we'll see in a few moments, even though DigiCert did suffer a breach which caused it to mis-issue a handful of code signing certificates, at no point

was the removal of any of their root certificates warranted. I hope Microsoft will put some safeguards in place to prevent such a thing in the future. BleepingComputer continues:

*The new Microsoft Defender updates will automatically install, and Windows users can manually force an update by going into Windows Security > Virus and threat protection > Protection updates and clicking on Check for Updates. After publishing this article, Microsoft confirmed that the false positives were linked to detections for compromised certificates from a recent DigiCert breach.*

*Microsoft told BleepingComputer: "Following reports of compromised certificates, Microsoft Defender immediately added detections for malware in our Defender Antivirus Software to help keep customers protected. Earlier today we determined false positive alerts were mistakenly triggered and updated the alert logic. Microsoft Defender suppressed and cleaned up the alerts for customer environments. Customers should update to Security Intelligence version 1.449.430.0 or later, but do not need to take additional action for these alerts. We have notified affected organizations and recommend administrators look for more details in the service health dashboard (SHD) within the M365 admin center."*

Hmmm. Well, that's an entirely unsatisfying answer from Microsoft, but I suppose given what Microsoft has become, it's the best we can expect. Nothing they wrote is untrue, but neither should it satisfy anyone who would have appreciated hearing them say something like: *"In response to reports of compromised certificates, Microsoft Defender was a bit overzealous and mistakenly removed some related certificates that should have remained. Microsoft Defender was immediately updated to cure that behavior and has replaced any certificates that were mistakenly deleted."* Is that so difficult to say? It should not be. Just wait until you see how thoroughly DigiCert took full responsibility for their part in this drama. Lawrence's reporting continues, writing:

*The false positives occurred shortly after a disclosed DigiCert security incident that enabled threat actors to obtain valid code-signing certificates used to sign malware. The DigiCert incident report explained: "A malware incident targeted a customer support team member. Upon detection, the threat vector was contained. Our subsequent investigation found that the threat actor was able to procure initialization codes for a limited number of code signing certificates, a few of which were then used to sign malware. The identified certificates were revoked within 24 hours of discovery and the revocation date set to their date of issuance. As a precautionary measure, all pending orders within the window of interest were cancelled. Additional details will be provided in our full incident report."*

There's a small sample of what full disclosure looks like. Lawrence continues:

*According to DigiCert's incident report, attackers targeted the company's support staff in early April by creating support messages containing a malicious ZIP file disguised as a screenshot. After multiple blocked attempts, one support analyst's device was eventually compromised, followed by a second system that went undetected for a time due to an endpoint protection "sensor gap." Using access to the breached support environment, the hacker used a feature in DigiCert's internal support portal that allowed support staff to view customer accounts from the customer's perspective.*

*While limited in scope, this access exposed "initialization codes" to previously approved, but undelivered, EV code-signing certificate orders. DigiCert explained: "Possession of an initialization code, combined with an approved order, is sufficient to obtain the resulting certificate. Since the threat actor was able to obtain these two pieces of information for a finite set of approved orders, they were able to obtain EV Code Signing certificates across a set of customer accounts and CAs."*

*DigiCert says it revoked 60 code-signing certificates, including 27 linked to a "Zhong Stealer" malware campaign. DigiCert explained: "11 were identified in certificate problem reports provided to DigiCert by community members linking the certificates to malware, and 16 were identified during our own investigation." This aligns with earlier reports from security researchers who had observed newly issued DigiCert EV certificates used in malware campaigns and reported them to DigiCert.*

*Researchers, including Squiblydoo, MalwareHunterTeam, and g0njxa, reported that certificates issued to well-known companies such as Lenovo, Kingston, Shuttle Inc, and Palit Microsystems were being used to sign malware. "What do Lenovo, Kingston, Shuttle Inc, and Palit Microsystems have in common?," posted Squiblydoo on X. "EV Certificates from **these** companies were issued and used by a Chinese crime group, #GoldenEyeDog (#APT-Q-27)!"*

*The malware in this campaign is named "Zhong Stealer," though analysis indicates it may be more like a remote access trojan (RAT) than an infostealer. The researcher says the malware was distributed through the following attacks:*

- *Phishing emails deliver a fake image or screenshot*
- *A first-stage executable that displays a decoy image*
- *Retrieval of a second-stage payload from cloud storage such as AWS*
- *Use of signed binaries and loaders, including components tied to legitimate vendors*

*After DigiCert disclosed the incident, the researchers said the incident report explains how the certificates used in these malware campaigns were obtained. It should be noted that the certificates flagged by Microsoft Defender are root certificates in the Windows trust store and do not match the revoked DigiCert code-signing certificates used to sign malware.*

Okay. So that's the great reporting posted Sunday before last by BleepingComputer's founder, Lawrence Abrams. Security industry experts have been citing DigiCert's up front incident report as a model of how this should be done. Starting 24 days ago, DigiCert began issuing a series of incident reports, with each succeeding report updating the previous one, with the final report being posted 7 days ago.

DigiCert named this event "**ENDPOINT2**" and their final report begins with the statement: "*This is an updated version of our Full Incident Report, which completes the investigation of **ENDPOINT2**.*" Their own overview description differs somewhat from what 3rd-parties reported and provides some additional detail. They wrote:

*On 2026-04-02, a threat actor contacted DigiCert's support team via a customer chat channel and delivered a ZIP file disguised as a customer screenshot. The file contained a .scr executable with a malicious payload. CrowdStrike and other security measures successfully blocked four delivery attempts. A fifth attempt compromised **ENDPOINT1**, a machine used by*

*a support analyst; this delivery attempt was detected and contained by our Trust Operations team on 2026-04-03. Following an immediate internal investigation based on the telemetry data at hand, it was assessed that the incident had been contained.*

Then we receive an interesting narrative, some of which Lawrence posted, but the deeper details are interesting. They wrote:

*DigiCert received the initial third party report related to this incident on 2026-04-05. Additional third party reports are identified in the timeline.*

Just to recap those dates before we go any further: The penetration occurred on April 2nd. DigiCert's Trust Operations team determined it had been contained the next day on the 3rd. And the first 3rd-party report of malicious code found in-the-wild, signed with a then-valid DigiCert EV code signing certificate, was the next day, April 4th. DigiCert's reporting continues:

*DigiCert regularly receives certificate problem reports from community members and security researchers for Code Signing certificates, and proven key compromise cases are revoked pursuant to the Code Signing Baseline Requirements. Initial problem reports ultimately linked to this incident report fit within the normal pattern of such revocations.*

So then 10 days go by. They write:

*On April 14th, further investigation identified that **ENDPOINT2**, a machine used by another analyst, was also compromised through the same delivery vector on April 4th. CrowdStrike was **not** installed on that endpoint, meaning the compromise was not detected during the earlier April 3rd investigation. The machine was established more than 3 years ago. Because our end-user machine logs are retained for three years, we cannot determine why CrowdStrike was not installed on this particular endpoint.*

Okay. So, it doesn't really matter why. What matters is that because CrowdStrike was not installed on that second infected ENDPOINT2 machine, its infection went undetected for 10 days. But in the interest of a full forensic after-the-fact investigation, they would have liked to know exactly why that machine had apparently never been protected by CrowdStrike. Their records only go back three years and that machine predates that logging cutoff. So today, they have no way of knowing what happened back when that machine was initially brought online. And now, of course, given that they found one machine that was missing its protection for an unknown reason, the question becomes: what other sensitive machines might also be missing their protection? DigiCert's report continues, writing:

*Our Trust Operations investigation found that the threat actor used the compromised analyst endpoint to access DigiCert's internal support portal. The threat actor used a limited function within the customer-support portal which allows authenticated DigiCert support analysts to access customer accounts from the customer's perspective to facilitate support tasks. This access is restricted and does not permit actions such as managing accounts, users, API-keys, or submitting or managing orders. However, the threat actor was able to use this function to*

*access initialization codes for orders that were approved but pending delivery for EV Code Signing certificate orders across a finite set of customer accounts.*

*Possession of an initialization code, combined with an approved order, is sufficient to obtain the resulting certificate. Since the threat actor was able to obtain these two pieces of information for a finite set of approved orders, they were able to obtain EV Code Signing certificates across a set of customer accounts and CAs.*

If you're wondering about DigiCert's phrase "*across a set of customer accounts and CAs*" the notion of differing CAs might seem strange, since we're only talking about DigiCert. But remember that when I was out shopping around for a new code signing certificate provider, and finally settled upon IdenTrust, I discovered that a surprising number of the many apparent alternative certificate authorities all shared utterly identical prices, terms and conditions with each other, and with DigiCert. It quickly became clear that DigiCert had been busily gobbling up much of the competition. So all of these other alternative CAs had become different storefronts for DigiCert. And what they have just written confirms that these various fronts are all sharing DigiCert's common back end. I'm not criticizing DigiCert. It's smart business. But it does mean that they now have much reduced competition, and that's not clearly best for consumers.

Now we get to some statistics and numbers. They write:

*During our investigation between April 14th and 17th, as DigiCert identified certificates potentially affected by the threat actor's actions, we revoked them. DigiCert revoked 60 certificates issued from the following CAs:*

- *DigiCert Trusted G4 Code Signing RSA4096 SHA256 2021 CA1*
- *DigiCert Trusted G4 Code Signing RSA4096 SHA384 2021 CA1*
- *GoGetSSL G4 CS RSA4096 SHA256 2022 CA-1*
- *Verokey High Assurance Secure Code EV*

*27 of the 60 revoked certificates were explicitly linked to the threat actor (11 were identified in certificate problem reports provided to DigiCert by community members linking the certificates to malware, and 16 were identified during our own investigation). Our investigation included review of the threat actor's activity in the support system as well as tracing delivery to IP addresses known to have been used by the threat actor. The IP addresses used by the bad actor to install the certificates included:*

(And then they list seven unredacted IP addresses: 82.23.186.8, 154.12.185.32, 45.144.227.12, 203.160.68.2, 154.12.185.30, 62.197.153.45, and 45.144.227.29.)

*In addition to the 27 [fraudulently issued and revoked certificates] identified above, 33 of the 60 total certificates were revoked during our own investigation as a precautionary measure. For these certificates, we could not explicitly confirm customer control. In addition, pending orders were cancelled, closing access to the threat actor. All identified certificates were revoked within 24 hours of discovery, with the revocation date set to their date of issuance.*

Note that this is DigiCert explicitly asserting that it has carefully followed the well established CA/Browser forum guidelines for proper CA behavior. This is where, as we've seen and reported,

the other disgraced Certificate Authorities fall well short. Those others were “under the rug sweepers” — first, hoping that no one would notice and catch them in their mistakes, and then once they’d been exposed, working overtime to minimize and hide their failures. In No one expects anyone in this arena to be perfect. They only expect them to always act responsibly and properly and truly with their best efforts.

They wrap-up their initial overview, writing:

*The exploited certificates identified by the community member were found to have been used to sign the "Zhong Stealer" malware family. Our Trust Operations team conducted a detailed review of activity from the compromised endpoints and associated user accounts. This review did not identify related activity within other user accounts. The investigations did not find any misuse of validation information, improper validation actions or other steps that could lead to account settings changes, nor non-Code Signing certificate mis-issuance. In our investigation, we did not find evidence that the threat actor misused other internal systems other than the Code Signing initialization codes within specific accounts.*

*As a result of this incident, additional network and application security changes and reviews, described in the action items, have been undertaken or are underway.*

The really interesting stuff comes next, it being: how they take themselves to task over how this happened and in detail the contributing factors that facilitated the attacker’s success. What I’m about to share is the reason I gave today’s podcast the title: “DigiCert, Doing it Right”. This is written so objectively that it feels more like the work of outside auditors than DigiCert’s own staff. It’s just so difficult to fully disconnect one’s own ego from truly indicting statements. But to DigiCert’s extreme credit there was no sign of the “rolling disclosure” we’ve seen so many times elsewhere. Microsoft could certainly learn a thing or three from DigiCert.

So, under the headline “Root Cause Analysis” we get the necessary contextual background for the reason their EV code signing issuance process is set up as it is, then a clear explanation of four factors that contributed to the trouble. They write:

*Technical Background: Understanding this incident requires understanding how EV Code Signing certificates are issued on hardware tokens:*

- *The customer requests a Code Signing certificate from DigiCert.*
- *Following validation, DigiCert securely provides an initialization code to the customer.*
- *The customer installs DigiCert's Hardware Certificate Installer software locally.*
- *The customer inputs the initialization code into the installer, which generates key pairs on the hardware token and submits the public key to the CA.*
- *The CA generates the certificate against the approved order.*
- *The installer retrieves the resulting certificate and installs it on the token.*

*The process is described in a public knowledgebase at <https://knowledge.digicert.com/solution/set-up-your-digicert-provided-etoken>. Possession of the initialization code, combined with an approved order, is functionally sufficient to generate and retrieve the corresponding certificate. The initialization code operates as a bearer credential for the approved order and is single use. This feature made it apparent which initialization codes had been used.*

I've done exactly this in prior years with DigiCert. When you think about it, the process of issuing a certificate that will be contained on a hardware token is somewhat tricky. The need is for the private key-half of the public/private key-pair to never, ever, for any reason, ever (just to be clear) exist outside the hardware. That means that it must be generated inside the dongle and that the dongle will never export its ultra-protected hardware key.

Web server public/private key pairs have no such requirement. So a web server just uses the underlying operating system's cryptographic system to synthesize a key pair, it holds onto the private key while the public key is placed into a CSR – a Certificate Signing Request – which the CA signs and returns. But forcing the private key to never leave the hardware dongle certainly complicates matters.

The point here is that DigiCert issues these "initialization codes" against a customer's account, sends them to the customer, who then uses them with DigiCert's own Hardware Certificate Installer app to validate their right to have a certificate, trigger the key-pair generation, the public-key upload to DigiCert and the installation of the signed certificate back into the hardware. From the user's perspective it all happens as if by magic. But the stateful nature of the operation does create some points of exploitability.

So, next, DigiCert shares four:

#### ***Contributing Factor #1: Inconsistent or incomplete endpoint detection coverage***

- *Security tooling (CrowdStrike) was not uniformly configured by DigiCert across the user population exposed to the attack. The CrowdStrike prevention setting on ENDPOINT1 was below the intended organizational standard at the time of the initial compromise, allowing the malicious payload to execute before blocking engaged. The CrowdStrike sensor on ENDPOINT2 was not installed. As a result, no detection fired on the compromised machine. Logs for end-entity machines are retained for three years. Since this machine was set up more than three years ago, our security team could not determine why this particular machine either did not have an installed CrowdStrike sensor.*
- *The sensor not being installed was identified on 2026-04-14 during the expanded investigation triggered by the third-party report. The original investigation on 2026-04-04 did not include a check of EDR enrollment status for all exposed users.*
- *If the sensor had been installed on ENDPOINT2, the connection on ENDPOINT2 would likely have been detected and contained in the same timeframe as the other targeted machine (ENDPOINT1). This created the window during which the threat actor was able to access the portal function (see Contributing Factor #2) and harvest initialization codes (see Contributing Factor #3).*

Speaking of Contributing Factor #2:

#### ***CF #2: Insufficient privilege minimization in the support portal function***

- *DigiCert's internal support portal includes a function that allows authenticated support analysts to proxy into specific customer accounts, to facilitate customer support. In this*

mode, certain functions are masked from the analyst. However, access to initialization codes for pending Code Signing certificate orders was not among the masked data elements, leaving those codes accessible to support a analyst operating in a proxied session.

The portal function had not been formally classified within DigiCert's privileged access management (PAM) framework. The definition of 'privileged access' was primarily scoped to direct access to CA and did not encompass this indirect account management function that had a path to certificate issuance. As a result, the portal function was not subject to the PAM controls applicable to privileged users under the CABF NetSec, including formal threat modeling against misuse scenarios, least-privilege design review, and access recertification.

- The portal function is a longstanding feature. On April 14th and 15th, following the discovery of the incident, we deployed a code change to mask initialization codes from proxied users on both our US and EU platforms using either the UI or API.
- The absence of this initialization code masking was identified during the investigation triggered by the third-party report on April 14th.
- Interaction with other factors: This factor defines the scope of the damage enabled by Contributing Factor #1, which was the lack of endpoint coverage. Without the EDR gap, the dwell time would have been minimal and the number of initialization codes accessible would have been limited. This factor also interacts with Contributing Factor #3 as the absence of masking is a direct consequence of the codes not being recognized as requiring credential-tier protection.

Which brings us to Contributing Factor #3:

### **Initialization codes not protected as bearer credentials**

- The EV Code Signing certificate pickup workflow was designed with a threat model that assumed initialization codes would **only** be accessible to the validated subscriber, delivered via a secure channel, and entered by the subscriber into their local Hardware Certificate Installer. The threat model did **not** account for the scenario in which initialization codes stored within DigiCert's internal support portal could be viewed by a compromised DigiCert analyst account operating through the portal function. Therefore, initialization codes were classified as intermediate workflow data rather than bearer credentials.
- This initialization code workflow was designed at the time the EV Code Signing token issuance process was developed. The issue was identified during the investigation triggered by the third-party report on April 14th. It was not identified through any previous design review prior to this incident.
- This factor made Contributing Factor #2 possible.

What we see here is the work of true forensic security professionals patiently working to understand, step by step, not only what happened, but why a supposedly carefully designed security system allowed this to happen. The result is guidance about what can be changed to prevent successful exploitation at each stage.

So this leaves us with the 4th and final Contributing Factor .... which explains how the bad guys managed to infect DigiCert's two analysts in the first place. They write:

### **Overly permissive file transfer capability in customer-facing support channels**

- *DigiCert's customer support chat channel and Salesforce case attachment workflow permitted delivery of inbound file attachments from external parties, including the general public, to CA support staff with insufficient restrictions on file type, automated sandboxing, and content inspection. This created a direct delivery path for malicious executable content to personnel with privileged access. The support chat channel had not been adequately evaluated as a potential attack surface for malware delivery against CA staff.*
- *The support chat channel and Salesforce case attachment integration were operational prior to this incident. The delivery vector was confirmed on April 4th and 5th, at which point additional malicious ZIPs were identified across other Salesforce cases and removed.*

*Corrective controls (file type restrictions, sandboxing evaluation) are in progress as described in the Action Items.*

- *The number of channels by which customers can reach support staff has grown. The file controls on the chat channel were believed to be sufficient prior to this incident.*
- *This factor is the initial attack vector that enabled all subsequent factors to come into play.*

And this brings us to the "Lessons Learned" conclusion with "what went well", "what did not go well", "where they got lucky" and a few final thoughts. So first we have "What went well":

- *Rapid initial containment on endpoint machines where EDR was working as intended. For these machines, DigiCert's Trust Operations team completed containment, process termination, registry cleanup, and artifact deletion within hours of detection on April 3rd.*
- *Proactive identification of the full delivery chain. The investigation identified the Salesforce case attachment auto-conversion mechanism as the delivery path and located additional malicious ZIP files across other cases before they could be opened, preventing further compromises from the same campaign.*
- *Same-day remediation of contributing vulnerabilities during incident response. Critical fixes were implemented without deferral: CrowdStrike prevention settings corrected on April 4th,; Okta FastPass disabled and MFA tightened on April 14th; initialization code masking deployed across US and EU environments on April 14th and 15th.*
- *Confident attribution of the second compromise through forensic analysis. Linking the compromised machines' activity logs to the same threat actor required analysis across identity events, endpoint telemetry, and support workflow artifacts.*
- *Revocations were completed within required timelines.*

So that's what went well. What did they feel did not go well?

- *File-type controls were insufficient on customer-facing support channels.*
- *Inconsistent and incomplete EDR coverage enabled a blind spot that was unknown prior to the incident and that directly enabled the attacker's dwell time.*
- *Initialization codes were not protected as bearer credentials. The portal function exposed initialization codes that are functionally equivalent to the certificates they enable, because they were classified as intermediate workflow data rather than credential material requiring masking and credential-tier protections.*
- *Device-bound authentication (Okta FastPass) was used as an MFA bypass. FastPass allowed a threat actor operating on a compromised device to inherit the device's authenticated session and satisfy MFA requirements without a genuine second factor, enabling access to the portal initialization codes after the initial endpoint compromise.*
- *Following containment of ENDPOINT1 on April 3rd, the investigation concluded that compromise attempts had been neutralized without validating all endpoints exposed through the same delivery vector.*

In retrospect, I imagine that they now wish that after that first attack, which was thwarted by CrowdStrike's endpoint defense, they had checked for other instances of that malicious ZIP file across the rest of their network. We know that they did that later and found a handful of other instances of it. It's sort of unclear why they did not do that at the time.

They share two points where they felt they got lucky:

- *A community member involved in security research reported the evolving pattern of misused certificates and engaged in dialogue with our support team. Without that report, the undetected compromise of ENDPOINT2 and the associated mis-issuance might have remained undiscovered for a longer period.*
- *Our investigation indicates that the threat actor's activity was focused on gaining access to Code Signing certificates. A differently-motivated threat actor might have attempted to use the compromised account for broader actions. Several of our action items are designed to address this risk.*

And they conclude with three points:

- *This incident demonstrates that internal support tooling with indirect paths to certificate issuance must be subject to the same security scrutiny as CA infrastructure. Tools that were designed for legitimate operational purposes can become high-value attack targets.*
- *The incident also illustrates the importance of defining 'privileged access' broadly enough to encompass any system or function with a path to certificate issuance, not just those with direct access to HSMs or signing infrastructure.*
- *The dwell time underscores the importance of comprehensive post-incident investigation scope and continuous EDR coverage monitoring. A single missed endpoint can negate the value of rapid containment elsewhere.*

They finally enumerate 21 individual Action Items, many of which they already articulated or implied. I'm not going to take everyone through each.

Suffice to say that there's little doubt that DigiCert was extremely unhappy and embarrassed by this event. Everything any Certificate Authority is about and a huge amount of security-focused design and 3rd-party contractor support, from the likes of Crowdstrike and several others, was intended to prevent anything like this from ever happening. But it did anyway.

However, at no point did we see what we have previously observed from so many other Certificate Authorities who have been struck by similar abuse. Almost invariably they first hoped that no one would notice. Then, when someone did, they would down-play the severity of the incident, hoping that would be it. Then when additional evidence of further exploit would surface they would apologize with some lame excuse about having intended to mention that too. Right.

What we have from DigiCert, the industry's largest commercial certificate authority, second only to Let's Encrypt, is full public disclosure and responsibility taking. The result has been a deep analysis followed by true action to prevent anything like this from transpiring again. As usual, I think that's more to recommend them than not. If their code signing certificates were not so unreasonably expensive I would not have invested so much time earlier this year preparing to jump ship and find another supplier. I'm glad I'll be moving to IdenTrust, but I'm only doing so because I object on principle to unconscionable costs for certificates, not the quality of their work.

But as for Microsoft...

Given what we know now, it's difficult to understand how Microsoft could have possibly fumbled their end of things so thoroughly. There would presumably have been some dialog between Microsoft and DigiCert, where DigiCert provided Microsoft with the thumbprints or serial numbers of the 60 certificates they had revoked and blacklisted. As we know, revocation is an imperfect answer within certificate management, but it's all we've got.

So Microsoft would have definitely wanted to add those certificates to Windows Defender's existing code signing deny list, so that nothing signed by them would have been allowed to run on Windows. But that just entails checking thumbprints against Defender's deny list. How Microsoft could possibly have fumbled this into the removal of DigiCert root certs is truly difficult to understand. It feels very much as though someone in control of that important process doesn't know what they're doing.

Fortunately, that too was fixable and was quickly fixed. So on we go to the next adventure!

